

Rails 学习笔记

- [Blog](#)
- [Archive](#)
- [RSS](#)
- [About Me](#)

如何通过阅读习得技能

能够靠阅读习得技能，真的是一件很酷的事情。想想看吧，在书价相对更便宜，电子书更加流行的时候，不用四处奔波去上课，只要有一台能上网的电脑就可以开始学习，想想就觉得幸福。

如何阅读？

首先，要只字不差的阅读，不懂的地方反复阅读，直到弄懂为止。

接着，一本书 只要必要 即便读不懂 也要读完。硬着头皮反复读，要相信自己的大脑有成长能力。

如何习得只字不差的阅读能力？

开始写作。阅读目标常常很宽泛比如说改变自己的观念，而写作目标反过来却常常非常具体，比如说清楚一个观念。所以有写作目标的时候，往往更容易做到仔细阅读和深入理解。

思考并践行

假如认同读到的某个道理，那么就认真思考如下问题：

过去我有哪些地方做错了需要改正？

将来根据学到的知识原则，我要做出哪些改变？

然后照着这些原则去做，把新习得的知识固化到自己的生活中去。

参考

[李笑来：我的读书经验](#)

September 28, 2016

[本周周记](#)

隐含的超高要求

即要求技术好、又会统筹规划、又善于沟通。嗯 这三个集齐的都是小概率事件。往往你的同事只有一条的，这个时候你该怎么办？

尝试转换视角

当下努力当下就有收益，基本是小概率事件。花一份心力努力，花两份心力焦虑别人的看法，不如三份都花去努力，等待长期的回报。你不用被患得患失整得死去活来，可以更专注于做好事情，且永恒有效。

改变

改变是一件很诡异的事情。有时候并不需要你换一个人，深度反思与忏悔。只需要热情或假装热情地说“哇～～你好厉害啊！”“这个好好吃，你在哪儿买的”“这个我会做耶！”就好了。WTF...

September 11, 2016

[Debug 思路：描述发生了什么？](#)

当我将程序 push 到 heroku，之后运行 heroku run rake db:migrate 时出现了以下错误：

rake aborted! NameError: uninitialized constant CarrierWave::Uploader::Base::Fog

查了下程序里面并没有 CarrierWave::Uploader::Base::Fog，Google 了错误资讯也并没有直接的解决方式。

我开始想一定是我 push heroku 的姿势不对，隔天再试了一下，还是同样的错误提示。

好了，我开始尝试[去 stackoverflow 上发问](#)，描述错误发生的情境，以及相关涉及到的文件 carrierwave.rb 和 Gemfile，突然发现 Gemfile 里面居然没有 gem "fog"，但是 carrierwave.rb 里面有一行 config.fog_credentials。

问了班长才知道，我们 image-upload 不再用 gem "fog"，换用 gem "carrierwave-aws"。然后根据 [carrierwave-aws](#) 的文档，把 config.fog_credentials 替换成 config.aws_credentials，再 push heroku 就可以了。

体会

stackoverflow 上面的活跃度极其高，问题贴上去不到 1 个小时就有人回复。

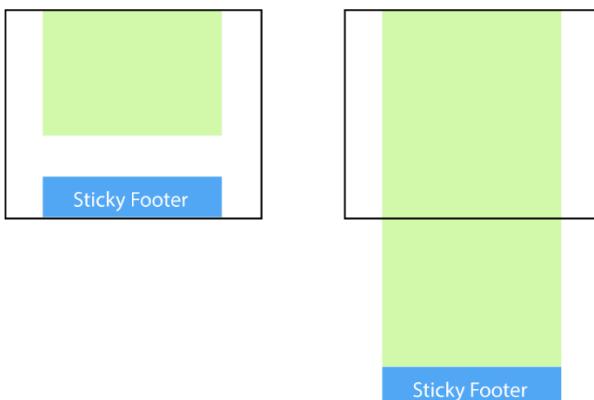
尝试清晰地描述在做什么的时候，发生了什么，相关的文件是什么样的，有助于解题。

September 10, 2016

[Sticky Footer](#)

什么是 Sticky Footer

如下图，内容不够的时候，sticky footer 在窗口最下面；内容足够的时候，sticky footer 在内容最下面。



方法 1 wrapper 里 负的 margin-bottom

用 wrapper 元素包住所有的元素，除了 footer。wrapper 有一个为负值的页边来等于 footer 的高度。

```
<body>
```

```
  <div class="wrapper">
    content
    <div class="push"></div>
  </div>
  <footer class="footer"></footer>
</body>
```

```
html, body{
```

```
  height: 100%;
  margin: 0;
}
```

```
.wrapper {
  min-height: 100%;
  /* Equal to height of footer */
  /* But also accounting for potential margin-bottom of last child */
  margin-bottom: -50px;
}

.footer, .push{
  height: 50px;}
```

这种方式需要额外的元素放在 content 区域，就是 push，以确保负值的页边不会把 footer 拉高盖住内容。

push 也很聪明的另外一点儿就是它很大可能会自己没有底部边距。这样的话，就必须考虑到负值边距，让这 2 个值不同看起来不会那么好。

也可以换另一种写法：

```
<body>
  <div class="wrapper">
    content
  </div>
  <footer class="footer"></footer>
</body>
html, body {
  height: 100%;
  margin: 0;
}
.wrapper {
  min-height: 100%;
  margin-bottom: -50px;
}
.wrapper:after {
  content: "";
  display: block;
}
.footer, .wrapper:after {
  height: 50px;
}
```

方法 2 footer 上 负的 margin-top

额外的元素 content-inside 来包裹内容，应用和 footer 等高的 padding-bottom 来避免负值页边把 footer 拉高遮住内容。

```
<body>
  <div class="content">
    <div class="content-inside">
      content
    </div>
  </div>
  <footer class="footer"></footer>
</body>
html, body{
  height: 100%;
  margin: 0;
}
.content{
  min-height: 100%;
}
```

```
.content-inside{  
    padding: 20px;  
    padding-bottom: 50px;  
}  
.footer {  
    height: 50px;  
    margin-top: -50px;  
}
```

方法 3 用 calc() 减小 wrapper 高度

用 calc() 来调整 wrapper 高度，这样就不需要有额外的元素。

```
<body>  
    <div class="content">  
        content  
    </div>  
    <footer class="footer"></footer>  
</body>  
.content{  
    min-height: calc(100vh - 70px);  
}  
.footer {  
    height: 50px;  
}
```

注意 calc() 里的 70px 和 footer 的高度 50px。这里预先有一个假设，假设内容里最后一个元素的 margin-bottom 为 20px。实际要从 calc() 里减去的是最后一个元素的 margin-bottom 和 footer 高度的总和。

方法 4 运用 flexbox

以上 3 种方法都需要 footer 固定高度，但实际的 web 中内容会灵活改变，固定 footer 高度通常会有问题。
运用 flexbox 不仅不需要额外的元素，还允许 footer 高度改变。

```
<body>  
    <div class="content">  
        content  
    </div>  
    <footer class="footer"></footer>  
</body>  
html {  
    height: 100%;  
}  
body {  
    min-height: 100%;  
    display: flex;  
    flex-direction: column;  
}  
.content {
```

```
flex: 1;
```

```
}
```

你甚至可以在上面增加 header 或在下面增加更多东西。flexbox 的技巧：

- flex: 1 用在你想要拉长来适配空间的元素上（这里我们用于 content）。
- 或 margin-top: auto 用在你想要推得离相邻元素要多远有多远的元素上（或者无论哪个方向）。

也可以参考 [flexbox 详细教程](#)。

方法 5 运用 grid

Grid 布局比 flexbox 更新，可以参考 [grid 详细教程](#)。

```
<body>
```

```
  <div class="content">
```

```
    content
```

```
  </div>
```

```
  <footer class="footer"></footer>
```

```
</body>
```

```
html {
```

```
  height: 100%;
```

```
}
```

```
body {
```

```
  min-height: 100%;
```

```
  display: grid;
```

```
  grid-template-rows: 1fr auto;
```

```
}
```

```
.footer {
```

```
  grid-row-start: 2;
```

```
  grid-row-end: 3;
```

```
}
```

这个例子可以用在 Chrome Canary 或者 Firefox 开发版上，可能在 Edge 上被降级到旧的 grid 布局版本。

参考

[Sticky Footer, Five Ways - CSS Tricks](#)

[Sticky Footer -Chris Coyier](#)

September 9, 2016

[Rails 中.nil? .empty? .any? .blank? .present?的区别](#)

这 5 个特别容易混淆的 method。

.nil ?

- Ruby method
- 可以用于任何 object 上，只有当 object 是 nil 时，值为 true。

```
nil.nil?      #true
```

```
[] .nil?      #false
```

```
{} .nil?      #false
```

```
"".nil?      #false  
" ".nil?     #false  
"abc".nil?   #false  
  
123.nil?    #false  
.empty?

- Ruby method
- 可以用于 strings、arrays 和 hashes 并且返回 true, 前提是：
  - String length == 0
  - Array length == 0
  - Hash length == 0
- 在值为 nil 的 object 上引用.empty?, 返回 NoMethodError。
- 相对的 method 是.any?

  
nil.empty?   #NoMethodError: undefined method 'empty?' for nil:NilClass
```

```
[] .empty?    #true  
{} .empty?    #true  
  
"".empty?    #true  
  
" ".empty?   #false, 空白不算空值
```

```
"abc".empty? #false  
  
123.empty?   #NoMethodError: undefined method 'empty?' for 123:Fixnum  
.any?

- Ruby method
- 可以用于 arrays 和 hashes, 不可用于 strings。
- 相对的 method 是.empty?

  
nil.any?     #NoMethodError: undefined method 'any?' for nil:NilClass
```

```
[] .any?      #false  
{} .any?      #false  
  
"".any?      #NoMethodError: undefined method 'any?' for """:String  
  
" ".any?     #NoMethodError: undefined method 'any?' for """:String
```

```
"abc".any?      #NoMethodError: undefined method 'any?' for """:String  
  
123.any?      #NoMethodError: undefined method 'any?' for 123:Fixnum  
.blank?  
• Rails method  
• 用于任何 object 上，只要是 nil 或空值都是 true。  
• 相对的 method 是.present? !obj.blank? == obj.present?  
nil.blank?    #true  
  
[].blank?     #true  
  
{}.blank?     #true  
  
"".blank?     #true  
  
" ".blank?    #true  
  
"abc".blank?  #false  
  
123.blank?   #false  
.present?  
• Rails method  
• 用于任何 object 上，只要是 nil 或空值都是 false。  
• 相对的 method 是.blank? !obj.blank? == obj.present?  
nil.present?  #false  
  
[].present?   #false  
  
{}.present?   #false  
  
"".present?   #false  
  
" ".present?  #false  
  
"abc".present? #true  
  
123.present?  #true
```

参考
[.nil? .empty? .blank? .present? in Ruby on Rails - Quora](#)
[.nil? .empty? .blank? .present? 傻傻分不清楚？ - Leon](#)

September 8, 2016

[Rails 应用如何实现 Email 发送](#)

一、区分 3 类邮件

- 1) Company Email
- 2) Transactional Email
- 3) Mailing list

二、配置 Transcational Email

配置 Devise

配置 ActionMailer

配置邮件服务商

Mandrill

SendGrid

Gmail

参考

一、区分 3 类邮件

一般应用里，你需要设置如下 3 类邮件。

1) Company Email

意思就是，你用你的公司邮箱给供应商/客户/同事等发的邮件，这时候你可以用你的 Gmail 解决掉这个问题，设置你的 Gmail 代收公司邮箱，然后用 Gmail 回复时，再选择从不同地址发送邮件。当然你也可以选择更专业版的 [Google Apps for Business](#)。

2) Transactional Email

意思就是，从应用中发出的邮件，就是用户行为触发的邮件，比如邀请注册、感谢注册、验证邮箱、修改密码、回复提醒、订单生成等。这时候你需要有一个邮件服务商("email service provider")，来提供 **SMTP 转发服务**。你需要专业的邮件服务商来避免你的邮件被作为垃圾邮件过滤掉、提升邮件送达稳定性、追踪邮件送达状况。

大多数邮件服务商都支持每天免费发 200 封邮件：

- [Mandrill by MailChimp](#)
- [SendGrid](#)
- [Mailgun](#)
- [Amazon Simple Email Service](#)
- [Elastic Email](#)
- [CritSend](#)
- [Mailjet](#)
- [MessageGears](#)
- [PostageApp](#)
- [Postmark](#)
- [turboSMTP](#)

3) Mailing list

意思就是，你对所有列表用户发送的新闻或通知或营销活动等。上面提到的 **SMTP 转发服务**比如 SendGrid 通常只提供基本的转发服务，这时候你需要去考虑更加专业的邮件服务来实现群发功能、管理未订阅者、管理不同的邮件模版等。

群发邮件，可以选择以下邮件服务商：

- [MailChimp](#)

- [MadMimi](#)
- [CampaignMonitor](#)
- [Constant Contact](#)
- [YMLP](#)
- [JangoMail](#)
- [iContact](#)
- [VerticalResponse](#)

二、配置 Transcational Email

第 1 类和第 3 类邮件，基本借助第三方服务就可以了。只有第 2 类，让你的应用发送邮件，需要同时配置应用和邮件服务商才能生效。

配置 Devise

假如你有使用 Devise 的 confirmable 模块来发验证邮件，修改下面的文件，填写你的发件人邮件地址：

config/initializers/devise.rb

```
config.mailer_sender = "sender@yourwebsite.com"
```

配置 ActionMailer

修改 production 环境里面的文件，填入如下代码，注意把 default_url 改成你自己的网站：

config/environments/production.rb

```
config.action_mailer.default_url_options = { :host => 'yourwebsite.com' }
```

```
# ActionMailer Config
```

```
# Setup for production - deliveries, no errors raised
```

```
config.action_mailer.delivery_method = :smtp
config.action_mailer.perform_deliveries = true
config.action_mailer.raise_delivery_errors = false
config.action_mailer.default_charset => "utf-8"
```

修改下面的文件，填写你的发件人邮件地址：

app/mailers/application_mailer.rb

```
class ApplicationMailer < ActionMailer::Base
  default from: "sender@yourwebsite.com"
  layout "mailer"
end
```

配置邮件服务商

实现 SMTP 转发功能。这里以 Mandrill, SendGrid 和 Gmail 来示范配置，实际是三种任选一种就可以了。注意，Gmail 有限制每天邮件上限 500 封，超过数量 Gmail 会被 Google 禁用 24 小时，除非你换用更专业版的 [Google Apps for Business](#)。

修改 production 环境里面的文件，填入如下代码，注意要先创建 Mandrill, SendGrid 和 Gmail 帐号和密码：

config/environments/production.rb

Mandrill

```
config.action_mailer.smtp_settings = {
  :address    => "smtp.mandrillapp.com",
  :port       => 25,
```

```
:user_name => ENV["MANDRILL_USERNAME"],  
:password  => ENV["MANDRILL_API_KEY"]  
}
```

SendGrid

```
config.action_mailer.smtp_settings = {  
address: "smtp.sendgrid.net",  
port: 25,  
domain: "heroku.com",  
authentication: "plain",  
enable_starttls_auto: true,  
user_name: ENV["SENDGRID_USERNAME"],  
password: ENV["SENDGRID_PASSWORD"]  
}
```

Gmail

```
config.action_mailer.smtp_settings = {  
address: "smtp.gmail.com",  
port: 587,  
domain: "example.com",  
authentication: "plain",  
enable_starttls_auto: true,  
user_name: ENV["GMAIL_USERNAME"],  
password: ENV["GMAIL_PASSWORD"]  
}
```

参考

[Send Email with Rails - Daniel Kehoe](#)

[How to get Devise, Figaro, Heroku and emailing to Play Together Nicely -](#)

[Aimee Knight](#)

September 7, 2016

第六週週記

1.套 html/css 模版排了首页和 sidebar 的布局。澜心选的图片超级美，P 上个 Rails 感觉图成了专门定制的，很聪明的小技巧，◎。还有就是图片超级难选，前 2 张定了的话，第 3 张更难选。

2.用 Heroku 上的插件 SendGrid 实现了在线发送验证邮件的功能。

September 5, 2016

CSS 冲突解决

CSS 冲突。刚排好一个页面，再叠加上另一个页面，然后同时只有一个正常。

CSS 并没有冲突的概念

首先 CSS 并没有 conflict (冲突) 的概念，它有 cascade (瀑布式)。意思是你可以定义多个规则来应用到同样的元素上，你放置 styles 的顺序反应出一种权重 (important)，比如如果它最后出现，它会覆盖前面的规则，也就是你说的冲突。

3 种解决方式

方式 A

1) 确定哪个 style sheet 是最重要的，把它放在第一位

- 2) 重写你的 styles 来避免混乱。
- 3) 标记重要的规则为!IMPORTANT

方式 B

[Firefox 插件 Dust-Me](#)。把所有的 styles 放在一个文件里，然后再用插件 Dust-Me 检查，移除不用的 styles。

方式 C

[Firefox 插件 Firebug](#)。查看你渲染页面的每一个元素。在"CSS"标签页查看元素的 style rules 的 cascade（瀑布式），来自于哪个 CSS 源文件。它也会向你显示被重写的规则。

Good luck~

参考

- 1.[conflict - Systematically resolve conflicting styles in css - Stack Overflow](#)
- 2.[Resolving conflicts in CSS made simple | Web TeacherWeb Teacher](#)
- 3.[Selectutorial: What happens when conflicts occur? - CSS Maxdesign](#)
- 4.[CSS: resolving conflict - an introduction by Graham Mansfield](#)

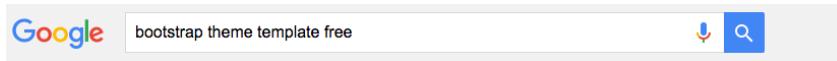
September 5, 2016

CSS 模版套用

CSS 模版套用。

[Google 搜索 bootstrap theme template free](#)

全世界的美好资源都奔涌而来 ~



全部 图片 视频 新闻 购物 更多 搜索工具

找到约 3,260,000 条结果 (用时 0.62 秒)

All Free Bootstrap Themes & Templates - Start Bootstrap

<https://startbootstrap.com/template-categories/all/> 翻译此页
Start Bootstrap's entire collection of free Bootstrap themes and templates. All of our themes are open source and free to download.
Creative · Stylish Portfolio · Agency · Landing Page

Start Bootstrap - Free Bootstrap Themes and Templates

<https://startbootstrap.com/> 翻译此页
Free open source Bootstrap themes and templates up to date for Bootstrap 3! An even faster way to develop websites in Bootstrap!

Bootstrap Themes at BootstrapZero

<https://www.bootstrapzero.com/> 翻译此页
Download free, responsive Bootstrap themes for Bootstrap 3 or Bootstrap 4. ... Download hundreds of free and premium bootstrap themes and templates.
Free bootstrap themes and ... · Admin · Just One Page · Clean Bootstrap Template

Premium Quality Free Bootstrap Templates & Themes | ShapeBootstrap

<https://shapebootstrap.net/free-templates/> 翻译此页
Himu - Free Responsive Bootstrap Template ... In stock. Cluster - Free Creative Portfolio
Bootstrap Template ... In stock. Sept - Free Bootstrap 3 Theme.

10 Free Responsive Bootstrap Templates For 2016 | Almsaeed Studio

<https://almsaeedstudio.com/.../10-Free-Responsive-Bootstrap-Templates-For-...> 翻译此页
2016年1月21日 - 10 Free Responsive Bootstrap Templates For 2016 ... I got excited to test out the Boot 4 theme until I realized that I had to whip out a debit card.

