为 Web 开发提供的 10 个 Ruby on Rails Gems

在这里，值得推荐的 Ruby on Rails 的 web 开发的插件包括 Capistrano 和 Foreigner。

我用 Ruby on Rails 开发已经有很多年了，用这个优秀的框架我完成了很多任务，其中有些可以算是非常复杂的项目。基于我的经验，我来推荐一些我觉得非常有用的 gem。本文就会分享这个推荐列表，还会告诉你如何找到 RoR 有用的 gem。

不幸的是，gem 的格式描述不支持分类和标签。所以我们不得不期望 gem 的作者在描述里列出我们需要的关键字，这样在搜索的时候才能更好的得到结果。你可以在 rubygems.org 或者 github.com 上找到很多 gem。可以通过描述去进行搜索(GitHub 上你还可以在语言列表中选上 Ruby)

另一个值得介绍的是 Ruby Toolbox。它可以让你按分类和热度进行搜索。不过不要仅依赖这个工具，因为 Ruby Toolbox 的作者是手工添加新的 gem 的。

Foreigner

这个 gem 可以帮你创建表的外键，它非常容易使用。你只需要简单的把它加到 Gemfile，它会为你的 migrations 增加两个方法 :add_foreign_key 和 remove_foreign_key。当然，你也可以使用这两个方法在 create_table 和 change_table 的时候添加或删除 key。

假设我们需要添加一个 key，这个 key 从 comment 表指向 posts 表。我们可以这样做：

```
01 class CreateComments < ActiveRecord::Migration
02 def change
03 create_table :comments do |t|
04 # … t.references :post
05 # ...
06 t.foreign_key :posts
07 end
08 # …
09 end
10 end
```

这些方法有一些额外的选项，例如 name，column，dependent。你可以参考文档。

也许有人会认为，准确的说这个 gem 并非只适用于 Rails 的新版本，不过它确实是只有 4.2 以上的版本才支持的（其实之前已经有第三方的支持了）。总之，我认为这个 gem 属于最有用的 gem 列表之一。

项目的 GitHub 链接

letter_opener

事实上，一个简单而有用的 gem，它是保存文件夹电子邮箱的一个插件而不是发送它们。通过激活这个 gem，你必须设置 letter_opener 作为应用配置中的递交方式（例如在 config/enviroments/development.rb之中）

太好啦！现在所有发出去的消息将被存贮在 /tmp/letter_opener

文件夹里，并且新的电子邮箱被发送之后会在浏览器中预览。它简单且实际。

请链接到Github。

Kaminari

该 gem 允许你很容易的创建任一复杂性的 paginators 模块。Kaminari 支持几个ORMs（ActiveRecord，Mongoid，MongoMapper)以及模板引擎(ERB，Haml，Slim)。

Kaminari 并不嵌套基础类：如 array，hash，Object 以及 ActiveRecord::Base。

让我们开始使用 Kaminari 吧，把它放在 gem 文件中，已经足够啦。某些功能变成可用之后，例如：页，每个以及对齐。现在，你可以借助于Kaminari.paginate_数组方法，来轻松地把你的数组转换成一个分页模块，然后许多有用的分页功能将可以访问。

```
1 @paginatable_array = Kaminari.paginate_array(my_array_object).page(params[:page]).per(10)
```

默认的配置将被生成在 Kaminari· 配置初始化程序中。

default_per_page, max_per_page, max_pages - 这是一个简要的能被设置的选项列表。

除此之外，该分页的每个模块能被独立的配置。

```ruby
class User < ActiveRecord::Base
paginates_per 50
end
```

如果你需要定制 paginator，你可以通过运行一个生成器来创建模板：

```
% rails g kaminari:views default # -e haml - if you want to use  HAML template engine.
```

该模板将被创建于 app/views/kaminari/ 之中，现在，你可以轻松地编辑他们。

本地化(I18n)标签，主题和友好的 urls，以及其他有用的 gem 文档选项能被找到。

请链接到 Github

CarrierWave

使用 CarrierWave 让你能够从你的 RoR 应用程序中上传任何文件。所有你需要做的事情如下：

创建一个 uploader：

```
rails generate uploader ProductPhotoUploader
```

添加一些选项：

```ruby
class ProductPhotoUploader < CarrierWave::Uploader::Base
include CarrierWave::MiniMagick
storage :file
def store_dir
"product_images"
end
def extension_white_list
%w(jpg jpeg gif png)
end
def filename
Digest::SHA1.hexdigest(self.read) + File.extname(original_filename)
end
end
```