70+ Free Bootstrap HTML5 Website Templates - freshDesignweb

<https://www.freshdesignweb.com/free-bootstrap-templates/> 翻译此页
2016年8月28日 - is a free bootstrap base HTML template suitable for digital and agency website. It is also responsive, and user friendly theme. Code by Jenn ...

Free Bootstrap Templates - Awe-inspiring Bootstrap themes by ...

bootrapious.com/free-templates 翻译此页
We usually release one or two Free Bootstrap templates per month. At the moment there are 13 Free and Premium Bootstrap themes and templates carefully ...

Free Bootstrap Themes & Templates for Responsive HTML5 Websites

<https://themewagon.com/themes/> 翻译此页
Hand Picked Collection of the Best Free Responsive Bootstrap Themes & Premium Quality Themes.
Download bootstrap templates free.

25 Best Free WordPress Themes Built With Bootstrap 2016 - Colorlib

<https://colorlib.com/wp/free-bootstrap-wordpress-themes/> 翻译此页
2016年8月26日 - Activello has been fully decked out with all the page templates, layout ... Dazzling is a beautiful Free WordPress theme based on Bootstrap 3, ...

September 5, 2016

Debug 思路：要什么给什么

在写 sidebar 的时候，我需要遍历整个 chapters 生成动态的目录页，也就是在 post 的 show 页面，我要能够使用 chapters。这一点我一直不知道怎么办，我只知道我显示 post，只有一个 chapter_id 可以用。因为我设定的 post 的 show 路径是 chapters/chapter_id/posts/id。

学霸说它要 chapters 你给它 chapters。WTF，我没有 chapters 我怎么给啊？！

好吧，写一个，给 show 写一个获取 chapters 的方法，就可以了，就可以了。我要跪了。这样调用 show 的时候就有了类似 index 那样的 chapters。

posts_controller.rb

```
before_action :get_chapters, only:[:show]
```

```
def get_chapters
```

```
    @chapters = Chapter.all
```

```
end
```

```
def show
  @chapter = Chapter.find(params[:chapter_id])

  if @post.is_hidden? || @chapter.is_hidden?
    flash[:warning] = "The content is archived"
    redirect_to root_path
  end

end
```

August 30, 2016

本周周记

看了 n 个教程，写出了注册用户发送验证邮件。实际的开发环境更复杂，有可能教程只针对作者的环境有用，但可能不符合你的环境，你要多参考几份，再根据报错去寻找解决方案。

解决了一个 bug，然后又来了 2 个 bug。老师说人生就是这样。😊。

开始每天学习 Rails 相关理论。通看了 [Rails102](#)。在跟 [Code School 上的 Rails 课程](#)。理顺一下我一直在用的代码都是什么意思。

同学的帮助。提前问了学霸，获得了关键点。迅速改出了进度条功能，感觉自己像个天才。往常我都是自己 debug 到快要崩溃的时候才找同学帮忙，这样往往已经耗费大量的时间。

August 28, 2016

本週遇到最大的坑

仔细看注释，说不定你的答案在注释里已经说明了。

db 贴了官方代码 [How To: Add :confirmable to Users - Devise](#)，还是提示异常 SQLite has no such function :NOW.，google 了一下也没找到答案，被卡了一下，后来突然发现，注释里说得很明显，把代码替换成 execute("UPDATE users SET confirmed_at = date('now')")就可以了。

```
# Remind: Rails using SQLite as default. And SQLite has no such function :NOW.
```

```
# Use :date('now') instead of :NOW when using SQLite.
```

```
# => execute("UPDATE users SET confirmed_at = date('now')")
```

How To: Add :confirmable to Users - Devise

```
class AddConfirmableToDevise < ActiveRecord::Migration
```

```
  # Note: You can't use change, as User.update_all will fail in the down migration
```

```
def up
```

```
  add_column :users, :confirmation_token, :string
```

```
  add_column :users, :confirmed_at, :datetime
```

```
  add_column :users, :confirmation_sent_at, :datetime
```

```
  add_column :users, :unconfirmed_email, :string # Only if using reconfirmable
```

```
  add_index :users, :confirmation_token, unique: true
```

```
  # User.reset_column_information # Need for some types of updates, but not for update_all.
```

```

# To avoid a short time window between running the migration and updating all existing

# users as confirmed, do the following

execute("UPDATE users SET confirmed_at = NOW()")
# All existing user accounts should be able to log in after this.

# Remind: Rails using SQLite as default. And SQLite has no such function :NOW.

# Use :date('now') instead of :NOW when using SQLite.

# => execute("UPDATE users SET confirmed_at = date('now')")

# Or => User.all.update_all confirmed_at: Time.now

end

def down
  remove_columns :users, :confirmation_token, :confirmed_at, :confirmation_sent_at, :unconfirmed_email
  # remove_columns :users, :unconfirmed_email # Only if using reconfirmable

end
end

August 26, 2016
插件 Octotree - Github 在线读代码
支持 Chrome, Firefox, Opera 和 Safari


- 安装链接 Chrome Web Store, Mozilla Add-ons Store, Opera Add-ons Store 或 Safari prebuilt package(需要双击安装)
- 支持 GitHub 或 GitLab 上代码
- 代码树在左侧面显示
- 开源软件：https://github.com/buonguyen/octotree/

```

The screenshot shows a GitHub repository page for 'buunguyen / octotree'. On the left, there's a tree view of the repository's directory structure. In the center, a commit for 'jquery-ui.js' is displayed. The commit message is: 'Update jquery-ui to latest version and bump to release to Mozilla store'. It was made by 'buunguyen' on Dec 21, 2015. The code snippet shows the beginning of the jQuery UI source code, which includes a license notice and AMD registration logic.

```

1 /*! jQuery UI - v1.11.4 - 2015-12-20
2 * http://jqueryui.com
3 * Includes: core.js, widget.js, mouse.js, resizable.js
4 * Copyright jQuery Foundation and other contributors; Licensed MIT */
5
6 (function( factory ) {
7     if ( typeof define === "function" && define.amd ) {
8         // AMD. Register as an anonymous module.
9         define( "jquery" , factory );
10    } else {
11        // Browser globals
12        factory( jQuery );
13    }
14 })(function( $ ) {
15
16 })

```

August 26, 2016

Onboarding 和养成习惯

今天 Xdite 讲了 Onboarding，顺便讲了 2 个有意思的点：

1. Onboarding 其实就是让用户养成使用你服务的习惯。
2. 她发现教 Ruby on Rails 时，有哪些习惯的同学更容易学好？

如何养成习惯

事先知道会发生什么事情

良好的体验（中间尽量去除挫折）

超乎预期的服务，希望重复

如何变成学霸

上课专心听，回去重复练习

有预习，有做功课

有写共笔

有来 meetup

August 26, 2016

在 Devise 上实现用户注册邮箱验证功能

功能实现

Step 1 修改 Model

Step 2 创建新的 Migration

Step 3 创建 Views

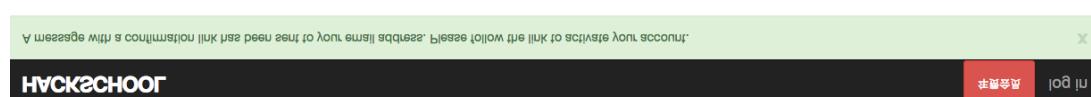
Step 4 创建 Controller

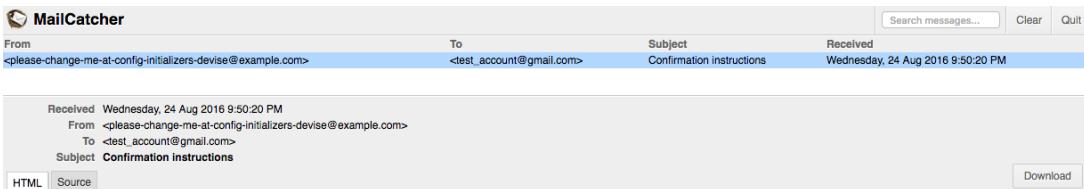
Step 5 配置环境 & 配置 MailCatcher

参考

功能实现

- 在 Devise 上实现邮箱验证功能：注册 - 提示验证邮件已发出 & 发送验证邮件 - 验证邮件 - 登入成功。
- 运用 Mailcatcher 在本地查看验证邮件。





Log in

Email

Password

Remember me



Step 1 修改 Model

首先默认你已经用 devise 实现了用户 signup/login/logout 的功能。添加 devise :confirmable 到 models/user.rb。

```
devise :registerable, :confirmable
```

```
config/initializers/devise.rb
```

```
config.reconfirmable = true
```

Step 2 创建新的 Migration

```
rails g migration add_confirmable_to_devise
```

在生成的 db/migrate/YYYYMMDDxx_add_confirmable_to_devise.rb, 填入如下代码。

```
class AddConfirmableToDevise < ActiveRecord::Migration
```

```
  # Note: You can't use change, as User.update_all will fail in the down migration
```

```
  def up
```

```
    add_column :users, :confirmation_token, :string
```

```
    add_column :users, :confirmed_at, :datetime
```

```
    add_column :users, :confirmation_sent_at, :datetime
```

```
    add_column :users, :unconfirmed_email, :string # Only if using reconfirmable
```

```
    add_index :users, :confirmation_token, unique: true
```

```
    # User.reset_column_information # Need for some types of updates, but not for update_all.
```

```

# To avoid a short time window between running the migration and updating all existing

# users as confirmed, do the following

execute("UPDATE users SET confirmed_at = NOW()")
# All existing user accounts should be able to log in after this.

# Remind: Rails using SQLite as default. And SQLite has no such function :NOW.

# Use :date('now') instead of :NOW when using SQLite.

# => execute("UPDATE users SET confirmed_at = date('now')")

# Or => User.all.update_all confirmed_at: Time.now

end

def down
  remove_columns :users, :confirmation_token, :confirmed_at, :confirmation_sent_at, :unconfirmed_email
  # remove_columns :users, :unconfirmed_email # Only if using reconfirmable
end

注意 execute("UPDATE users SET confirmed_at = NOW()") 使用 SQLite 可能会有异常，可以根据注释中的提示把这条指令改为 execute("UPDATE users SET confirmed_at = date('now')")。
跑 rake db:migrate, 然后重启服务器。

Step 3 创建 Views
跑如下指令生成 Devise views。
rails generate devise:views users

Step 4 创建 Controller
创建 app/controllers/confirmations_controller.rb
class ConfirmationsController < Devise::ConfirmationsController
  # GET /resource/confirmation/new

  def new
    self.resource = resource_class.new
  end

  # POST /resource/confirmation

  def create

```

```

self.resource = resource_class.send_confirmation_instructions(resource_params)
yield resource if block_given?

if successfully_sent?(resource)
  respond_with({}, location: after_resending_confirmation_instructions_path_for(resource_name))
else
  respond_with(resource)
end
end

# GET /resource/confirmation?confirmation_token=abcdef

def show
  self.resource = resource_class.confirm_by_token(params[:confirmation_token])
  yield resource if block_given?

  if resource.errors.empty?
    set_flash_message!(:notice, :confirmed)
    respond_with_navigational(resource){   redirect_to     after_confirmation_path_for(resource_name, resource) }
  else
    respond_with_navigational(resource.errors, status: :unprocessable_entity){ render :new }
  end
end

protected

# The path used after resending confirmation instructions.

def after_resending_confirmation_instructions_path_for(resource_name)
  is_navigational_format? ? new_session_path(resource_name) : '/'
end

# The path used after confirmation.

def after_confirmation_path_for(resource_name, resource)
  if signed_in?(resource_name)
    signed_in_root_path(resource)
  else
    new_session_path(resource_name)
  end
end

```

```
def translation_scope
  'devise.confirmations'
end
创建 app/controllers/registrations_controller.rb
class RegistrationsController < Devise::RegistrationsController
# POST /resource

def create
  build_resource(sign_up_params)
  if resource.save
    # this block will be used when user is saved in database

    if resource.active_for_authentication?
      # this block will be used when user is active or not required to be confirmed

      set_flash_message :notice, :signed_up if is_navigational_format?
      sign_up(resource_name, resource)
      respond_with resource, :location => after_sign_up_path_for(resource)
    else
      # this block will be used when user is required to be confirmed

      user_flash_msg if is_navigational_format? #created a custom method to set flash message

      expire_session_data_after_sign_in!
      respond_with resource, :location => after_inactive_sign_up_path_for(resource)
    end
  else
    # this block is used when validation fails

    clean_up_passwords resource
    respond_with resource
  end
end

private

# set custom flash message for unconfirmed user

def user_flash_msg
  if resource.inactive_message == :unconfirmed
```

```
#check for inactive_message and pass email variable to devise locals message

    set_flash_message :notice, :"signed_up_but_unconfirmed", email: resource.email
else
    set_flash_message :notice, :"signed_up_but_#{resource.inactive_message}"
end
end
end
```

Step 5 配置环境 & 配置 Mailcatcher

config/environments/development

```
config.action_mailer.delivery_method = :smtp
```

```
config.action_mailer.smtp_settings = { :address => "localhost", :port => 1025 }
```

```
config.action_mailer.default_url_options = { :host => 'localhost:3000' }
```

安装 Mailcatcher，抓取 Devise 注册邮件。

```
gem install mailcatcher
```

在 terminal 里输入 mailcatcher，运行 Mailcatcher。

```
mailcatcher
```

打开 http://localhost:3000/users/sign_up，注册新用户，然后打开 <http://localhost:1080>，在本地查看验证邮件。

参考

[How To: Add :confirmable to Users - Devise](#)

[Add Email Confirmations to Your Rails App](#)

[ConfirmationsController - Devise](#)

[Devise: Pass unconfirmed email address back to sign in page](#)

[Rails 4 how to: User sign up with email confirmation in five minutes, using Devise and Mailcatcher](#)

[MailCatcher](#)

August 24, 2016

本周最大的收获

朋友是必需品。当我情绪化，进入钻牛角尖状态。能够感知但很难一下接受时，有个小伙伴告诉我“每个人都有自己特别艰难的点，熬过去就是成长。加油！”终于可以平静接受，并尝试转换思路。

August 21, 2016

本周最大的坑

开发比我想象的要缓慢。我以为我 6 点可以写完，我 10 点才写完。

August 21, 2016

本周周记

思考团队合作。大家都度过了不愉快的一周。从刚开始生气吵翻天，团队可笑的解决方式，到平静下来想想问题都是可以解决的，不知道为什么当时吵成那样，事情为什么会变成这一步，想想还是挺搞笑的。

August 21, 2016

Nested routes 的使用

有时候需要设计一个嵌套路径 <http://localhost:3000/courses/1/chapters/1/posts/1>，来显示 course1 的 chapter1 的 post1。

Routes

```

resources :courses do
  resources :chapters do
    resources :posts
  end
end
Model
建立 course, chapter 和 post 之间的关联。
class Course < ApplicationRecord
  has_many :chapters
end
class Chapter < ApplicationRecord
  has_many :posts
  belongs_to :course
end
跑 rails g migration AddCourseIdToChapters course_id:integer, 然后再 rake db:migrate。
class Post < ApplicationRecord
  belongs_to :chapter
end
跑 rails g migration AddChapterIdToPosts chapter_id:integer, 然后再 rake db:migrate。
Controller
class PostsController < ApplicationController
  before_action :get_chapter

  def get_chapter
    @chapter = Chapter.find(params[:chapter_id])
  end

  def index
    @posts = @chapter.posts
  end

  def show
    @post = @chapter.posts.find(params[:id])
  end
end
View
app/views/posts/show.html.erb
<h1><%= @post.title %><h1>
<p><%= @post.description %></p>

```

Seeds.rb

写 seeds 创建初始 course, chapter 和 post 信息，然后再 rake db:seed。

```

create_courses = for i in 1..3 do
  Course.create!([title: "Course no.#{i}"])
end
puts "3 Courses created by admins."

create_chapters = for i in 1..3 do
  Chapter.create!([id: "#{i}", course_id: "1", title: "Chapter#{i}"])
end
puts "3 Chapters of course_id[1] created by admins."

create_posts = for i in 1..5 do
  Post.create!([chapter_id: "1", title: "Post no.#{i}", description: "这是用 seeds 生成的第#{i}个 post。"])
end
puts "5 Posts of chapter_id[1] created by admins."

```

August 20, 2016

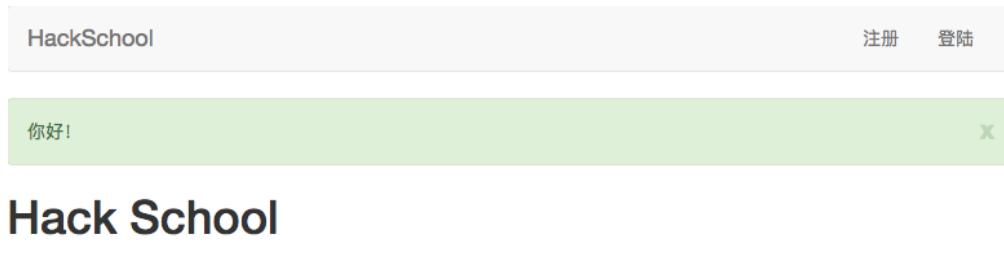
[8/16 Bug 相关 - Hack School](#)

- 1.add config/database.yml to gitignore. 项目 clone 下来第一步就做了这个。怕把 database 中的数据 push 到 github。
- 2.<div class="container-fluid"> 这个 application.html.erb 里面不添加的话，会导致 layout 紧挨侧边，异常显示。

```

<%= render "common/navbar" %>
<%= render "common/flashes" %>
<%= yield %>

```



加上就正常了。

```

<div class="container-fluid">
  <%= render "common/navbar" %>
  <%= render "common/flashes" %>
  <%= yield %>
</div>

```

你好!

X

Hack School

3. 提示异常 undefined method 'each' for nil:NilClass, 发现 index view 这里并没有异常。Google 了一下，提示我检查 controller，果然 groups_controller 里面，index 动作里面的 groups 写成了 group，因此报错。

```
def index
  @groups = Group.all
end
```

参考：[ruby on rails - undefined method `each' for nil:NilClass... why? - StackOverflow](#)

4. 我特别容易写错的地方。controller 里面的 before action 冒号点错啊，逗号少些啊，这些错误老是不停。

```
before_action :authenticate_user!, only: [:new, :create, :update, :edit, :destroy]
```

5. 用 seeds.rb 生成帐号时总是提示异常。

```
rake aborted!
```

ActiveRecord::RecordInvalid: Validation failed: Email has already been taken

跑 rails c，然后 User.all 查看到目前所有的 user：

```
[0] #<User:0x007f8c62259068> {
  :id => 1,
  :email => "admin@gmail.com",
  :encrypted_password =>
"$2a$11$RCNQ/rpVHM7uxgjNLN7tFOyYZREEfFy3GqeWs.yI/2MHZs7XbvANq",
  :reset_password_token => nil,
  :reset_password_sent_at => nil,
  :remember_created_at => nil,
  :sign_in_count => 0,
  :current_sign_in_at => nil,
  :last_sign_in_at => nil,
  :current_sign_in_ip => nil,
  :last_sign_in_ip => nil,
  :created_at => Tue, 16 Aug 2016 07:55:37 UTC +00:00,
  :updated_at => Tue, 16 Aug 2016 07:55:37 UTC +00:00,
  :is_admin => true
},
[1] #<User:0x007f8c6225b340> {
  :id => 2,
  :email => "admin\#{i}@gmail.com",
  :encrypted_password =>
"$2a$11$7Z.jC2eMNenbZp4ROrcC6.LQ3pGL8aM3cReRJpt6foHIIJCVuBOYm",
```

```

    :reset_password_token => nil,
    :reset_password_sent_at => nil,
    :remember_created_at => nil,
    :sign_in_count => 0,
    :current_sign_in_at => nil,
    :last_sign_in_at => nil,
    :current_sign_in_ip => nil,
    :last_sign_in_ip => nil,
    :created_at => Tue, 16 Aug 2016 07:55:38 UTC +00:00,
    :updated_at => Tue, 16 Aug 2016 07:55:38 UTC +00:00,
    :is_admin => true
}

```

对比之前的 seeds.rb 发现是引号的问题。#{i}起作用的是"admin#{i}@gmail.com"，双引号。而之前用成了单引号'admin#{i}@gmail.com'，rails 不能识别，email 会被读成这样:email => "admin\#{i}@gmail.com"。。

```
create_admin_accounts = for i in 1..10 do
```

```
  User.create!([email: "admin#{i}@gmail.com", password: '12345678', password_confirmation: '12345678',
  is_admin: 'true'])
end
```

```
puts "10 admin_accounts created."
```

6.用 seed.rb 为 user_id 赋值，来区分登入用户或创建用户。

```
create_groups = for i in 1..21 do
```

```
  Group.create!([title: "Group no.#{i}", description: "这是用 seed 建立的第 #{i} 个产品", user_id: "#{i}"])
end
```

```
puts "21 Groups created by admins & users."
```

August 16, 2016

重命名 rails controller 和 model

尝试写 Hack School 的问题讨论区的时候，参考了之前项目 groups 建立，后来想想叫 groups 不是很妥当，应该叫 questions。想要把 groups 重命名成 questions，这时候我已经有了一堆 model, view 和 controller。Google 了一下，这样的方法有效，但基本要手工操作：

修改 model

1.修改 table name

跑 rails g migration RenameGroupsToQuestions,修改生成的 db 文件。

```
class RenameGroupsToQuestions < ActiveRecord::Migration[5.0]
```

```
  def change
    rename_table :groups, :questions
  end
end
```

然后 rake db:migrate。

2.修改 model name group.rb -> question.rb

3.修改 question.rb 里面的代码 class Group -> class Question

4. 重命名所有的 model association has_many :groups->has_many :questions ,
belongs_to :group->belongs_to :question。

跑 rails g migration RenameGroupIdToQuestionId,修改生成的 db 文件，重命名 group_id-> question_id。

```
class RenameGroupIdToQuestionId < ActiveRecord::Migration[5.0]
  def change
    rename_column :posts, :group_id, :question_id
  end
end
```

然后 rake db:migrate。

修改 Controller

- 1.修改 controller 名 groups_controller.rb->questions_controller.rb
- 2.修改 questions_controller.rb 里面的代码 class GroupsController->class QuestionsController, CRUD 里面的 group->question。

修改 View

- 1.重命名 views 名字 groups->questions。
- 2.重命名 index/new/edit/show 里面的代码 group->question。

修改 routes

- 1.重命名 routes resources :groups->resources :questions

数据库

- 1.修改 seeds.rb group->question, 然后 rake db:reset。

参考

[How to rename rails controller and model in a project - Stack Overflow](#)

August 16, 2016

[Push seeds.rb 到 rails app \(on Heroku\)](#)

写了 seeds.rb 预设了 3 个用户，本地 OK；部署到 Heroku 居然没有效果，google 了一下，发现要把 seeds push 到 heroku 上面，需要这样：

需要先

```
heroku run rake db:migrate
```

然后

```
heroku run rake db:seed。
```

参考

[ruby - How to push seeds.rb to existed rails app \(on Heroku\)? - Stackoverflow](#)

August 16, 2016

[8/15 日记](#)

尝试过之后，就能打开一个世界：

- 写了一个写作课的 [Landing Page](#)，好开心。
- 写了 Hack School 的用户故事 Version 1，发现编故事会让你在场景里面去思考功能，比单纯去思考一个功能，思路能够更加系统全面。

August 15, 2016

[本週週記](#)

- 1.尝试转换策略，把注意力从博客转到代码。
- 2.整理了商店开发思路，背了小部分代码。

- 1) fork 项目
- 2) 套上 Bootstrap

- 3) 安装 simple_form
- 4) 设定基本的 navbar, footer 和 flashes
- 5) 安装 devise
- 6) Products & 上传图片
- 7) Admin
- 8) 购物车
- 9) 订单 & 结账页
- 10) 订单生成
- 11) 部署 JDStore 到 Heroku

3.速度巨慢，老师说这样太保守，要多赶作业。

August 14, 2016

第三週學到最棒的概念或工具

最近开始要自己尝试解题的时候越来越多，时不时会有疑问“为什么资料好像查了很多，而自己写代码的时候感觉还是不会写？”，看采铜写的《精进》的时候突然找到了答案：

从理论出发不一定能指导实践，只有在实践中通过反复积累的知识才能指导实践。“行动科学”对这个观点有比较深入的阐述。组织行为学克里斯·阿吉里斯在《行动科学》一书中写道，科学理论诞生于“维持其他变量恒定”的理想情境，而实际的问题则处于一个多种因素互相作用、相互依存又互相冲突的“复杂场域”中，并且具有某种独特性，事实上，当人在解决现实问题时，更依赖于隐性的知识和隐性的推理。行动科学的另一位大师唐纳德·舍恩认为，“三思而后行”并不一定正确，很多时候甚至可以是“行动先于思考”的，因为“人们的机智行动是高度技巧及复杂推理而形成的，而其中绝大多数有都是隐性的”，因此在行动之后反思，可能会反过来发展我们的认识。

和老师聊天的结论也是，写代码，先破破烂烂的写起来，解题。

所以，当你解题很恐慌，不知道该怎么下手时，不要害怕，保持耐心，要相信自己能解出来。你肯定能透过代码来学会代码的。

August 14, 2016

第三週遇過最大的坑

没有踩老师故意挖的坑。老师在把 JDStore 部署到 Heroku，并实现上传图片功能的教程里，埋了很多坑，人造一个“成长之不可避免”，让同学们亲自踩一踩，然后终身难忘。

我因为做得慢，还没做到的时候，就听到同学们一片哀嚎。等我做到的时候，研究了学霸的博客和老师的教材，写了 Step by Step 的步骤，然后才去部署的，中间基本没有踩什么大坑，为此我沾沾自喜了一下，直到周五晚上 Xdite 说，我就是教材作者，我就是要你们踩坑的，没踩坑说明你不看说明书。

我表示，无法反驳。好吧，同学们，看说明书不等于帮你跳过大坑，主要看说明书作者的原意。

August 14, 2016

部署 JDStore 到 Heroku

Step1: 部署到 Heroku

Step2: 获取 AWS Credentials & Bucket

- 1) 下载 Credentials

2) 创建 Bucket

Step3: 用 figaro 管理配置信息

1) 安装 figaro gem

2) 复制 application.yml 作为示例文件。

3) 修改 config/application.yml

4) 将配置信息同步到 Heroku

Step4: Heroku 可以上传图片

1) 安装 fog

2) 新增 carrierwave.rb

3) 修改 image_uploader.rb

Step5: 重新 push 到 heroku。

参考

部署 JDStore 到 Heroku

本文档实现如下 2 点：

1) 通过 Step1, 部署 JDStore 到 Heroku

2) 通过 Step2-5, 实现部署到 Heroku 的 JDStore 可以上传图片

Step1: 部署到 Heroku

跑如下指令, 将程序部署到 Heroku。没有部署过 Heroku 的可以参考 [上传 Heroku](#)。

heroku create

bundle install #修改 Gemfile 跳过 sqlite

```
git add .
```

```
git commit -m "move sqlite3 to development group & add pg gem"
```

```
git push heroku branch-name:master
```

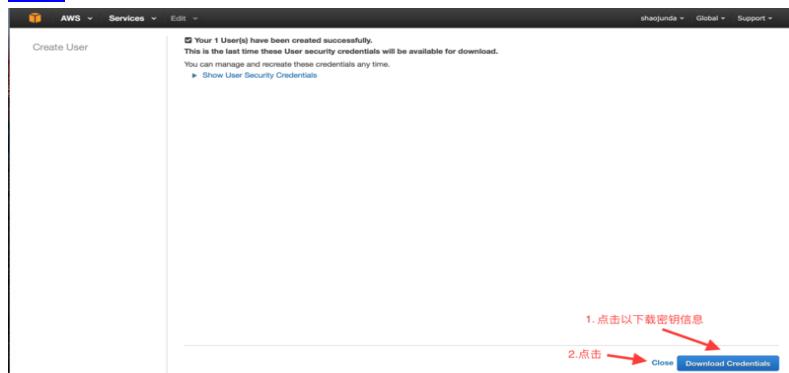
```
heroku run rake db:migrate
```

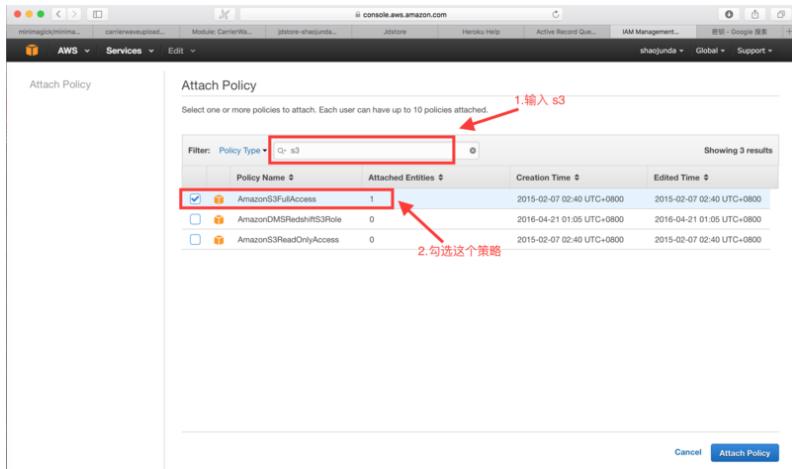
```
heroku open
```

Step2: 获取 AWS Credentials & Bucket

1) 下载 Credentials

登入 [Amazon Web Services \(AWS\)](#), 在你账号的下拉选单 选择 Security Credentials > Users > Create New Users。创建用户名为 king , 且只有操作 s3 的权限的用户, 并下载 credentials。详细步骤可参考[关于 AWS 密钥](#), 这里贴里面的 2 张示意图。





credentials.csv 文件中包含新生成的 AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY，用于 Step3 的设置。

2) 创建 Bucket

在 Amazon AWS，选择 Services > S3 > Create Bucket。创建 bucket 后，在 Properties 里找到 bucket name，在 Static Website Hosting 找到 region，用于 Step3 的设置。

Step3: 用 figaro 管理配置信息

1) 安装 figaro gem

在 Gemfile 中添加 gem 'figaro'
bundle install
figaro install

2) 复制 application.yml 作为示例文件。

cp config/application.yml config/applicatin.yml.example

3) 修改 config/application.yml

在 terminal 里用 atom config/application.yml 打开，因为这个文件被放在 gitignore 里面，禁止上传 git，默认为隐藏状态。

填写 Step2 中生成的 AWS Credentials & Bucket 信息，这里用 xxx 标示要修改的地方。

production:

```
AWS_ACCESS_KEY_ID: "xxx"
AWS_SECRET_ACCESS_KEY: "xxx"
AWS_BUCKET_NAME: "xxx"
REGION: "xxx"
```

4) 将配置信息同步到 Heroku

执行 figaro heroku:set -e production。

使用 heroku config 可以查看 heroku 中当前的所有配置信息。

Step4: Heroku 可以上传图片

1) 安装 fog

在 Gemfile 中添加 gem 'fog'，跑 bundle install。

2) 新增 carrierwave.rb

在 Terminal 里输入 touch config/initializers/carrierwave.rb，新增 carrierwave 文件，填入如下代码。

```
CarrierWave.configure do |config|
```

```
  if Rails.env.production?
```

```
    config.storage :fog
```

```
    config.fog_credentials = {
```

```
      provider:           'AWS',
```

```
      aws_access_key_id: ENV["AWS_ACCESS_KEY_ID"], # 读取 你的 key
```

```
      aws_secret_access_key: ENV["AWS_SECRET_ACCESS_KEY"], # 读取 你的 secret key
```

```
      region:             ENV["REGION"], # 读取 你 S3 bucket 的 Region 位置
```

```
}
```

```
  config.fog_directory = ENV["AWS_BUCKET_NAME"] # 读取 你设定的 bucket name
```

```
else
```

```
  config.storage :file
```

```
end
```

```
end
```

3) 修改 image_uploader.rb

打开 app/uploaders/image_uploader.rb，注释掉 storage :file

```
class ImageUploader < CarrierWave::Uploader::Base
```

```
  # storage :file
```

Step5: 重新 push 到 heroku。

```
git add .
```

```
git commit -m "add image upload"
```

```
git push heroku branch-name:master
```

```
heroku run rake db:migrate
```

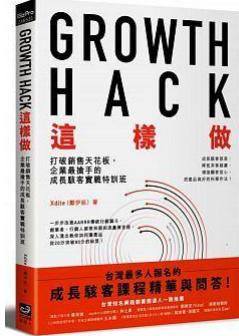
```
heroku open
```

参考

1. [上传 Heroku - mayalin](#)
2. [关于 AWS 密钥 - shaojunda](#)
3. [figaro 管理配置信息 - shaojunda](#)

August 13, 2016

[十分钟读完《Growth Hack 这样做》](#)



by Xdite(郑伊廷)

60 分前专注于 Growth

大部分的公司，其实不一定需要 Growth Hack，大家所需要的是 Growth。

硅谷常见给不同层级团队打分的标准是：

10 分：没人用的产品

20 分：Minimal Viable Product (MVP, 拥有最小可行性产品)

40 分：产品可以跟顾客收钱了

60 分：Product Market Fit (PMF, 产品达到让客户需要你的市场规模)

70 分：客户回流高

90 分：垄断市场

而硅谷对创业团队的建议是，还没 Product Marketing Fit 的团队，不需要玩 Growth Hack，只要专注于 Growth 就好！

如何快速冲到 60 分

找到快速 PMF 主题，让转化率大于流失率。什么是快速 PMF 主题？1)创业者自身很熟的主题，2)周遭需求量很大的主题。

Growth Hack 是什么？

Growth Hack 是一种融合了技术开发、产品运营和市场营销的新时代成长手段。Growth Hack 的原则是降低来客疑虑，增强顾客信心，把产品做好的科学作法！

你为什么需要 Growth Hack？

克服成长的困扰，技术、运营、销售各自为政的状况，一切成长决策皆以“数字”与“指标”为主。以“科学的方式”找出成长的“断点”，将资源投注在对用户有价值的功能开发上，利用聪明巧妙的“杠杆”开发出指数成长策略。

Growth Hack 策略

Growth Hack 三大步骤

1. 使用量测工具，抓出 Funnel 断点
2. 进行研究，推测可能成因
3. 针对假设进行 A/B Testing

Landing Page

网站营运上，Landing Page 就是指顾客点击“行销频道”（如 FB 广告、Email 等等...）上的链接后，“着陆”（Landing）的页面。

所以，Landing Page 可能会是：

- 首页
- 注册页
- 报名页
- 购买商品页
- 软件下载页

Landing Page 的目标是说服顾客，希望他们可以进行你指定的动作。这些你想要引导顾客采取的行为通常会是：

- 购买付费
- 注册账号
- 软件下载
- 填写 Email
- 订阅电子报

Landing Page 的基本结构可以分成六大块，他们分别是：

- 用一句话形容自己
- 使用这个服务(产品)的三大好处
- 叙述服务的运作原理或制作示意图、示意影片
- 加上使用者见证或媒体报道
- CALL TO ACTION(呼叫来客行动的按钮)
- 强化信心的补充：免费注册或退款保障

Landing Page 的精神是什么？其实就是做对三件事情：

- 降低来客对这个服务的疑虑
- 增强来客对使用这个服务的信心
- 最后 CALL TO ACTION，敦促来客行动

要怎么做到上面三件事情？则是在设计页面时运用 4P 原则：

- Picture (描绘愿景)
- Promise (承诺提供)
- Prove (证明有效)
- Push (敦促行动)

A/B Testing

A/B Testing 的意思就是实际做不同的测试，例如更动版面上的排版方式，是否能够让顾客更愿意执行你希望他执行的动作，转化率更为提高。

通常我们可以在网站上的下列几种地方，实作 A/B Testing：

- 文案
- 按钮位置
- 按钮颜色
- 功能区块
- 版面排版

更进一步，你应该质疑自己或许目前的做法还不是最好的做法，有没有可能改进一个什么小地方，就带来新的流量呢？A/B Testing 也是要帮你激发更好的网站流程或服务页面的手段。

Onboarding

User Onboarding 翻为"用户引导"，也就是要让新用户注册后，服务可以透过一系列的互动引导：

- 让用户了解这个产品如何使用
- 让用户体验到产品的核心价值
- 让用户养成使用这个产品的习惯

这个流程决定了 Customer 是否会回头再使用这个产品，是否继而爱上这个产品，养成习惯并成为回头客。

Onboarding 引导用户的三法则八步骤

要做到 Onboarding 的三大法则，有一些具体的网站经营步骤，让我们一起来看看。

- 消除疑虑、降低挫折
 - Signup：让注册步骤变得尽可能简单，若是付费服务就提供一定期间试用。
 - Welcome：马上感谢顾客注册，并让顾客马上感受到他们是为了什么而来。
- 马上为用户带来好处
 - Engage immediately：马上出现引导，提供有价值的服务，鼓励他们做出关键行为。
 - Ask for feedback：在第一周就建立沟通管道，听取服务回馈。
 - Provide feedback：给予如何更加强化服务价值的建议。
- 根据行为给予小惠
 - Ask for referrals 提供一些优惠，鼓励顾客邀请朋友前来消费。
 - Being Customizing the experience using data analytics：使用资料分析，给予顾客更量身打造的产品使用体验。
 - Transition to a nurturing program：持续提供资讯让会员能够更强化他们来消费的体验。

Customer Support

为什么 Customer Support 跟 Growth 很有关系？好的客服支援，可以让你发现很多没有自己从来想到的"客户流失"原因。

客服不只是支援，而是降低疑虑、增强信心手段。透过客服反馈的过程，发现产品成长关键：

- 有真实需要的题目(消费者愿意付钱)
- 产品具体改进的方向
- 之前完全没想过的潜在市场与顾客

Email List

用 Email List 做两件事：

- 留在可能会流失的潜在客户
- 养成未来潜在有兴趣买单的用户

Referral 与 NPS

Refferal，用户推介，他的原理建立在：

- 朋友用过
- 你相信朋友
- 推荐又有 Discount
- 你相信别人测过的结果(特别是很贵的产品) 所以建立信任与市场教育成本相对较低。

Net Promoter Score(NPS)，净推荐值，最流行的顾客忠诚度分析指标，一个客户愿意向其他人推荐你产品的数值。

如何衡量 NPS 的公式

NPS 问卷中，分数的意义怎么评量？作者提供下面的方法。

- 9-10 分是会推荐的人

- 7-8 是中立者，对这个产品满意，但不会主动推荐，除非有人问起。(因为小部分不少很满意)
- 1-6 分是反推荐者，因为产品没有给自己带来实际的好处，或者体验很差