你已经可以使用 uploader 在存储器中保存文件了。

```ruby
uploader = ProductPhotoUploader.new
uploader.store!(my_file)
```

CarrierWave 还允许你在临时存储、文件系统、甚至是云空间中储存文件。

你可以将 CarrierWave uploader 连接到 AR（还有其他 ORM 适配器）模型，并通过在数据库中保存一条记录来存储你的文件：

```ruby
class Retailer < ActiveRecord::Base
mount_uploader :logo, RetailerLogoUploader
end
retailer = Retailer.new
File.open('logo.jpg') { |f| retailer.logo = f }
retailer.save!
retailer.logo.url # => '/url/to/logo.png'
```

此外，也可以改变上传图像的品质，对它进行剪裁，加密文件以及在运行中做更多的事情，在上传时 - 所有这一切都可以在文档中找到。

访问项目的 GitHub 链接。

Urlify

用于将 diacritic 字符串转换为 ASCII-safe URI 字符串的一个简单且易于使用的 gem。在安装好这个 gem 之后，你可以为任何东西调用 urlify 函数，它将会立即被转换为一个等价的兼容 URI。

```ruby
URLify.urlify('Kj?le Test') #=> kjoele_test
```

或：

```ruby
URLify.urlify('Kj?le Test', '-') #=> kjoele-test
```

访问项目的 GitHub 链接。

WickedPdf

WickedPdf 是一个用于将 html 生成为 PDF 的 RoR 插件。在添加这个 gem 之后，你需要做以下步骤：

创建初始化：

```
rails generate wicked_pdf
```

注册 mine-type：

```
Mime::Type.register "application/pdf", :pdf
```

考虑到这个 gem 使用了 wkhtmltopdf，所以应该在 gem 的设置中指定其路径。

```
WickedPdf.config = { exe_path: '/usr/local/bin/wkhtmltopdf' }
```

访问该项目的 GitHub 链接。

Countries

如果你面临着需要处理关于不同国家的信息的任务，这个 gem 将会提供你解决这个问题所需要的所有数据。有按国家名称和区域搜索，货币信息（符号，代码），各种形式的电话号码，坐标等功能。所有这一切需要你在安装它之后创建一个 country 对象（或者从一个全局 helper 中获取），然后你就可以得出必要的信息了：

```
c = ISO3166::Country.new('UA')
# or to use global helper  = Country['UA']
c.name      #=> "Ukraine"
c.alpha2#=> "UK" c.alpha3#=> "UKR"
c.longitude #=> "49 00 N"
c.latitude  #=> "32 00 E"
```

以下为关于如何获取货币信息的代码：

```
c.currency.code   #=> "UAH"
c.currency.symbol #=> "?"
c.currency.name   #=> "Hryvnia"
```

或者检查该国是否为欧盟成员：

```
c.in_eu? #=> false
```

以下为通过名称以及货币代码搜索国家的示例：

```
c = ISO3166::Country.find_country_by_name('Ukraine')
c = ISO3166::Country.find_country_by_currency('UAH')
```

访问该项目的 GitHub 链接。

CanCanCan

这个库让你可以很方便的在 RoR 中配置权限。使用它你可以很容易的实现对特定资源的访问的限制。其便利之处在于权限的管理是跟用户分离的，而且所有的权限都存储在一个单独的地方。这就是为什么你不用再控制器、视图和数据库请求中重复地做一些事情。

这个库需要在控制器中定义一个 current_user 方法。所以你首先要对身份认证信息进行配置(这次我用到了 Devise, 不过其它的gem也可以做到)。然后. Then, generate an ability:

```
rails g cancan:ability
```

使用 :read, :create, :update 和 :destroy 这些命令来指定用户权限 (can/cannot 方法)， (你可以在文档中找到更多)。

```
class Article::Ability
  include CanCan::Ability
  def initialize(user)
case user
  when Admin
    cannot :manage, :profile
    can :read, :all
  when Moderator
    can :manage, [Apartment, RoomPrice], { lessor_id: user.id }
    can :manage, Photo, { photographer_id: user.id }
    can :manage, Lessor, { id: user.id }
    can :manage, :profile
end
  end
end
```

之后，在视图中你可以使用 can? 和 cannot? 辅助方法来检查当前用户对于给定action的权限：

```
<% if can? :update, @article %> <%= link_to "Edit", edit_article_path(@article) %> <% end %>
```

你还可以使用 authorize! 来对控制器中的action进行权限认证：

```
def show
  @article = Article.find(params[:id])
  authorize! :read, @article
end
```

或者你也用到前置过滤器 load_and_authorize_resource， 它会加载资源同时尝试对其进行权限认证。

你可以使用下面的方式来catch由 CanCan::AccessDenied抛出的异常来处理有关权限认证方面的错误：

```
class ApplicationController < ActionController::Base
rescue_from CanCan::AccessDenied do |exception|
redirect_to root_url, :alert => exception.message
end
end
```

其它信息可以在 GitHub 上的文档中找到。

链接到 GitHub.