计算方式是：

(推荐者人数 - 反推荐者人数) / 总回收问卷人数的"百分比"。

比如 100 个人缴交问卷，60 人推荐，10 人不推荐，30 人中立，那么 NPS 就是 50%，也就是 50 分。

计算出 NPS 总分的意义

一般说来，有达到 Product Market Fit 的服务，NPS 会在 20 分以上。

- 30 分是 Good
- 50 分是 Great
- 70 分是 Excellent 一般来说 30 分以上的产品就适合进行 Referral Program 了！50 分就是口碑极好。70 分以上只有一家常胜军做到，叫 Apple，已经不大科学了。NPS 被称之为"The Ultimate Question"，你只要用这一题就可以找出"你的服务 什么口碑散不出去"的真正原因。客户回馈意见很多也很杂，也许你对如何改进没有头绪。但对于成长这个主题来说，其实你只要关注 NPS 这一题在"7-8 分"的顾客的意见，看这一块就足够。因为他们是"你口碑无法散出去的原因"，一旦改进，就能够真正成长。

[点击跳转原书](#)

August 10, 2016

Destroy_all & Delete_all

Destroy_all 类似于彻底清除，一点儿痕迹都不留。delete_all 类似于，扔垃圾桶里了，(之前有的) 使用痕迹还在。

destroy_all(conditions = nil)

实例化记录，调用 destroy 方法。执行每个对象的回调。

delete_all(conditions = nil)

delete_all 是一条 SQL DELETE 声明，直接用于数据库，比 destroy_all 更加高效。不会实例化记录，不会调用 destroy 方法，不会引用回调。

举例

比如要删除一个 user，且他的 user ID 用于许多表格。删除这个用户，以及删除所有表格中这个 ID 的所有记录。

```
u = User.find_by_name('JohnBoy')
```

```
u.usage_indexes.destroy_all
```

```
u.user_stats.destroy_all
```

```
u.delete
```

但如果把 destroy_all 换成 delete_all，只会删掉用户自己表单中的 ID，其他表单中这个用户的记录不会被删掉，只是在 ID 栏位变为空。

参考

[ruby on rails - delete_all vs destroy_all? - Stack Overflow](#)

[destroy_all \(ActiveRecord::Relation\) - API dock](#)

[delete_all \(ActiveRecord::Relation\) - API dock](#)

August 9, 2016

8/8 日记

ORID 是一个思考框架，网络上面也有很多相关的资讯。利用 ORID 整理今天的思绪。

Objective

關於今天的課程，你記得什麼？

完成了什麼?=>商品加入购物车，清空购物车，删除不要的商品。

Reflective

你要如何形容今天的情緒

今天的高峰是什麼?=>在 google 和学霸的帮助下，用一行代码实现了，购物车里商品唯一。

validates_uniqueness_of :product_id, :scope => :cart_id

今天的低點是什麼?=>没有什么低点。

Interpretive

我們今天學到了什麼？

今天一個重要的領悟是什麼?=>自己先尝试写开发步骤，厘清思路，然后卡在某个步骤，再去求助，问题会更容易解决。但是要注意控制时间，30-60 分内完成一个小功能，尽量不要一个功能 2-3 小时，这样你基本已经踩坑。

Decisional

我們會如何用一句話形容今天的工作

有哪些工作需要明天繼續努力?=>尝试自己分拆开发步骤。

August 9, 2016

第二周 周记

探索基本结束

1.[捷径就是避开大部分的错，犯少量的错，迅速走到终点。](#)

2.要不要背代码？当然要啊。问你自己一个问题，现在你去面试，然后你要找一个后端开发的工作，你要怎么办？难道你还有机会回来翻教材？

3.怎么快速背会代码？按自己思路整理教材，拆成你容易记住的模块。多抄几遍，默写。查代码含义，帮助记忆。分模块写网站，记录出错的地方。重复几遍。然后就记住了。

输出

博客写了 20+ 篇。之前我想的是培训结束后，我要是写 100 篇博客，我应该就能找到工作了吧？！呵呵，当时非常的天真。不过不停地写确实可以提高表达能力，最近明显感觉出 8 月写的文章比 6 月写的 能够表达的更清楚了。

我为什么不停地写呢？！因为一个时间点我并不能找到人和我聊某个话题，我只好写博客，假装我在和人聊天。还有我比较懒，一样的话，我不想和一个人讲一次，再和另一个人再讲一次。

推荐阅读

[用 Migrations 指令创建修改表格 & schema](#)

[Rake routes 显示你定义的 routes](#)

August 7, 2016

[Rake routes 显示你定义的 routes](#)

我一直以为 rake routes 是天上掉馅饼地出来，是系统给你自动生成的。今天我突然发现，不是的，rake routes 出来的都是你一步一步自己定义出来的。你 rake routes 一下，只是把你之前定义好的东西显示出来。以 welcome 页为例来说明：

Step1: 定义 Controller Action

定义 welcome 的 index 动作。

```
class WelcomeController < ApplicationController
  def index
    flash[:notice] = "早安！你好！"
  end
end
```

注意：实际还需要创建对应的 index 页面，不然后面用浏览器打开主页会提示异常。

Step2: 修改 routes.rb

把主页指向 welcome#index。

```
Rails.application.routes.draw do
  root 'welcome#index'
end
```

Step3: rake routes 显示你定义的 routes

Prefix	Verb	URI Pattern	Controller#Action
root	GET	/	welcome#index

August 7, 2016

捷径（一）：少犯错或错误可控

捷径是我一直想写的主题，但是一直写不出来。最近我好像知道了一些。

捷径就是少犯错或错误可控。

错误是无尽的，但是你的时间是有限的。并不是犯 1 个错，长 1 个知识点，就可以叫做进步。而是你要先想办法避免犯大部分的错，之后再犯小部分的错，迅速走到终点。

比如，学习的时候你会有这样 2 种做法：

做法 A：

乱试 > 出错 > 回去翻教程或问别人或自己 google，解决问题 > 记录错误，下次不再犯。

嗯 看起来好像也没什么问题。你靠实践犯了错，印象深刻，且之后有记录，不大可能重犯，看着也完全符合“每天进步一点点”。但实际的情况是，你犯错的主要原因是没有先看教程，你记错了指令，犯了原本可以避免的错误。

做法 B：

按照教程练习 > 出错 > 仔细比对教程或问别人或自己 google，解决问题 > 记录错误，下次不再犯。

照着教程这一步本身就能让你跳过 80% 的坑，接着你出错更大可能会是输入错误，漏写等等，基本不会有大的错误。看着一点儿也不惊险不刺激，也不忙的样子，甚至还有点儿小无聊。你旁边同学在死去活来 debug 的时候，你在比对字符，还有比这更无聊的事情吗？嗯 看着的确是这样的。

其实，做法 A 在第 1 步就输了，因为把自己置身在 100% 的错误里，做法 B 在第 1 步就避开了 80% 的错。做法 A 面对的可能是未知的尚待探索的错误。做法 B 面对的是错误已知，或基本已知。从起点走到终点，是花样百出地犯 100% 的错走得快，还是花样百出地犯 20% 的错走得快？

因此，假如你刚开始自己开发，你应该这样：

写 step by step 的开发步骤 > 出错 > 检查开发步骤，解决问题 > 记录错误，下次不再犯。

千万不要乱开发，就是感觉这样可以。或者我边看以前的，边修改成我现在想要的。然后把自己推到未知的坑里，天天忙着 debug。

写 step by step 的开发步骤跳过 80% 的坑，然后再去 debug，解决一小部分问题，迅速走到终点。

August 7, 2016

电商网站开发中 遇到的 bug

bug 顺手记录一下，希望犯过的错误不要再犯。

1.flash 后面多输入空格，异常提示。删掉就可以了。

```
1 class WelcomeController < ApplicationController
2   def index
3     flash[:notice] = "早安！你好！"
4   end
5 end
6
```

SyntaxError in WelcomeController#index

/Users/yalinma/jdstore/app/controllers/welcome_controller.rb:3: syntax error, unexpected '=', expecting keyword_end flash
[:notice] = "早安！你好！" ^

Extracted source (around line #3):

```
1 class WelcomeController < ApplicationController
2   def index
3     flash[:notice] = "早安！你好！"
4   end
5 end
```

Rails.root: /Users/yalinma/jdstore



2.括号位置错误，导致异常。

SyntaxError in WelcomeController#index

/Users/yalinma/jdstore/app/views/common/_navbar.html.erb:13: syntax error, unexpected ',', expecting ')' ...fer.append=(
link_to("登入"), new_user_session_path);@out... ... ^

Extracted source (around line #13):

```
11   <% if !current_user %>
12   <li><%= link_to("注册", new_user_registration_path) %></li>
13   <li><%= link_to("登入"), new_user_session_path %></li>
14   <% else %>
15   <li class="dropdown">
16     <a href="#" class="dropdown-toggle" data-toggle="dropdown">
```

3.current_user 拼错，导致异常。

SyntaxError in WelcomeController#index

/Users/yalinma/jdstore/app/views/common/_navbar.html.erb:13: syntax error, unexpected ',', expecting ')' ...fer.append=(
link_to("登入"), new_user_session_path);@out... ... ^

Extracted source (around line #13):

```
11   <% if !current_user %>
12   <li><%= link_to("注册", new_user_registration_path) %></li>
13   <li><%= link_to("登入"), new_user_session_path %></li>
14   <% else %>
15   <li class="dropdown">
16     <a href="#" class="dropdown-toggle" data-toggle="dropdown">
```

4.多写了 div 标签导致布局异常。

August 5, 2016

第二周遇过最大的坑

“回答别人的问题是加速成长的秘诀之一” 听了觉得很有道理，但之前一直没有做。觉得“别人的问题和我在做的事情中间隔着巨大的鸿沟” “我也不确定我是否能回答”，一直没有尝试过回答别人的问题，直到为了挣分，才跑到 issue 上慌慌张张回答别人的问题。

实际的体会是别人的问题，你也不大可能张口就能讲出来，但你可以多查几份资料，然后可以懂一点儿，然后用自己的话描述一下，或者把你自己的讲解的比较简明易懂的资料贴过去，这样都可以算回答别人问题。别人的问题反过来可以让你思考一下你在练习的教材，哪些是你可以拿教材来做示例的，然后试着解释概念的时候举个例子。

嗯 这就是别人的问题可以给你灵感，然后你会组织，链接你的知识点，积累你谈话的资料。非常微小的感觉。

August 5, 2016

[用 Migrations 指令创建修改表格 & schema](#)

创建表格

rails g migration Create<TableName>Table [columnName:type] [columnName:type]

举例，创建一个名为 Products 的表格，包含 title 和 price 这 2 列，类型分别是 string 和 float。

跑指令 rails g migration CreateProducts title:string price:float，生成的数据库文件。

db/migrate/20160805041849_create_products.rb

```
class CreateProducts < ActiveRecord::Migration[5.0]
  def change
    create_table :products do |t|
      t.string :title
      t.float :price
    end
  end
end
```

跑 rake db:migrate，更新 schema。

db/schema.rb

```
ActiveRecord::Schema.define(version: 20160805041849) do
```

```
  create_table "products", force: :cascade do |t|
    t.string "title"
    t.float "price"
  end
```

end

添加行

rails g migration Add<Anything>To<TableName> [columnName:type]

举例，在 Products 的表格中，添加一列 quantity，类型为 integer。

跑指令 rails g migration AddQuantityToProducts quantity:integer, 生成的数据库文件。

db/migrate/20160805042454_add_quantity_to_products.rb

```
class AddQuantityToProducts < ActiveRecord::Migration[5.0]
```

```
  def change
```

```
    add_column :products, :quantity, :integer
```

```
  end
```

```
end
```

跑 rake db:migrate, 更新 schema。可以看到 quantity 这一列已经添加进去了。

db/schema.rb

```
ActiveRecord::Schema.define(version: 20160805042454) do
```

```
  create_table "products", force: :cascade do |t|
```

```
    t.string "title"
```

```
    t.float "price"
```

```
    t.integer "quantity"
```

```
  end
```

```
end
```

移除行

rails g migration Remove<Anything>From<TableName> [columnName:type]

举例，移除刚刚添加的那一行 quantity。

跑指令 rails g migration RemoveQuantityFromProducts quantity:integer, 生成的数据库文件。

db/migrate/20160805042834_remove_quantity_from_products.rb

```
class RemoveQuantityFromProducts < ActiveRecord::Migration[5.0]
```

```
  def change
```

```
    remove_column :products, :quantity, :integer
```

```
  end
```

```
end
```

跑 rake db:migrate, 更新 schema。可以看到 quantity 那一列已经移除。

db/schema.rb

```
ActiveRecord::Schema.define(version: 20160805042834) do
```

```
  create_table "products", force: :cascade do |t|
```

```
    t.string "title"
```

```
    t.float "price"
```

```
  end
```

```
end
```

添加空的 migration

做更加灵活的修改。

rails g migration <name>

举例，为 price 添加默认值 1.0。

跑 rails g migration AddPriceDefaultValue，在生成的数据库文件里，添加 change_column :products, :price, :float, default:1.0。

```
db/migrate/20160805043112_add_price_default_value.rb
class AddPriceDefaultValue < ActiveRecord::Migration[5.0]
```

```
  def change
    change_column :products, :price, :float, default:1.0
  end
end
```

跑 rake db:migrate，更新 schema。可以看出 price 上已经有了默认值 1.0。

```
db/schema.rb
```

```
ActiveRecord::Schema.define(version: 20160805043112) do
```

```
  create_table "products", force: :cascade do |t|
    t.string "title"
    t.float  "price", default: 1.0
  end
```

```
end
```

August 5, 2016

第二週學到最棒的概念或工具

捷径是不要犯同样的错。最近重复练习的时候，有刻意去写自己遇到的 bug 或疑问。打错一次，写下来，下次的确不会重复打错，每次再遇到，都会下意识注意一下，再检查一下，有这种很刻意注意的感觉。

坑踩多了自然记住了，这是一种很天真的想法。乱写 无规律地踩坑，犯同样的错，都是浪费时间的行为。

August 4, 2016

HTML & CSS 前端资源参考

分享几个特别简明易懂的前端资源，一般的 HTML CSS 问题可以先翻这几个网站。

1.Bootstrap - 全局 CSS 样式

设置全局 CSS 样式；基本的 HTML 元素均可以通过 class 设置样式并得到增强效果；还有先进的栅格系统。

全局 CSS 样式

设置全局 CSS 样式；基本的 HTML 元素均可以通过 class 设置样式并得到增强效果；还有先进的栅格系统。

概览

深入了解 Bootstrap 底层结构的关键部分，包括我们让 web 开发变得更好、更快、更强壮的最佳实践。

HTML5 文档类型

Bootstrap 使用到的某些 HTML 元素和 CSS 属性需要将页面设置为 HTML5 文档类型。在你项目中的每个页面都要参照下面的格式进行设置。

```
<!DOCTYPE html>
<html lang="zh-CN">
...
</html>
```

[复制](#)

移动设备优先

概览
栅格系统
排版
代码
表格
表单
按钮
图片
辅助类
响应式工具
使用 Less
使用 Sass
[返回顶部](#)
[主题预览](#)

2. 编码规范 by @mdo

编写灵活、稳定、高质量的 HTML 和 CSS 代码的规范。

编码规范 by @mdo

编写灵活、稳定、高质量的 HTML 和 CSS 代码的规范。



目录

HTML

- 语法
- HTML5 doctype
- 语言属性（Language attribute）
- 字符编码
- IE 兼容模式
- 引入 CSS 和 JavaScript 文件

CSS

- 语法
- 声明顺序
- 媒体查询（Media query）的位置
- 带前缀的属性
- 单行规则声明
- 简写形式的属性声明

3. W3Schools Online Web Tutorials

W3Schools 提供网页开发教程，涵盖 HTML, CSS, JavaScript, PHP, SQL, Bootstrap, 和 jQuery。

HTML and CSS

Learn HTML
Learn CSS
Learn W3.CSS
Learn Colors
Learn Bootstrap
Learn How To

JavaScript

Learn JavaScript
Learn jQuery
Learn jQueryMobile
Learn AppML
Learn AngularJS
Learn JSON

HTML Graphics

Learn Canvas
Learn SVG
Learn Icons
Learn Google Maps

Server Side

Learn SQL
Learn PHP
Learn ASP

HTML

The language for building web pages

[LEARN HTML](#) [HTML REFERENCE](#)

HTML Example:

```
<!DOCTYPE html>
<html>
<title>HTML Tutorial</title>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

[Try it Yourself >](#)

CSS

The language for styling web pages

CSS Example:

```
body {
    background-color: lightblue;
}
h1 {
    color: white;
```

August 4, 2016

8/4 日记

Objective

關於今天的課程，你記得什麼? => 尝试自己写个网上商店。

完成了什麼? => 上传图片及排版。

Reflective

你要如何形容今天的情緒 => 又开始小迷惘了。

今天的高峰是什麼? => 噢 今天我们组挣了 7.6 万分。

今天的低點是什麼? => 小累，想睡一会儿，又睡不着。只好提前起来写日记。

Interpretive

我們今天學到了什麼? =>

今天一個重要的領悟是什麼? => 黑客松比赛其实是一种社交。你得想办法让评委/主办方颁奖给你。

Decisional

我們會如何用一句話形容今天的工作 => 辣是一个憋出 CSS 布局的过程。

有哪些工作需要明天繼續努力? => 整理一下思路。

August 4, 2016

8/3 日记

Objective

關於今天的課程，你記得什麼? => 尝试自己写个网上商店。

完成了什麼? => 商店的登入登出系统，商品的前端和后台 基础页面显示。

Reflective

你要如何形容今天的情緒 => 平常心。

今天的高峰是什麼? => 老师说回答 issue 可以挣分，写出收藏夹功能可以挣分😊。

今天的低點是什麼? => 起得有点儿晚。

Interpretive

我們今天學到了什麼? =>

今天一個重要的領悟是什麼? => 帮助别人是最快的成长。最近一直在自己练习，为了挣分回答了 issue。

嗯，感觉和我在做的事情中间隔着巨大的鸿沟，我在不停的敲代码的时候，对面同学想知道理论基础。之前因为查了太多理论，或者写篇博客就能花一天，导致我代码练习偏少，我一直想刻意纠正，尽量少查理论基础，多敲代码练习。今天在 Xdite 的邪恶诱导下，我写了 6 个主题的 comment 回 issue，尝试回答别人的问题。

发现“帮助别人”，之前并不在我思考范围之内：我还没有练到很熟练啊！这个和我现在练习的隔得好远，我上去回答不会显得自作多情啊！

听了道理但不照着做，好像和我没关系一样。“帮助别人是最快的成长。”应该要认真对待，作为一个问题去尝试思考一下。

Decisional

我們會如何用一句話形容今天的工作 => 先打草稿 知道自己要改哪些東西，然后基本就不会踩坑。

有哪些工作需要明天繼續努力? => 明天要怎么挣分?

August 3, 2016

8/2 日记

Objective

關於今天的課程，你記得什麼？=> ! & ?的區別。

完成了什

Reflective

你要如何形容今天的情緒 => 平常。

今天的高峰是什麼? => 下午晚上, 感冒的感觉过去了, 心情好了许多。

今天的低點

Interpretive

今天一個重要的領悟是什麼? => 捷徑是不要犯同样的错。最近重复练习的时候，有刻意去写自己遇到的 bug 或疑问。打错一次，写下来，下次的确不会重复打错，每次再遇到，都会下意识注意一下，再检查一下，有这种行为才算真正的学习。

这种很刻意

我們會仔細思考這個方案的可行性，並考慮到各方面的影響。

我們會如何用一句話形容今天的工作 =>很平吊， 重複性高， 有那些工作重要明天繼續努力。 明天要讲新的内容。

有哪些工作需要

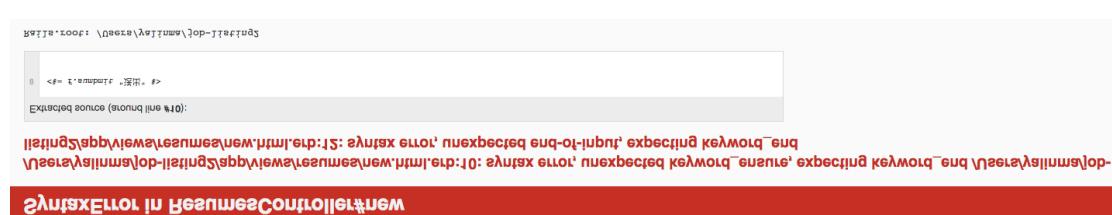
August 2, 2016

10-0

希望狠过的错误不要再犯，希望经得住下一道练习时能自动有答案。

(解答) 切磚網上管上部八

1. [OK] 退寫了<% and %> submit 按鍵 提交



(解答) 招聘網站第八部分

1. [OK] params[:job_id]写错，提示异常。

```

app/controllers/admin/resumes_controller.rb
class Admin::ResumesController < ApplicationController
  before_filter :authenticate_user!
  before_filter :require_is_admin

  layout 'admin'

  def index
    @job = Job.find(params[:job_id])
    @resumes = @job.resumes.order('created_at DESC')
  end
end

```

ActiveRecord::RecordNotFound in Admin::ResumesController#index

Couldn't find Job with 'id'=

Extracted source (around line #8):

```

6
7   def index
8     @job = Job.find(params[:id])
9     @resumes = @job.resumes.order('created_at DESC')
10    end
11  end

```

2. [OK] bug 一样，同一文件上传多次，admin 在后台下载时，会有乱七八糟的异常提示。实际上再换个文件上传就能正常下载了。

August 2, 2016

招聘網站 6-8 部分 练习第 3 遍

bug 疑问顺手记录一下，希望犯过的错误不要再犯，希望疑问在下一遍练习时能自动有答案。

(解答) 招聘網站第七部分

[OK] simple_format 写错导致异常提示。

NoMethodError in Admin::Jobs#show

Showing /Users/yalinma/job-listing2/app/views/admin/jobs/show.html.erb where line #4 raised:

```
undefined method `simple_form' for #<#<Class:0x007fce97b40b78>:0x007fce9859b5f0>
Did you mean?  simple_format
```

Extracted source (around line #4):

```

2
3   <p>
4     <%= simple_form(@job.description) %>
5   </p>

```

Rails.root: /Users/yalinma/job-listing2

(解答) 招聘網站第八部分

[OK] 注意这里的 resources :resumes 是在 admin 里面的 jobs 下面，写错位置会导致后面的履历列表不能正常显示。

config/routes.rb

```

Rails.application.routes.draw do
  devise_for :users

```

```

resources :jobs do
  resources :resumes
end
namespace :admin do
  resources :jobs do
    member do
      post :publish
      post :hide
    end
  end

  resources :resumes
end
root 'jobs#index'
end

```

The screenshot shows a 'Job Listing' page with a blue header bar containing the text 'Jobs'. Below the header, the title 'Job no.10 - 履历列表' is displayed. The main content area contains two resume entries. Each entry has a light gray background and includes the following information:

- Email: example@gmail.com
- Name: ccccc or sss
- A small blue link labeled '% Download Resume'.

August 2, 2016

Publish/Hide 功能实现

想要实现工作职位的公开/隐藏状态，要定义 MVC 和 routes。

Controller Action

在 controller 里面定义 Publish/Hide 方法。

app/controllers/admin/jobs_controller.rb

```

def publish
  @job = Job.find(params[:id])
  @job.publish!
  redirect_to :back
end

def hide
  @job = Job.find(params[:id])
  @job.hide!

```

```
    redirect_to :back
  end
```

routes

定义浏览器对 publish/hide 的响应方式是 post。

routes.rb

```
namespace :admin do
  resources :jobs do
    member do
      post :publish
      post :hide
    end
  end
end
```

跑 rake routes 结果：

Prefix	Verb	URI Pattern	Controller#Action
	publish_admin_job	POST /admin/jobs/:id/publish(.:format)	admin/jobs#publish
	hide_admin_job	POST /admin/jobs/:id/hide(.:format)	admin/jobs#hide

View

对 publish_admin_job_path 和 hide_admin_job_path 的使用。

app/views/admin/jobs/index.html.erb

```
<% if job.is_hidden %>
```

```
  <%= link_to("Publish", publish_admin_job_path(job), :method => :post, :class => "btn btn-xs btn-default") %>
  <% else %>
    <%= link_to("Hide", hide_admin_job_path(job), :method => :post, :class => "btn btn-xs btn-default") %>
  <% end %>
```

Model

更进一步，可以使用 model 定义的方法，来进一步简化 action。

app/models/job.rb

```
def publish!
```

```
  self.is_hidden = false
  self.save
end
```

```
def hide!
```

```
  self.is_hidden = true
  self.save
end
```

参考

1.[HTTP 方法：GET 对比 POST](#)

August 2, 2016

4 个最常用的 DB migration 命令

1.rails g migration <name> 创建空的 migration。

2.rake db:migrate 数据库迁移。类似于电脑中的 Save，将修改的栏位保存进数据库。

3.rake db:rollback 回退。类似于电脑中的 Ctrl + Z，退回上一步。

4.rake db:reset 重启。一套命令集合，运行这条指令等价于运行如下 4 条指令：

rake db:drop(数据库删除)

rake db:create(数据库建立)

rake db:schema:load(数据库栏位建立)

rake db:seed(建立 seeds 数据)

使用场景

刚 migrate 完发现错了，跑如下指令

rake db:rollback 退回上一步

修改完

rake db: migrate 保存进数据库。

重命名栏位，跑如下指令

rails g migration fix_xxxx_xxxx 创建空的 migration。

rename_column :xxxx, :xxxx 在上一条指令生成的 db 文件中重命名栏位。

rake db: migrate 保存进数据库。

添加预设值，跑如下指令

rails g migration fix_xxxx_xxxx 创建空的 migration。

change_column_defalut 在上一条指令生成的 db 文件中修改预设值。

rake db: migrate 保存进数据库。

数据库改乱了，回到初始数据。重启数据库，跑如下指令

rake db:reset 重启。执行一套命令集合，包含 4 条指令：rake db:drop(数据库删除) => rake db:create(数据库建立) => rake db:schema:load(数据库栏位建立) => rake db:seed(建立 seeds 数据)

seeds 数据是你设定的 数据库初始数据。

参考

1.[撰寫 seed.rb 檔\(自動產生資料庫數據\)及 rake db:reset](#)

2.[Migrations - Rails Guides](#)

August 2, 2016

[8/1 日記](#)

Objective

關於今天的課程，你記得什麼? => 简历筛选，投递；简历后台管理

完成了什麼? => 招聘网站 6-8 部分，重复练习 1.5 遍。博客写了 6 篇，加上这一篇就 7 篇了。

Reflective

你要如何形容今天的情緒 => 基本恢复平常心了。

今天的高峰是什麼? => Xdite 签名的《Growth Hack》。笑来老师深夜送的猪蹄。

今天的低點是什麼? => 朋友圈点赞数太少。靠！老子不发了。

Interpretive

我們今天學到了什麼? =>

今天一個重要的領悟是什麼? => 进阶。Xdite 分享了一本对她影响最大的书[《師父：那些我在課堂外學會的本事》](#)：有些人失败一生就是因为总是在犯同样的错。

跳出困局的方法是 尝试别人的方法。更精确的说法是，尝试比你优秀的人告诉你的高阶方法。有时候可能受困于自己的经验，你不理解，但你可以乖乖照做，或者尽量诚恳的去尝试，做过之后你就反过来明白为什么大神让你那么做，你可能就跳出困局了。

总觉得自己的最好，自己的方法最对，自己理解不了的就是不对的，很可能会把自己导向困局。重复不停犯同样的错。

Decisional

我們會如何用一句話形容今天的工作 => 开心，明天还想继续练。

有哪些工作需要明天繼續努力? => 招聘网站 继续重复练。

August 1, 2016

招聘網站 6-8 部分 练习第 2 遍

Branch: new2

bug 疑问顺手记录一下，希望犯过的错误不要再犯，希望疑问在下一遍练习时能自动有答案。

(解答) 招聘網站第六部分

1. [OK] <%= job.wage_lower_bound %> 多写了一个)，提示异常。

SyntaxError in JobsController#index

```
/Users/yalinma/job-listing2/app/views/jobs/index.html.erb:47: syntax error, unexpected ')', expecting keyword_end ...ppend=(  
job.wage_lower_bound);@output_buffer.safe_append=' ... ^
```

Extracted source (around line #47):

```
45     </td>  
46     <td>  
47     <%= job.wage_lower_bound %>  
48     </td>  
49     <td>  
50     </td>
```

Rails.root: /Users/yalinma/job-listing2

2. [OK] 可以用 schema.rb 查看当前生成的表格

schema.rb

```
ActiveRecord::Schema.define(version: 20160730050750) do
```

```
create_table "jobs", force: :cascade do |t|  
  t.string    "title"  
  t.text      "description"  
  t.datetime "created_at",           null: false  
  t.datetime "updated_at",           null: false  
  t.integer   "wage_upper_bound"  
  t.integer   "wage_lower_bound"  
  t.string    "contact_email"  
  t.boolean   "is_hidden",          default: true  
end
```

```
create_table "users", force: :cascade do |t|  
  t.string    "email",             default: "",    null: false  
  t.string    "encrypted_password", default: "",    null: false  
  t.string    "reset_password_token"  
  t.datetime "reset_password_sent_at"
```

```

t.datetime "remember_created_at"
t.integer "sign_in_count", default: 0, null: false
t.datetime "current_sign_in_at"
t.datetime "last_sign_in_at"
t.string "current_sign_in_ip"
t.string "last_sign_in_ip"
t.datetime "created_at", null: false
t.datetime "updated_at", null: false
t.boolean "is_admin", default: false
t.index ["email"], name: "index_users_on_email", unique: true
t.index ["reset_password_token"], name: "index_users_on_reset_password_token", unique: true
end

```

end

3. [OK] created_at 写错，导致异常。主页打不开，还居然提示没有 is_hidden 这个栏位。

ActiveRecord::StatementInvalid in Jobs#index

Showing /Users/yalinma/job-listing2/app/views/jobs/index.html.erb where line #39 raised:

SQLite3::SQLException: no such column: create_at: SELECT "jobs".* FROM "jobs" WHERE "jobs"."is_hidden" = ? ORDER BY create_at DESC

Extracted source (around line #39):

```

37
38   <tbody>
39     <% @jobs.each do |job| %>
40       <tr>
41         <td>
42

```

Rails.root: /Users/yalinma/job-listing2

app/models/job.rb

scope :recent, ->{order("created_at DESC")}

(解答) 招聘網站第七部分

1. <% end %> 写 错 。 异 常 提 示 。

ActiveRecord::StatementInvalid in Jobs#index

Showing /Users/yalinma/job-listing2/app/views/jobs/index.html.erb where line #39 raised:

SQLite3::SQLException: no such column: create_at: SELECT "jobs".* FROM "jobs" WHERE "jobs"."is_hidden" = ? ORDER BY create_at DESC

Extracted source (around line #39):

```

37
38   <tbody>
39     <% @jobs.each do |job| %>
40       <tr>
41         <td>
42

```

Rails.root: /Users/yalinma/job-listing2

August 1, 2016

redirect_to 3 种用法

1.url 指的是“/”，默认的。

redirect_to posts_path

2.url 指的是确切的 url “<http://localhost:3000>”

redirect_to posts_url

3.redirect_to 某个路径

redirect_to "http://www.google.com"

August 1, 2016

scope 是什麼？怎麼用

scope 簡介

scope 的作用是将经常使用或重复的 ORM 语法模块化，下次使用时直接用 scope 就可以了。比如：

```
class Job < ApplicationRecord
```

```
  scope :recent, -> { order("created_at DESC") }
```

```
end
```

上面的这段代码我们定义了 recent 的 scope，只要我们使用 recent 指令，就等于使用("created_at DESC")一样。这样就可以让代码更加简洁。

使用场景

当有过于复杂的资料查询

当有重复使用的资料查询

使用方式

不带参数的方式

```
class Job < ApplicationRecord
```

```
  scope :published, -> {where(is_hidden: false)}
```

```
end
```

带参数的方式

```
class Job < ApplicationRecord
```

```
  scope :created_before, ->(time) { where("created_at < ?", time) }
```

```
end
```

可以连在一起，顺序没有影响

```
class Job < ApplicationRecord
```

```
  scope :published, -> {where(is_hidden: false)}
```

```
  scope :created_before, ->(time) { where("created_at < ?", time) }
```

```
end
```

```
Job.published.created_before(Time.now)
```

参考

1. [ActiveRecord Query Interface - 資料表操作](#)

2. [scope - Rails 102](#)

August 1, 2016

controller 内 render 的用法

创建响应

从控制器的角度来看，创建 HTTP 响应有三种方法：

调用 render 方法，向浏览器发送一个完整的响应；

调用 redirect_to 方法，向浏览器发送一个 HTTP 重定向状态码；

调用 head 方法，向浏览器发送只含报头的响应；

渲染文本

调用 render 方法时指定 :plain 选项，可以把没有标记语言的纯文本发给浏览器：

```
render plain: "OK"
```

渲染普通文本，不带格式。不推荐使用，可用 :plain 代替

```
render :text
```

渲染纯文本主要用于 Ajax 或无需使用 HTML 的网络服务。

默认情况下，使用 :plain 选项渲染纯文本，不会套用程序的布局。如果想使用布局，可以指定 layout: true 选项。

渲染动作的视图

```
render :new
```

如果想渲染同个控制器中的其他模板，可以把视图的名字传递给 render 方法：

```
def create
```

```
  @job = Job.new(job_params)
```

```
  if @job.save
```

```
    redirect_to jobs_path
```

```
  else
```

```
    render :new
```

```
  end
```

```
end
```

执行以上代码，会控制器 view 的 new.html.erb 模板。

render 方法的选项

render 方法一般可接受四个选项：

```
:content_type
```

```
:layout
```

```
:location
```

```
:status
```

:layout 选项

render 方法的大多数选项渲染得到的结果都会作为当前布局的一部分显示。

:layout 选项告知 Rails，在当前动作中使用指定的文件作为布局：

```
render layout: "special_layout"
```

区别：如果在 controller 上部写 layout: "special_layout"，则所有的 action 都套用这个布局；而某个 action 里面写 render layout: "special_layout"，则只有当前 action 套用这个布局。

也可以告知 Rails 不使用布局：

```
render layout: false
```

参考

1. [render 参数汇总与详解](#)

2. [Rails 布局和视图渲染](#)

[在 view 里面的 render 用法](#)

```
render :partial
```

partial 的意思是别人的 拿来用。

局部样式 Partials

局部样式可以将 Template 中重复的程序抽出来。例如我们经常用来设定页面导航标题的 navbar.html.erb。Partial Template 的命名规则是下划线开头，调用时不需要加下划线，例如：

```
<%= render :partial => "common/navbar" %>
```

在这种情况下，可以省略:partial：

```
<%= render "common/navbar" %>
```

这样会使用 app/views/common/_navbar.html.erb 这个样式。

注意：如果使用 Partial 的样式和 Partial 所在的目录相同，可以省略第一段的 common 路径。但是不推荐这种做法，如果代码较多的话，过多省略容易导致混乱。

在 Partial 样式中是可以直接用实例变量(也就是@开头的变量)。不过好的做法是透过:locals 明确传递区域变量，这样程序比较清楚，partial 样式也比较容易被重复使用：

```
<%= render :partial => "common/nav", :locals => { :a => 1, :b => 2 } %>
```

在这个情况下，也可以进一步把 locals 省略：

```
<%= render "common/nav", :a => 1, :b => 2 %>
```

这样在 partial 样式中，就可以存取到区域变量 a 和 b。

参考

1. Action View - 樣板設計

August 1, 2016

招聘網站 6-8 部分 练习第 1 遍

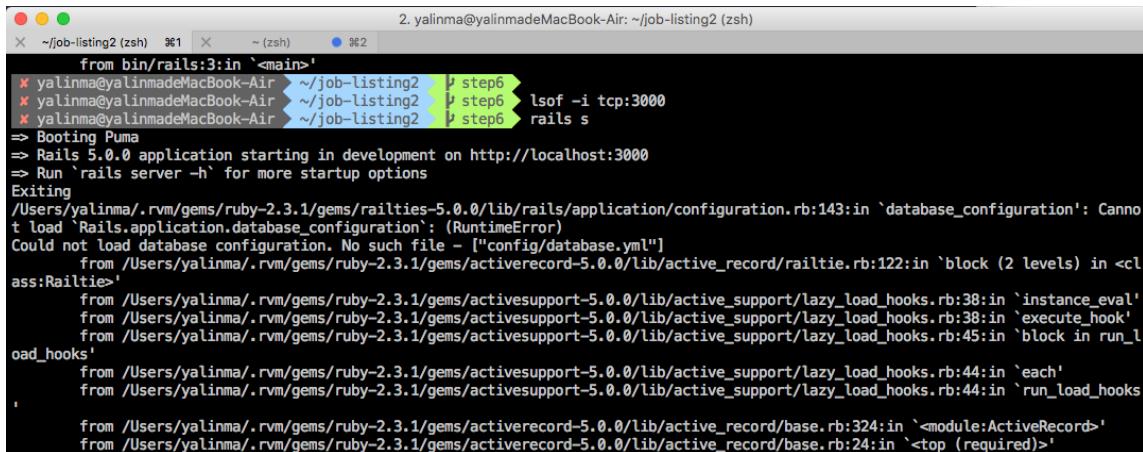
bug 疑问顺手记录一下，希望犯过的错误不要再犯，希望疑问在下一遍练习时能自动有答案。

(解答) 招聘網站第六部分

1. [OK] 在 rails s 的时候，遇到如下错误提示：

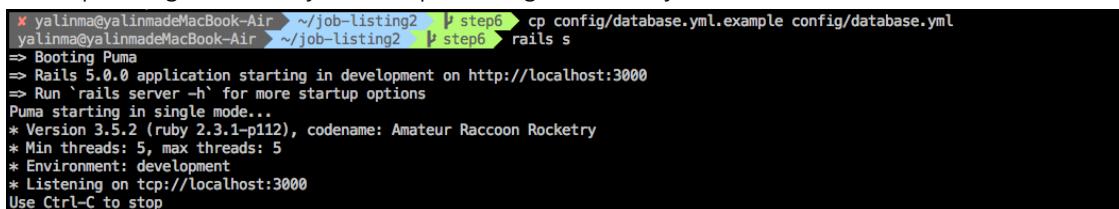
Cannot load `Rails.application.database_configuration`: (RuntimeError)

Could not load database configuration. No such file - ["config/database.yml"]