Formtastic

该 gem 提供很棒的 DSL 支持，让你可以很容易的构建出漂亮直观的基于语义的 rich form，它提供的 DSL 很容易上手：只需要在一个 semantic_form_for 代码块里列出所有的字段，然后就可以得到一个漂亮的 form：

```
<%= semantic_form_for @post do |f| %>
<%= f.inputs "Basic", :id => "basic" do %>
<%= f.input :title %>
<%= f.input :body %>
<% end %>
<%= f.inputs :name => "Advanced Options", :id => "advanced" do %>
<%= f.input :slug, :label => "URL Title", :hint => "Created automatically if left blank", :required => false %>
<%= f.input :section, :as => :radio %>
<%= f.input :user, :label => "Author" %>
<%= f.input :categories, :required => false %>
<%= f.input :created_at, :as => :string, :label => "Publication Date", :required => false %>
<% end %>
<%= f.actions do %>
<%= f.action :submit %>
<% end %>
<% end %>
```

你也可以使用嵌套的资源：

```
<%= semantic_form_for [@author, @post] do |f| %>
```

也支持嵌套的 form。你可以使用 f.semantic_form_for(Rails 风格)，不过 Formtastic 风格的写法更好看一些，你可以使用 :for 选项。

```
<%= semantic_form_for @post do |f| %>
<%= f.inputs :title, :body, :created_at %>
<%= f.inputs :first_name, :last_name, :for => :author, :name => "Author" %>
<%= f.actions %>
<% end %>
```

你可以很容易的更改 input 的行为：

```
class StringInput < Formtastic::Inputs::StringInput
def to_html
puts "this is my modified version of StringInput"
super
end
end
```

基于已有的input创建自己的input：

```
class FlexibleTextInput < Formtastic::Inputs::StringInput
def input_html_options
super.merge(:class => "flexible-text-area")
end
end
```

或者创建一个全新的 input 类型：

```
class DatePickerInput
include Formtastic::Inputs::Base
def to_html
# ...
end
end
```

使用方式：

```
:as => :date_picker
```

Formtastic 支持相当多的 input 类型 (select, check_boxes, radio, time_zone, datetime_select, range)，基本及高级的本地化，belongs_to，has_many和has_and_belongs_to_many 的关联以及其他很多特性，具体参考文档。

GitHub 的项目链接。

Capistrano

该工具允许在多个远程机器上通过 ssh 的并发来执行命令。此 gem 容易使用 DSL。它使您能够定义将应用于某些角色机器的任务，并且通过网关机器来支持隧道连接。

打开 gem 之后，你必须执行：

```
bundle exec cap install
```

```
通过配置创建文件夹
```

如果你将使用不同的环境，你必须增加 STAGES 参数，例如：STAGES = local, sandbox, qa, production。运行一个 cap 脚本，使用该命令-

bundle exec cap [environments separated by gaps] [command]。例如，将部署看起来就如此的舞台环境：

```
bundle exec cap staging deploy
```

Capistrano DSL 通过 Rake 来借用。以下是该任务的一个简单例子：

```
server 'example.com', roles: [:web, :app]
server 'example.org', roles: [:db, :workers]
desc "Report Uptimes"
task :uptime do
on roles(:all) do |host|
execute :any_command, "with args", :here, "and here"
info "Host #{host} (#{host.roles.to_a.join(', ')}):\t#{capture(:uptime)}"
end
end
```

参阅所有可能的参数文档，更多关于定义任务的详细细节，请连接该插件和其他东西。

链接到Github。

好啦，我们来回顾十个最有用的 Ruby on rails gems，该是我的故事将要大结局了，我希望你能找到这个有用的信息。在我与 Evgeniy Maslov 合作而发表的“Ruby on rails 之中的 eway 支付网关集成”文章里，你将阅读更多关于 Ruby on rails。谢谢你们，再见了，亲爱的读者。

本文地址：http://www.oschina.net/translate/10-ruby-on-rails-gems-for-web-development
原文地址：https://dzone.com/articles/10-ruby-on-rails-gems-for-web-development

---