```
2. yalinma@yalinmadeMacBook-Air: ~/job-listing2 (zsh)
~/job-listing2 (zsh) 361 ~ (zsh) 362
from bin/rails:3:in `main'
x yalinma@yalinmadeMacBook-Air ~/job-listing2 ↵ step6
x yalinma@yalinmadeMacBook-Air ~/job-listing2 ↵ step6 lsof -i tcp:3000
x yalinma@yalinmadeMacBook-Air ~/job-listing2 ↵ step6 rails s
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:3000
=> Run 'rails server -h' for more startup options
Exiting
/Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/application/configuration.rb:143:in `database_configuration': Cannot load `Rails.application.database_configuration`: (RuntimeError)
Could not load database configuration. No such file - ["config/database.yml"]
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activerecord-5.0.0/lib/active_record/railtie.rb:122:in `block (2 levels) in <class:Railtie>'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activesupport-5.0.0/lib/active_support/lazy_load_hooks.rb:38:in `instance_eval'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activesupport-5.0.0/lib/active_support/lazy_load_hooks.rb:38:in `execute_hook'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activesupport-5.0.0/lib/active_support/lazy_load_hooks.rb:45:in `block in run_load_hooks'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activesupport-5.0.0/lib/active_support/lazy_load_hooks.rb:44:in `each'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activesupport-5.0.0/lib/active_support/lazy_load_hooks.rb:44:in `run_load_hooks'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activerecord-5.0.0/lib/active_record/base.rb:324:in `<module:ActiveRecord>'
    from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activerecord-5.0.0/lib/active_record/base.rb:24:in `<top (required)>'
```

运行 cp config/database.yml.example config/database.yml，server 正常开启。



```
x yalinma@yalinmadeMacBook-Air ~/job-listing2 ↵ step6 cp config/database.yml.example config/database.yml
yalinma@yalinmadeMacBook-Air ~/job-listing2 ↵ step6 rails s
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:3000
=> Run 'rails server -h' for more startup options
Puma starting in single mode...
* Version 3.5.2 (ruby 2.3.1-p112), codename: Amateur Raccoon Rocketry
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop
```

参考：[rails : Could not load database configuration. No such file -](#)

2. [OK] 提示 Migrations are pending.

ActiveRecord::PendingMigrationError

Migrations are pending. To resolve this issue, run: bin/rails db:migrate RAILS_ENV=development

Extracted source (around line #572):

```
570      # Raises <tt>ActiveRecord::PendingMigrationError</tt> error if any migrations are pending.
571      def check_pending!(connection = Base.connection)
572        raise ActiveRecord::PendingMigrationError if ActiveRecord::Migrator.needs_migration?(connection)
573      end
574
575      def load_schema_if_pending!
```

Rails.root: /Users/yalinma/job-listing2

修改 seed.rb, 跑 rake db:reset 就可以了。

参考：[Rails issue 解決筆記 - Migrations are pending](#)

出现以上 1-2 的问题，是因为你用 source tree 的时候，跳转分支，数据库要重建。你某个时间点数据库只能保持一个是正常的。所以这个是正常的流程，并不是操作异常导致的状况。每次你用 source tree 跳转分支，你都要重建数据库。

3. [OK] Jobs#Controller index 里面 Job.published.order.recent 多写了 order，提示异常。删掉就 OK 了。

ArgumentError in JobsController#index

The method .order() must contain arguments.

Extracted source (around line #11):

```
9       Job.published.order('wage_upper_bound DESC')
10      else
11        Job.published.order.recent
12      end
13
14
```

Rails.root: /Users/yalinma/job-listing2

(解答) 招聘網站第七部分

1. [OK] rail g model...少写了 model name resume 异常提示。rails g model resume...

```
x yalinma@yalinmadeMacBook-Air ~ ~/job-listing2 ↵ step7± ➤ rails g model job_id:integer user_id:integer content:text
Running via Spring preloader in process 2866
  invoke active_record
  /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/generators/base.rb:258:in `const_defined?': wrong constant name Job
Id:integer (NameError)
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/generators/base.rb:258:in `block in class_collisions'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/generators/base.rb:249:in `each'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/generators/base.rb:249:in `class_collisions'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/generators/named_base.rb:228:in `block in check_class_
collision'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/command.rb:27:in `run'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/invocation.rb:126:in `invoke_command'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/invocation.rb:133:in `block in invoke_all'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/invocation.rb:133:in `each'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/invocation.rb:133:in `map'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/invocation.rb:133:in `invoke_all'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/group.rb:232:in `dispatch'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/thor-0.19.1/lib/thor/invocation.rb:115:in `invoke'
```

2. [OK] gem "carrierwave" 拼错 导致异常。

```
x yalinma@yalinmadeMacBook-Air ~ ~/job-listing2 ↵ step7± ➤ bundle install
Fetching gem metadata from https://rubygems.org/
Fetching version metadata from https://rubygems.org/
Fetching dependency metadata from https://rubygems.org/
Could not find gem 'carrierware' in any of the gem sources listed in your Gemfile or
available on this machine.
```

3. [OK] rails g uploader attachment attachment 写错，导致后面步骤提示错误。

```
yalinma@yalinmadeMacBook-Air ~ ~/job-listing2 ↵ step7± ➤ rails g uploader attachment
Running via Spring preloader in process 3335
  create app/uploaders/attachment_uploader.rb
```

NameError in ResumesController#new

uninitialized constant Resume::AttachmentUploader

Extracted source (around line #5):

```
3   belongs_to :job
4
5   mount_uploader :attachment, AttachmentUploader
6
7   validates :content, presence: true
8 end
```

Rails.root: /Users/yalinma/job-listing2

August 1, 2016

[招聘網站 1-5 部分 练习第 4 遍](#)

Branch: new4

bug 疑问顺手记录一下，希望犯过的错误不要再犯，希望疑问在一遍练习时能自动有答案。

解答（第一天）

1. [OK] 心 得 。 安 装 gem "devise" 后 的 rake routes 。

```
yalinma@bogon ~/job-listing2 $ new4± rake routes
      Prefix Verb    URI Pattern          Controller#Action
        new_user_session GET   /users/sign_in(.:format)  devise/sessions#new
        user_session POST  /users/sign_in(.:format)  devise/sessions#create
      destroy_user_session DELETE /users/sign_out(.:format) devise/sessions#destroy
        user_password POST  /users/password(.:format) devise/passwords#create
      new_user_password GET   /users/password/new(.:format) devise/passwords#new
     edit_user_password GET   /users/password/edit(.:format) devise/passwords#edit
                  PATCH /users/password(.:format)  devise/passwords#update
                  PUT   /users/password(.:format)  devise/passwords#update
    cancel_user_registration GET   /users/cancel(.:format) devise/registrations#cancel
      user_registration POST  /users(.:format)  devise/registrations#create
    new_user_registration GET  /users/sign_up(.:format) devise/registrations#new
   edit_user_registration GET  /users/edit(.:format)  devise/registrations#edit
                  PATCH /users(.:format)  devise/registrations#update
                  PUT   /users(.:format)  devise/registrations#update
                  DELETE /users(.:format)  devise/registrations#destroy
      root GET   /           welcome#index
```

(解答) 招聘網站第二部分

1. [OK] before_filter :authenticate_user!, only: [:new, :create, :update, :edit, :destroy] 错写成了 only, 提示异常。
2. [OK] navbar.html.erb <=%> 输出内容, 写在了 li class 的位置。dropdown 显示异常。

```
<ul class="dropdown-menu">
  <li <%= link_to("Admin Panel", admin_jobs_path) %></li>
  <li <%= link_to("退出", destroy_user_session_path, method: :delete) %></li>
```



July 31, 2016

[第一周 周记](#)

混乱迷茫

刚上课感觉一片混乱，迷茫。哦，我 Rails101 没练完 3 遍，我要不要先练完再开始做新的作业；新的作业我要不要自己写，不看答案；新作业我还是照着答案练 3 遍吧？！我写博客花了好长时间，我要练习代码更长时时间才算正途吧？！总之，感觉混乱，不停在尝试新的策略。

迅速完成

问自己一个问题，假如 1 个小时或半个小时之后就要完成，你会怎么做？压缩自己的工作时间，想办法迅速做完最重要的几个点。并不是“好像没人催你，你就可以磨磨叽叽拖拖拉拉无边无际地做一件事”。时间是有成本的，时间是紧迫的。

拆解任务的诀窍

接触了 user story 的概念，觉得一切都有“捷径”，一切都有“方法论”一样。初始让你开发一个招聘网站，你会一脸懵逼，我要如何入手啊？！但是经过一周的学习，练习，你居然会觉得，嗯 我有思路了。拆出招聘网站的用户，然后拆出用户可能有的行为，然后一点一点完成。有趣，期待下周的课程。

重复练习

这是一个反复被强调的概念。不停的重复练练练。刚把这周的课练习了 3 遍，把练习中遇到的问题，解的 bug 都写了下来，发现自己会动不动打错一个字符，打错字符就出奇奇怪怪的小问题。嗯 明天我再练 2 遍，争取一镜到底，不出错。

对努力的误解

努力并不是把你一天的时间精力都耗尽才算数。保持一天的注意力比如 7 小时 在最重要的事情上，然后剩余的时间该听分享听分享，该玩玩，干点儿其他事情。

比如你上了一个收费很贵的课，你可能会一不小心地觉得“好像我要把自己累死才能收回成本一样”。不是这样的，你收回成本的地方在另外的地方，上课的正确姿态还是，保持自己精力旺盛，高效。

July 30, 2016

招聘網站 1-5 部分 练习第 3 遍

Branch: new3

bug 疑问顺手记录一下，希望犯过的错误不要再犯，希望疑问在下一遍练习时能自动有答案。

解答（第一天）

无。

（解答）招聘網站第一部分

无。

（解答）招聘網站第二部分

1. [OK]require_is_admin 方法里面，if 条件里面的!current_user.admin?，少些了否！，导致 admin 管理员被判定为“非管理员。”
2. def require_is_admin
3. if !current_user.admin?
4. flash[:alert] = 'You are not admin'
5. redirect_to root_path
6. end
7. end

（解答）招聘網站第三部分

1. [OK] job_params 方法，结尾多写了一个)，导致异常。

```
2.     def job_params
3.   params.require(:job).permit(:title, :description, :wage_lower_bound, :wage_upper_bound, :contact_email)
4. end
```

(解答) 招聘網站第四部分

- [OK] db/migrate 文档 里面多写了 def...change...end 导致 migrate 不能添加新的 column, 删掉就正常了。顺便明白了, rake db:drop 的意思, 是把所有之前 migrate 的过的, 都重新 migrate 一遍。

```
yalinma@bogon ~/job-listing2 [new3_step4±] $ rake db:drop
Dropped database 'db/development.sqlite3'
Database 'db/test.sqlite3' does not exist
yalinma@bogon ~/job-listing2 [new3_step4±] $ rake db:migrate
== 20160730071135 DeviseCreateUsers: migrating ==
-- create_table(:users)
  -> 0.0044s
-- add_index(:users, :email, {:unique=>true})
  -> 0.0014s
-- add_index(:users, :reset_password_token, {:unique=>true})
  -> 0.0009s
== 20160730071135 DeviseCreateUsers: migrated (0.0076s) ==

== 20160730073615 CreateJobs: migrating ==
-- create_table(:jobs)
  -> 0.0014s
== 20160730073615 CreateJobs: migrated (0.0015s) ==

== 20160730082349 AddIsAdminToUser: migrating ==
-- add_column(:users, :is_admin, :boolean, {:default=>false})
  -> 0.0015s
== 20160730082349 AddIsAdminToUser: migrated (0.0016s) ==

== 20160730084001 AddMoreDetailToJob: migrating ==
-- add_column(:jobs, :wage_upper_bound, :integer)
  -> 0.0015s
-- add_column(:jobs, :wage_lower_bound, :integer)
  -> 0.0008s
-- add_column(:jobs, :contact_email, :string)
  -> 0.0006s
== 20160730084001 AddMoreDetailToJob: migrated (0.0031s) ==

== 20160730085025 AddIsHiddenToJob: migrating ==
== 20160730085025 AddIsHiddenToJob: migrated (0.0000s) ==
```

```
1  class AddIsHiddenToJob < ActiveRecord::Migration[5.0]
2    def change
3      def change
4        add_column :jobs, :is_hidden, :boolean, default: true
5      end
6    end
7  end
```

```

yulinma@bogon ~/job-listing2 $ new3_step4± rake db:drop
Dropped database 'db/development.sqlite3'
Database 'db/test.sqlite3' does not exist
yulinma@bogon ~/job-listing2 $ new3_step4± rake db:migrate
== 20160730071135 DeviseCreateUsers: migrating ==
-- create_table(:users)
  -> 0.0056s
-- add_index(:users, :email, {:unique=>true})
  -> 0.0011s
-- add_index(:users, :reset_password_token, {:unique=>true})
  -> 0.0009s
== 20160730071135 DeviseCreateUsers: migrated (0.0078s) ==

== 20160730073615 CreateJobs: migrating ==
-- create_table(:jobs)
  -> 0.0018s
== 20160730073615 CreateJobs: migrated (0.0019s) ==

== 20160730082349 AddIsAdminToUser: migrating ==
-- add_column(:users, :is_admin, :boolean, {:default=>false})
  -> 0.0009s
== 20160730082349 AddIsAdminToUser: migrated (0.0010s) ==

== 20160730084001 AddMoreDetailToJob: migrating ==
-- add_column(:jobs, :wage_upper_bound, :integer)
  -> 0.0009s
-- add_column(:jobs, :wage_lower_bound, :integer)
  -> 0.0007s
-- add_column(:jobs, :contact_email, :string)
  -> 0.0007s
== 20160730084001 AddMoreDetailToJob: migrated (0.0025s) ==

== 20160730085025 AddIsHiddenToJob: migrating ==
-- add_column(:jobs, :is_hidden, :boolean, {:default=>true})
  -> 0.0008s
== 20160730085025 AddIsHiddenToJob: migrated (0.0009s) ==

```

(解答) 招聘網站第五部分

- [OK] admin 后台 index 里面的 class 拼错，导致 Publish 不是按钮形式显示。<%= link_to("Publish", publish_admin_job_path(job), :method => :post, :class => "btn btn-xs btn-default") %>

July 30, 2016

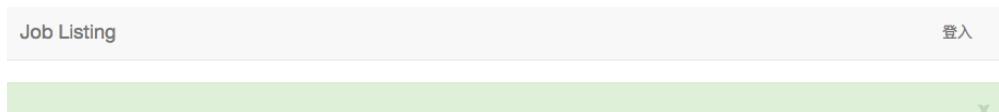
招聘網站 1-5 部分 练习第 2 遍

Branch: new2

bug 疑问顺手记录一下，希望犯过的错误不要再犯，希望疑问在下一遍练习时能自动有答案。

解答（第一天）

- [OK] “早安” 中文 引号 报错一次。
- [OK] flash 一 半 显 示 , 不 知 道 为 啥 ?

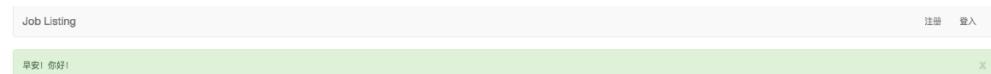


Hello World!

Copyright ©2016 Rails101
Design by yulin

因为

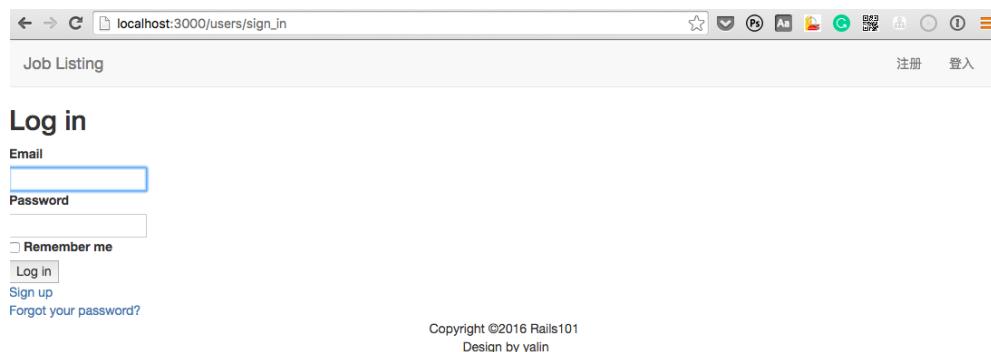
/Users/yalinma/job-listing2/app/views/common/_flashes.html.erb 里面少写了，<%= value %>，加上



就显示正常了。

(解答) 招聘網站第一部分

1. [OK] db 那里提示 user 已存在。rake db:drop rake db:migrate 然后就可以了。
2. [OK] routes.rb 那里的 resources 拼错。
3. [OK] /Users/yalinma/job-listing2/app/views/jobs/index.html.erb 里面的 jobs.each 错写成了 job.each
4. log-in 左 边 紧 挨 页 面 , 不 知 道 为 啥 ?



5. [OK] /Users/yalinma/job-listing2/app/controllers/jobs_controller.rb authenticate_user! 错误的输成 authenticate_user

(解答) 招聘網站第二部分

1. [OK] 在没有注册用户的情况下 用 rails c 把用户改为 admin，报错。

```
yalinma@bogon ~ ~/job-listing2 new2_step2± rails console
Running via Spring preloader in process 1214
Loading development environment (Rails 5.0.0)
2.3.1 :001 > u = User.first
  User Load (0.2ms)  SELECT "users".* FROM "users" ORDER BY "users"."id" ASC LIMIT ? [[["LIMIT", 1]]]
=> nil
2.3.1 :002 > u.is_admin = true
NoMethodError: undefined method `is_admin=' for nil:NilClass
  from (irb):2
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/commands/console.rb:65:in `start'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/commands/console_helper.rb:9:in `start'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/commands/commands_tasks.rb:78:in `console'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/commands/commands_tasks.rb:49:in `run_command!'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/railties-5.0.0/lib/rails/commands.rb:18:in `<top (required)>'
  from /Users/yalinma/.rvm/gems/ruby-2.3.1/gems/activesupport-5.0.0/lib/ac
```

2. [OK] /Users/yalinma/job-listing2/app/controllers/admin/jobs_controller.rbCRUD 里面的:create, 错写成了:create.

3. [OK] /Users/yalinma/job-listing2/app/controllers/admin/jobs_controller.rb CRUD 里面的 def...end, if else end, end 标错 漏写。

(解答) 招聘網站第三部分

无。

(解答) 招聘網站第四部分

1. [OK] /Users/yalinma/job-listing2/app/helpers/jobs_helper.rb 里面的 if job.is_hidden 错写成了 if render_job_status(job)
2. [OK] 心得 不论 jobs 还是 admin/jobs 下面的 index 都要 用 admin_job_path <%= link_to(job.title, admin_job_path(job)) %>

(解答) 招聘網站第五部分

无。

July 30, 2016

练习

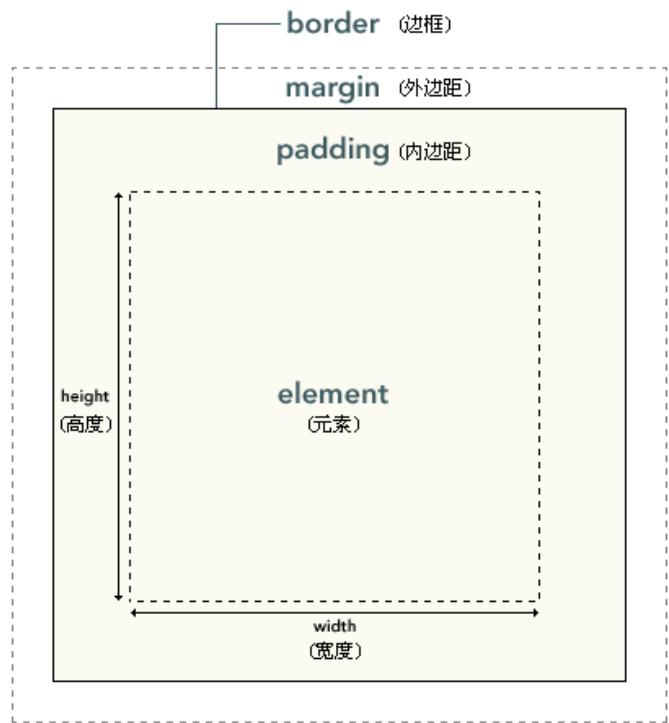
CSS 自学作业

1.margin 與 padding 的差異 ?

margin 代表外边距。padding 代表内边距，边框到内容的距离。

2.什麼是 box model ?

CSS 框模型 (Box Model) 规定了元素框处理元素内容、内边距、边框 和 外边距 的方式。



3.為何要使用 em 而非 px 來定義字的大小 ?

为了避免在 Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替 pixels。

1em 等于当前的字体尺寸。如果一个元素的 font-size 为 16 像素，那么对于该元素，1em 就等于 16 像素。在设置字体大小时，em 的值会相对于父元素的字体大小改变。

浏览器中默认的文本大小是 16 像素。因此 1em 的默认尺寸是 16 像素。

可以使用下面这个公式将像素转换为 em : pixels/16=em

(注 :16 等于父元素的默认字体大小, 假设父元素的 font-size 为 20px, 那么公式需改为 :pixels/20=em)

4.h1 {margin : 10px 0px 15px 5px;}

margin-top:

margin-right:

margin-left:

margin-bottom 各是多少 ?

margin 以顺时针方向上、右、下、左来设定值, 因此

margin-top: 10px

margin-right: 0px

margin-left: 5px

margin-bottom: 15px

July 29, 2016

[第一周遇过最大的坑](#)

花了太长时间写博客

我能够改 1 篇博客花 2 个小时, [写 1 篇博客](#)花 1 天。

我不知道我是不是已经走上了邪路, 通过和 Xdite 聊天, 我知道了另一种思路 : 想办法迅速做完最重要的事情。问自己一个问题, 假如 1 个小时或半个小时之后就要完成, 你会怎么做 ? 压缩自己的工作时间, 想办法迅速做完最重要的几个点。成长的初期尽量不要追求完美, 笔记暂时只对自己有用。

July 29, 2016

[总结 周记](#)

[第一周学到的最棒的概念](#)

练习到熟练

基础类的, 自己已经知道的, 要练到出手就能搞定。比如 Git 常用命令、Atom 常用命令等。假如练习得少, 这些会在你解决复杂问题的时候成为“障碍”, 你要不停地再翻翻查查, 不要这样, 抽空就练习几遍。

更进一步是默写或背起来。有时候不是你抄多几遍, 你就自动记住了。你要刻意去记忆。默写 背诵一下啊。

比如我默写了 Group CRUD 和 groups rake routes。对 RESTful 和 Controller#Action 的理解更进了一步。你会发现, 原来用 rake routes 跑出来的信息, 没有一点儿是多余的啊 ~

抄写和理解并不冲突。并不是你只有 1.5 时, 都那去理解了, 哪有时间抄写啊。这 2 个并不冲突, 你可以理解了, 但还是要不停的抄写, 练习到熟练。

July 29, 2016

[总结 周记](#)

[7/28 日记](#)

今天我最重大的领悟 : 想办法迅速做完最重要的事情。问自己一个问题, 假如 1 个小时或半个小时之后就要完成, 你会怎么做 ? 压缩自己的工作时间, 想办法迅速做完最重要的几个点。成长的初期尽量不要追求完美, 笔记暂时只对自己有用。要成才到出手就对别人有用, 还需要时间去累积。

July 28, 2016

[日记](#)

[HTML 基础概念](#)

1. 块级与内联元素

1.1 <div> 元素

1.2 元素

2. 属性

2.1 id 属性

2.2 class 属性

3. 段落

3.1 段落

3.2 断行

3.3 假如排版一首诗

4. Table

4.1 定义 Table

4.2 边框

4.3 字体

4.4 内边距

4.5 对齐

4.6 背景色

1. 块级与内联元素

1.1 <div> 元素

<div>用来定义文档中的一部分 (块)。听着很崩溃？！想象一下报纸里面的“豆腐块内容”的那个“块”。<div>用来定义网页上的“一块内容”。

<div>通常用作容器，来盛放其他 HTML 元素。<div>没有必须的 attributes，但可以定制 style 和 class。

[Read on →](#)

July 28, 2016

[HTML block table](#)

[7/27 日记](#)

全栈培训 Week : 1 Day: 3

今天我最重大的领悟：抄写和理解并不冲突啊。并不是你只有 1.5 时，都那去理解了，哪有时间抄写啊。这 2 个并不冲突，你可以理解了，但还是要不停的抄抄抄，练习到熟练。比如我最近比较讨厌的 view form，一来就 4 张表，index/new/show/edit，还巨多的鬼符%<=-，看了够够的。不管了，我找个模版，多抄几遍，明天见～

July 28, 2016

[7-26 日记](#)

全栈培训 Week : 1 Day: 2

今天我最重大的领悟：抄写和默写是完全不同的概念啊。不是你抄多几遍，你就自动记住了。该你记住的基础，就乖乖去默写。今天默写了 Group CRUD 和 groups rake routes。对 RESTful 和 Controller#Action 的理解更尽了一步。原来用 rake routes 跑出来的信息，没有一点儿是多余的啊，感觉之前的自己好“盲”啊～

Prefix	Verb	URI Pattern	Controller#Action
--------	------	-------------	-------------------

groups	GET	/groups(.:format)	groups#index
--------	-----	-------------------	--------------

POST		/groups(.:format)	groups#create
------	--	-------------------	---------------

```
new_group GET      /groups/new(.:format)      groups#new
edit_group GET     /groups/:id/edit(.:format) groups#edit
group GET       /groups/:id(.:format)      groups#show
PATCH   /groups/:id(.:format)      groups#update
PUT     /groups/:id(.:format)      groups#update
DELETE  /groups/:id(.:format)      groups#destroy
root GET      /                  groups#index
```

July 26, 2016

7-25 ORID 日记

Objective

關於今天的課程, 你記得什麼? => 普通人学习程序失败的流程真的触目惊心。

完成了什麼? => 通过 fork 一个项目, 然后 pull request 来更新一个版本。

Reflective

你要如何形容今天的情緒

今天的高峰是什麼? => 可以撰写 [fullstack-course wiki](#) 好开心。

今天的低點是什麼? => 在明显没有出错的情况下, 最后一步因为多做了很多步, 最后变成踩坑事件 ; 不得不重新 fork, 照着答案贴 code, 再 pull request。

Interpretive

我們今天學到了什麼? => 大问题如何拆解成可以完成的小任务。

今天一個重要的領悟是什麼? => 基础类的, 自己已经知道的, 要练到出手就能搞定。比如 Git 常用命令、Atom 常用命令、CRUD 等, 假如练习得少, 这些会在你解决复杂问题的时候成为“障碍”, 你要不停地再翻查查, 不要这样, 抽空就练习几遍。

Decisional

我們會如何用一句話形容今天的工作 => 时间紧迫啊。

有哪些工作需要明天繼續努力? => 作业。

July 25, 2016

git is not a git command

因为把 git clone 写成了 git 收到了这样一条错误提示 git is not a git command。回去翻, 发现果然每条命令都是前面带上 git , git 相当于只是 git command 的前缀。

```
[yalinmadeMacBook-Air:~ yalinma$ git git@github.com:mayalin/job-listing.git
git: 'git@github.com:mayalin/job-listing.git' is not a git command. See 'git --help'.]
```

犯了一次这样的低级错误, 又写了这篇笔记, 又把常用 git 命令抄写 3 遍, 应该以后不会再犯这样的错误了吧 ? ! 或者至少在出错第一时间能识别出来。

July 25, 2016

Git

TryGit - 15 分钟学会基本的 Git 命令

TryGit 是一个有趣的线上练习 Git 基本命令的小教程, 是 Code School 和 Github 合作出的作品。过完 25 关, 你就掌握了 Git 最常用的命令。



1.1 · Got 15 minutes and want to learn Git?

Git allows groups of people to work on the same documents (often code) at the same time, and without stepping on each other's toes. It's a distributed version control system.

Our terminal prompt below is currently in a directory we decided to name "octobox". To initialize a Git repository here, type the following command:

```
git init
```



The screenshot shows a terminal window with the command `git init` entered. Below the terminal is a help panel titled "My Octobox Repository" with sections for "Advice" and "Commands". The "Advice" section defines "Directory" as a folder for multiple files and "Repository" as a directory initialized for version control. The "Commands" section lists basic Git commands like `git clone`, `git add`, and `git commit`.

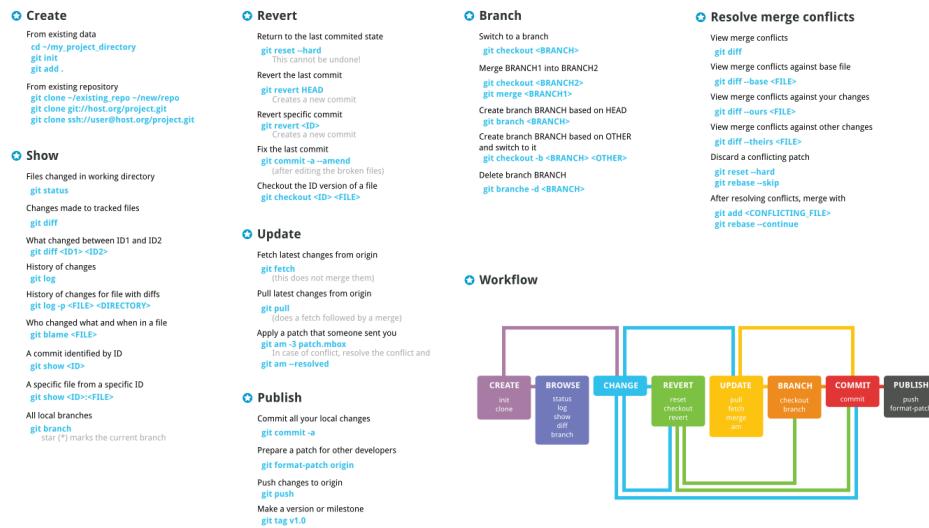
July 21, 2016

Git

Git Cheat Sheet

最近改变了策略。不懂的 没关系 背起来。学习 Git 也一样，找一份 [Git Cheat Sheet](#)，抄写 3 遍，先迅速知道怎么用的。来用去脉放在下一步。

Git Cheat Sheet



Made by Hylique Bons based on work by Zack Rusin and Sébastien Pierre. This work is licensed under the Creative Commons Attribution 3.0 License.

资源

1. [Git Cheat Sheet - fournova](#)
2. [猴子都能懂的 GIT 入门 | 贝格乐 \(Backlog\)](#)
3. [Git 教程 - 廖雪峰的官方网站](#)
4. [Git - Documentation](#)

July 21, 2016

[Git Cheat-Sheet](#)

[快速习得一门编程语言的方法论](#)

最近有幸尝试了 Xdite 快速学习的理念。以下是我实践后的总结版：

原则

想尽一切办法成长到下一个阶段，想尽一切办法把你学习的东西变成自己的永久技能。

操作方法

第 1 阶段：贴代码，跑通程序。1 个时间单位。

第 2 阶段：敲代码，跑通程序。2 个时间单位。

第 3 阶段：敲代码，搞懂代码含义。重复 3 次，一共 6 个时间单位。

- 弄懂代码含义。看说明。问助教。问行业前辈。问 Google。
- 写笔记。记录自己懂了的部分，出错的地方。运用写笔记，把这个变成自己的永久技能。
- 自己暂时不懂的 没关系，背起来。

其他说明

反复练习不是什么？ 反复练习不是盲目重复第 1 第 2 阶段 5 次。就好比你一直在小学，一直不升学一样。

第 1 第 2 阶段都 1 遍迅速完工，这些是基本功阶段。第 3 阶段开始重复进行 3 次。不停的抄写代码，想办法弄懂更多的含义。

可不可以跳过第 1 阶段？ 对新手来说不可以。因为你新接触一门语言，直接上来就敲代码，很容易出错。代码出错，程序跑不通，你会很慌张很沮丧，会觉得学这个东西非常受挫！

可不可以跳过第 2 阶段？ 也不要吧！这一阶段主要让你适应新的语言，可能你并不理解含义，但没关系，你只要照着抄写，练习，形成肌肉记忆就可以了。给你下一阶段的练习一个确信：我代码是可以抄写正确的。

重复练习做不下去该怎么办？ 填写追踪表，比对你每次做练习花费的时间。填过你会发现，你毫无规划一股脑去练习瞎忙碌的时候，其实花了很长时间，但未必真的高效率。一刻不停忙碌只是给自己一种虚幻的努力感觉。等你尝试有计划地练习 3 次，你会非常惊讶的发现你练 3 次的时间反而比你瞎练一次要花更少的时间，且效果更好。

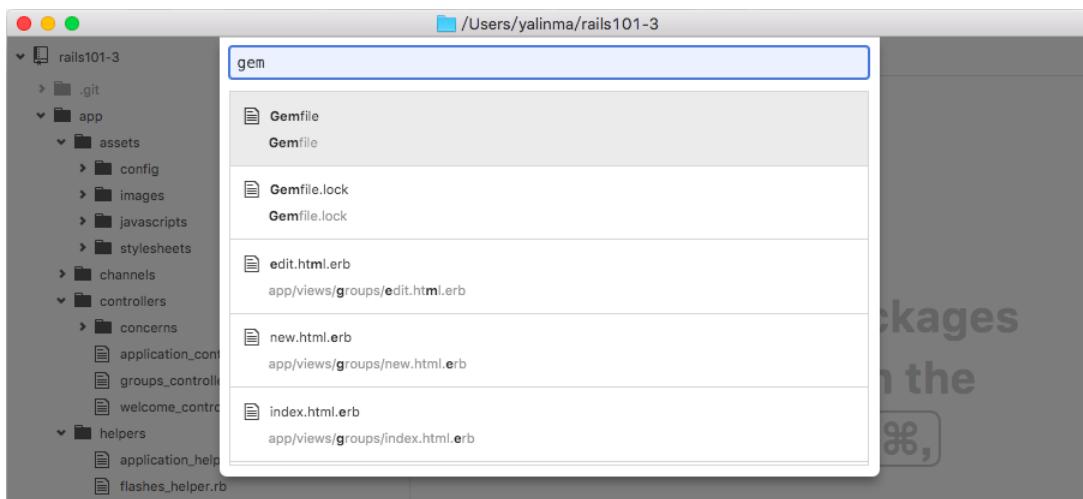
July 20, 2016

[快速学习 方法论 xdite](#)

[Atom Editor Cheat Sheet](#)

当你用 [Atom](#) 看代码的时候，知道一些常用的快捷键会让看代码改代码过程变得高效愉悦很多。

比如，你要修改 Gemfile 时，按 `⌘P` 查找文件，输入 Gemfile，按 enter，直接进入 Gemfile 进行修改。又由于输入框里使用的是 Fuzzy matching，你输入到 gem 的时候，估计 Gemfile 已经排在第一位了，当它是第一位的时候，直接按 enter 就可以了。



再比如，你要修改 groups_controller.rb 里面的 create 方法时，按 ⌘R 查找方法，输入 create，按 enter，直接进入 create 所在行进行修改。同样由于 Fuzy matching，你输入到 crea 的时候，create 方法可能已经排在第一位了，直接按 enter 就可以了。

A screenshot of a code editor window titled 'groups_controller.rb — /Users/yalinma/rails101-3'. The search bar at the top contains the text 'crea'. The code editor displays the following Ruby code:

```
5
6  def new
7      @group = Group.new
8  end
9
10 def show
11     @group = Group.find(params[:id])
12 end
13
14 def edit
15     @group = Group.find(params[:id])
16 end
17
18 def create
19     @group = Group.new(group_params)
20
```

The 'create' method is highlighted in the code area, indicating it is the current target for modification.

Atom Editor Cheat Sheet

File

Save All ⌘⌥S Option + Command + S

Close Tab ⌘W Command + W

Close Window ⌘⌘W Shift + Command + W

Edit

Copy Path ⌘⏏C Control + Shift + C

Move to Top ⌘↑ Command + ↑

Move to Bottom ⌘↓ Command + ↓

View

Toggle Tree View ⌘\ Command + \

Selection

Select Line ⌘L Command + L

Select to End of Line ⌘→ Shift + Command + →

Find

Find File ⌘P Command + P

Find Method ⌘R Command + R

Find in Project ⌘⌘F Shift + Command + F

1

[Download PDF](#)

July 18, 2016

[Atom Cheat-Sheet](#)

[Rails view](#)

[Rails 学习笔记](#)

65 / 88

增加投票 view

app/views/topics/index.html.erb

```
15      <% @topics.each do |topic| %>
16        <tr>
17          <td><%= topic.title %></td>
18          <td><%= topic.description %></td>
19          <td><%= pluralize(topic.votes.count, "vote") %></td>
20          <td><%= button_to '+1', upvote_topic_path(topic), method: :post %></td>
21          <td><%= link_to 'Show', topic %></td>
22          <td><%= link_to 'Edit', edit_topic_path(topic) %></td>
23          <td><%= link_to 'Destroy', topic, method: :delete, data: { confirm: 'Are
24            you sure?' } %></td>
25        </tr>
26      <% end %>
```

pluralize(topic.votes.count, "vote")会输出票数，并且根据单复数在后面加上 vote 或 votes。

button_to '+1'加一个 HTML 的按钮 (button)，里面有字'+1'。

upvote_topic_path(topic) 产生我们要调用的操作的 URL。这里我们要对目前的 topic 投票，回传 /topics/42/upvote。

method: :post 确保我们使用的是 CRUD 里面的 create 操作，而非 read。

降低投票 view

<td><%= button_to '-1', downvote_topic_path(topic), method: :delete %></td>

button_to '-1'加一个 HTML 的按钮 (button)，里面有字'-1'。

downvote_topic_path(topic) 产生我们要调用的操作的 URL。这里我们要对目前的 topic 投票，回传 /topics/42/downvotee。

method: :destroy 确保我们使用的是 CRUD 里面的 destroy 操作。

超链接

app/views/topics/index.html.erb

<td><%= topic.title %></td>

改成这样：

<td><%= link_to topic.title, topic %></td>

意思是 topic.title 链接到 topic

flash 消息

HTML tag，下面加入：

app/views/layouts/application.html.erb

```
14  </html>
15  <% flash.each do |name, msg| %>
16    <div><%= msg %></div>
17  <% end %>
```

July 18, 2016

[Rails 初级 全栈](#)

[Rails controller](#)

增加投票

在 controller 里面自定义 upvote method，加在 private 上面。

app/controllers/topics_controller.rb

```
63  def upvote
64    @topic = Topic.find(params[:id])
65    @topic.votes.create
66    redirect_to(topics_path)
67  end
68  private
```

@topic = Topic.find(params[:id]) 从数据库里用 id 找到 topic，然后存进变量@topic 里面。

@topic.votes.create 给目前这篇 topic 新增一笔投票记录，然后存进数据库。

redirect_to(topics_path) 告诉浏览器回到 topics_path (topics 的列表)。

降低投票

在 controller 里面自定义 downvote method，加在 private 上面。

app/controllers/topics_controller.rb

```
68  def downvote
69    @topic = Topic.find(params[:id])
70    @topic.votes.destroy
71    redirect_to(topics_path)
72  end
73  private
```

@topic = Topic.find(params[:id]) 从数据库里用 id 找到 topic，然后存进变量@topic 里面。

@topic.votes.destroy 给目前这篇 topic 删除一笔投票记录，然后存进数据库。

redirect_to(topics_path) 告诉浏览器回到 topics_path (topics 的列表)。

回到话题列表

新增 (create) topic 后可以回到 topics 列表。告诉浏览器回到 topics_path (topics 的列表)。

找到 create method, format.html { redirect_to @topic, notice: 'Topic was successfully created.' }，把 @topic 改成 topics_path，像这样：format.html { redirect_to topics_path, notice: 'Topic was successfully created.' }

app/controllers/topics_controller.rb

```
26  def create
27    @topic = Topic.new(topic_params)
28
29    respond_to do |format|
30      if @topic.save
31        format.html { redirect_to topics_path, notice: 'Topic was successfully created.' }
32        format.json { render :show, status: :created, location: @topic }
33      else
34        format.html { render :new }
35        format.json { render json: @topic.errors, status: :unprocessable_entity }
36      end
37    end
38  end
```

编辑 (edit) topic 后可以回到 topics 列表。

找到 update method, format.html { redirect_to @topic, notice: 'Topic was successfully updated.' }，把 @topic 改成 topics_path，像这样：format.html { redirect_to topics_path, notice: 'Topic was successfully updated.' }

app/controllers/topics_controller.rb

```

42  def update
43    respond_to do |format|
44      if @topic.update(topic_params)
45        format.html { redirect_to topics_path, notice: 'Topic was successfully updated.' }
46        format.json { render :show, status: :ok, location: @topic }
47    else
48      format.html { render :edit }
49      format.json { render json: @topic.errors, status: :unprocessable_entity }
50    end
51  end
52 end

```

format.html { redirect_to topics_path, notice: 'Topic was successfully created.' } :

format.html 指的是当浏览器向 server 请求 HTML 格式的时候要做的事（要做的事在{}里面）。

redirect_to topics_path 指的是在新增或编辑完成一篇 topic 之后要回到 topics 列表页面。

notice: 'Topic was successfully created/updated.' 把信息丢到 flash 里面，这样回到 topics 列表页面的时候就会看到。

July 18, 2016

Rails 初级 全栈

Rails associations

在 Rails 里，model 和 model 之间的关联叫做 associations。Associations 通常成对出现。

has_many

一个 topic 可以有许多 votes，所以 topic model 里面写入 has_many :votes，其中:votes 代表 vote model 的表格 id。

```

1 class Topic < ActiveRecord::Base
2   has_many :votes, dependent: :destroy
3 end
4

```

dependent:

当自己被删除时，删除相关的资料也很重要。has_many :votes 之后的 dependent: :destroy 意思是说，当一个 topic 被删除 (destroy) 时，它的投票记录 (votes) 也要一起删除。:destroy 代表 destroy 方法。

```

1 class Topic < ActiveRecord::Base
2   has_many :votes, dependent: :destroy
3 end
4

```

belongs_to

一个 vote 对应到某个 topic，所以 vote model 里面写 belongs_to :topic，其中:topic 代表 topic model 的 id。

```

1 class Vote < ActiveRecord::Base
2   belongs_to :topic
3 end

```

参考

1. [Active Record Query Interface — Ruby on Rails Guides](#)

July 17, 2016

Rails 初级 全栈

Rails routes

Rails routes 会列出所有你定义的 routes, 用于追踪路由问题或者让你查看 app 中所有的 URLs。
routes

rake routes

这条命令帮我们列出了 app 中所有的 URLs, topics#index 代表 topics 列表页。

```
[yalinmadeMacBook-Air:suggestotron yalinma$ rake routes
Prefix Verb URI Pattern Controller#Action
topics GET /topics(.:format) topics#index
       POST /topics(.:format) topics#create
new_topic GET /topics/new(.:format) topics#new
edit_topic GET /topics/:id/edit(.:format) topics#edit
topic GET /topics/:id(.:format) topics#show
       PATCH /topics/:id(.:format) topics#update
       PUT /topics/:id(.:format) topics#update
       DELETE /topics/:id(.:format) topics#destroy]
```

首页

在 config/routes.rb 里使用 root 'topics#index' 把首页设为 topics 列表页：

```
routes.rb
1 Rails.application.routes.draw do
2   root 'topics#index'
3   resources :topics
4   # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
5 end
```



增加投票 route

修改 config/routes.rb 里面的 resources :topics, 添加 upvote 的 route。

```
3   resources :topics do
4     member do
5       post 'upvote'
6     end
7   end
```

现在检查 route 是否有添加成功，输入 rake routes 或打开 <http://localhost:3000/rails/info>。你应该能看到有一行：

Prefix Verb URI Pattern Controller#Action

upvote_topic POST /topics/:id/upvote(.:format) topics#upvote

降低投票 route

再添加 downvote 的 route。

```
3   resources :topics do
4     member do
5       post 'upvote'
6       post 'downvote'
7     end
8   end
```

现在检查 route 是否有添加成功，输入 rake routes 或打开 <http://localhost:3000/rails/info>。你应该能看到有一行：

Prefix Verb URI Pattern Controller#Action

downvote_topic POST /topics/:id/downvote(.:format) topics#downvote

参考

1. [The Rails Command Line - Rails Guides](#)

July 17, 2016

[Rails 初级 全栈](#)

[Rails generate](#)

Rails generate 命令使用模版来创建一整套东西。

Scaffold

Scaffold 是一整套的 Model, View 和 Controller 以及 Database 等。

rails generate scaffold topic title:string description:text

这里我们创建一个名为 topic 的 scaffold，并定义 2 个项目：title 和 description。

看看这条命令自动帮我们生成了什么：

- Model: topic 路径 : app/models/topic.rb
- View: topics 路径 : app/views/topics
- Controller: topics_controller 路径 : app/controllers/topics_controller.rb
- Database: 20160717121240 CreateTopics 路径 : db/migrate/20160717121240_create_topics.rb
- 其他一些辅助组件等。

```
[yalinmadeMacBook-Air:suggestotron yalinma$ rails generate scaffold topic title:s]
string description:text
Running via Spring preloader in process 2430
  invoke  active_record
  create    db/migrate/20160717121240_create_topics.rb
  create    app/models/topic.rb
  invoke  test_unit
  create    test/models/topic_test.rb
  create    test/fixtures/topics.yml
  invoke  resource_route
  route    resources :topics
  invoke  scaffold_controller
  create    app/controllers/topics_controller.rb
  invoke  erb
  create    app/views/topics
  create    app/views/topics/index.html.erb
  create    app/views/topics/edit.html.erb
  create    app/views/topics/show.html.erb
  create    app/views/topics/new.html.erb
  create    app/views/topics/_form.html.erb
  invoke  test_unit
  create    test/controllers/topics_controller_test.rb
  invoke  helper
  create    app/helpers/topics_helper.rb
  invoke  test_unit
  invoke  jbuilder
  create    app/views/topics/index.json.jbuilder
  create    app/views/topics/show.json.jbuilder
  invoke  assets
  invoke  coffee
  create    app/assets/javascripts/topics.coffee
  invoke  scss
  create    app/assets/stylesheets/topics.scss
  invoke  scss
  create    app/assets/stylesheets/scaffolds.scss
yalinmadeMacBook-Air:suggestotron yalinma$ ]
```

rake db:migrate

更新数据库，创建名为 topics 的表格。

```
[yalinmadeMacBook-Air:suggestotron yalinma$ rake db:migrate
== 20160717121240 CreateTopics: migrating =====
-- create_table(:topics)
  -> 0.0017s
== 20160717121240 CreateTopics: migrated (0.0018s) =====
yalinmadeMacBook-Air:suggestotron yalinma$ ]
```

Model

rails generate model vote topic_id:integer

这里我们创建一个名为 vote 的 model，并定义 1 个项目：topic_id。

看看这条命令自动帮我们生成了什么：

- Model: vote 路径：app/models/vote.rb
- Database: 20160717133631 CreateVotes 路径：db/migrate/20160717133631_create_votes.rb
- 其他一些辅助组件等。

```
[yalinmadeMacBook-Air:suggestotron yalinma$ rails generate model vote topic_id:integer
Running via Spring preloader in process 2615
  invoke  active_record
    create    db/migrate/20160717133631_create_votes.rb
    create    app/models/vote.rb
    invoke    test_unit
    create      test/models/vote_test.rb
    create      test/fixtures/votes.yml
```

rake db:migrate

更新数据库，创建名为 votes 的表格。

```
[yalinmadeMacBook-Air:suggestotron yalinma$ rake db:migrate
== 20160717133631 CreateVotes: migrating =====
-- create_table(:votes)
  -> 0.0103s
== 20160717133631 CreateVotes: migrated (0.0104s) =====
```

参考

1. [The Rails Command Line - Rails Guides](#)

July 17, 2016

[Rails 初级 全栈](#)

[基础 Ruby](#)

介绍 Ruby 的一些基本语法操作。

1. 进入 irb
2. 四则运算
3. 变量
4. 数组
5. 类
6. 方法
7. 循环
8. 条件
9. 自定义方法

进入 irb

在终端输入 irb，进入 Ruby 编程工具（Interactive Ruby Shell）。

```
● ○ ●  yalinma — irb rvm_bin_path=/Users/yalinma/.rvm/bin — 80x24
Last login: Sun Jul 17 14:15:16 on ttys000
[yalinmadeMacBook-Air:~ yalinma$ irb
2.3.1 :001 > ]
```

输入 exit，可以退出 Ruby 编程工具。

```
[2.3.1 :001 > exit
yalinmadeMacBook-Air:~ yalinma$ ]
```

四则运算

做一些简单的四则运算，输入：

1 + 1

4 - 1

3 * 2

8 / 2

```
[yalinmadeMacBook-Air:~ yalinma$ irb
[2.3.1 :001 > 1 + 1
 => 2
[2.3.1 :002 > 4 - 1
 => 3
[2.3.1 :003 > 3 * 2
 => 6
[2.3.1 :004 > 8 / 2
 => 4]
```

变量

Variables (变量) 用来存储一个可变的值。

x = 3, 创建一个初始值为 3 的变量, x。

```
[2.3.1 :005 > x = 3
=> 3]
```

对变量做运算, 并重新赋值 :

x = x + 5

x = x * 4

```
[2.3.1 :008 > x = x + 5
=> 8
[2.3.1 :009 > x = x * 4
=> 32]
```

数组

Array (数组) 用来存储多个值。

fruits = ["apple", "banana", "cherry"], 创建一个初始包含 3 种水果的数组, fruits。

```
[2.3.1 :033 > fruits = ["apple", "banana", "cherry"]
=> ["apple", "banana", "cherry"]]
```

对数组进行添加移除项目操作, 并重新赋值 :

fruits = fruits + ["lemon"]

fruits = fruits - ["cherry"]

```
[2.3.1 :034 > fruits = fruits + ["lemon"]
=> ["apple", "banana", "cherry", "lemon"]
[2.3.1 :035 > fruits = fruits - ["cherry"]
=> ["apple", "banana", "lemon"]]
```

类

看看我们刚刚都用到了什么 Class (类)。

x.class

"apple".class

fruits.class

```
[2.3.1 :036 > x.class
=> Fixnum
[2.3.1 :037 > "apple".class
=> String
[2.3.1 :038 > fruits.class
=> Array]
```

方法

每个类都有 Methods (方法)。

fruits.methods 看看我们创建的数组都有哪些方法：

```
[2.3.1 :039 > fruits.methods
=> [:fill, :assoc, :rassoc, :uniq, :uniq!, :compact, :compact!, :flatten, :to_h
, :flatten!, :shuffle!, :shuffle, :include?, :combination, :repeated_permutation
, :permutation, :product, :sample, :repeated_combination, :bsearch_index, :bsear
ch, :select!, :&,amp;, :*, :+, :-, :sort, :count, :find_index, :select, :reject, :col
lect, :map, :pack, :first, :any?, :reverse_each, :zip, :take, :take_while, :drop
, :drop_while, :cycle, :insert, :|, :index, :rindex, :replace, :clear, :<=>, :<<
, :==, :[], :[][], :reverse, :empty?, :eql?, :concat, :reverse!, :inspect, :delet
e, :length, :size, :each, :slice, :slice!, :to_ary, :to_a, :to_s, :dig, :hash, :at
, :fetch, :last, :push, :pop, :shift, :unshift, :frozen?, :each_index, :join,
:rotate, :rotate!, :sort!, :collect!, :map!, :sort_by!, :keep_if, :values_at, :d
elete_at, :delete_if, :reject!, :transpose, :find, :entries, :sort_by, :grep, :g
rep_v, :detect, :find_all, :flat_map, :collect_concat, :inject, :reduce, :partit
ion, :group_by, :all?, :one?, :none?, :min, :max, :minmax, :min_by, :max_by, :mi
nmax_by, :member?, :each_with_index, :each_entry, :each_slice, :each_cons, :each
```

fruits.length 计算了我们数组中包含的项目数。

```
[2.3.1 :040 > fruits.length
=> 3
```

循环

Array 有个 method 叫做 each，会遍历它里面每一项并执行操作。

```
fruits.each do |fruit|
  puts fruit
end
```

这会把 fruits 里面的第一项 ("apple") 取出，赋值给变量 fruit，并用 puts 操作输出 fruit 当前值。然后对其他项重复这些操作。上面这段程序会列出 fruits 里的项目。

```
[2.3.1 :041 > fruits.each do |fruit|
[2.3.1 :042 >   puts fruit
[2.3.1 :043?> end
apple
banana
lemon
=> ["apple", "banana", "lemon"]
```

条件

Condition (条件判断) 只有在条件满足时才执行，这里介绍 if 语句。

```
if x > 1
  puts "x is bigger than 1."
end
```

上面的程序要执行的是：当 x>1 时，输出"x is bigger than 1."。我们知道我们 x 的值是 32，32 大于 1，所以这段输出程序会被执行。

```
[2.3.1 :026 > if x > 1
[2.3.1 :027?>   puts "x is bigger than 1."
[2.3.1 :028?> end
x is bigger than 1.
=> nil
```

自定义方法

这里我们创建了一个 pluralize 方法，针对传入的参数 word，自动增加一个's'，变为复数。

```
def pluralize(word)
  word + "s"
```

```
end
```

使用自定义方法：

```
pluralize("apple")
```

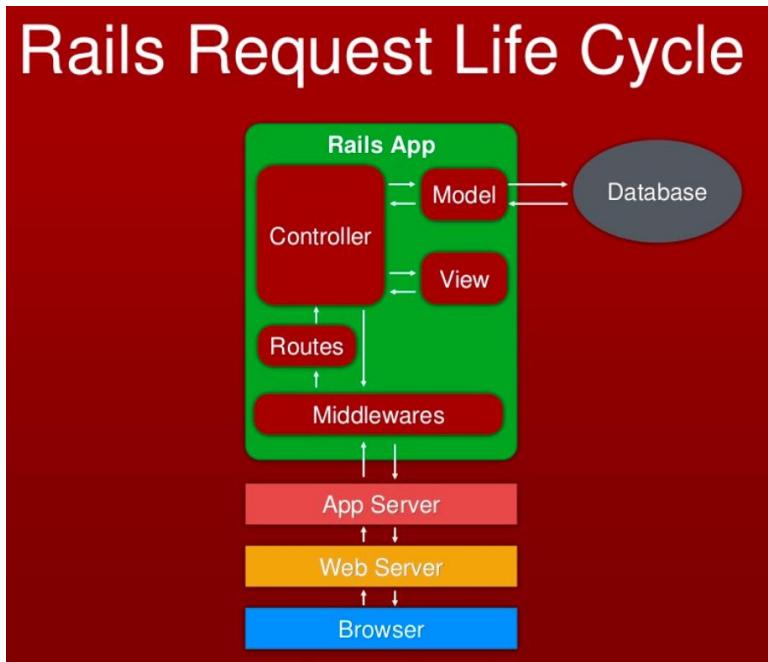
```
[2.3.1 :029 > def pluralize(word)
[2.3.1 :030?>   word + "s"
[2.3.1 :031?>   end
=> :pluralize
[2.3.1 :032 > pluralize("apple")
=> "apples" _
```

July 17, 2016

[Ruby IRB 语法](#)

[Rails 请求的生命周期](#)

这张图好有趣，感觉一眼能看清好多逻辑。



概念

1. **Model** 表示应用程序核心（比如数据库记录列表）。
2. **View** 显示数据。
3. **Controller** 输入和输出的路由，从页面取数据插入到相关的 model，以及从 model 拿数据渲染相关的 view。
4. **Routes** 用于在网络中选择最优路径。
5. **Middlewares** 提供操作系统和应用软件之间服务的软件，可以被描述为“软件胶”。
6. **Database** 数据库是有组织的数据集合，是模式，表，查询，报表视图和其他对象的集合。
7. **App Server** 你在 app server 上运行你的程序。比如你用 rails s 生成了一个本地 app server 来运行你的程序，比如你用 heroku app 创建一个 heroku app server 来运行你的程序。
8. **Web Server** 哟，我们要把 app server 放到更大更广阔的网络，需要 web server 来干好这个工作。
9. **Browser** 比如你常用的 Safari、Chrome 和 Firefox 等。

参考

1. [Rails Request & Middlewares](#)

2. [MVC 架构 - Chef 之道](#)
3. [Routing - Wikipedia, the free encyclopedia](#)
4. [Middleware - Wikipedia, the free encyclopedia](#)
5. [Database - Wikipedia, the free encyclopedia](#)
6. [Configure Heroku App for Custom Domain](#)

July 17, 2016

Rails 中级 全栈

Rake aborted!

rake aborted!

在将 app 上传 Heroku，数据库迁移时出过这样一种异常。前面指令都没有任何异常，只卡到数据库迁移时提示异常。

```
[yalinmadeMacBook-Air:first_app yalinma$ heroku run rake db:migrate ]  
Running rake db:migrate on ⚙safe-dusk-58912... up, run.3442  
rake aborted!  
Gem::LoadError: Specified 'postgresql' for database adapter, but the gem is not  
loaded. Add `gem 'pg'` to your Gemfile (and ensure its version is at the minimum  
required by ActiveRecord).  
/app/vendor/bundle/ruby/2.2.0/gems/activerecord-5.0.0/lib/active_record/connecti  
on_adapters/connection_specification.rb:176:in `rescue in spec'  
/app/vendor/bundle/ruby/2.2.0/gems/activerecord-5.0.0/lib/active_record/connecti  
on_adapters/connection_specification.rb:173:in `spec'  
/app/vendor/bundle/ruby/2.2.0/gems/activerecord-5.0.0/lib/active_record/connecti  
on_handling.rb:53:in `establish_connection'  
/app/vendor/bundle/ruby/2.2.0/gems/activerecord-5.0.0/lib/active_record/railtie.  
rb:125:in `block (2 levels) in <class:Railtie>'
```

仔细看描述，提示“gem 未下载，添加 gem pg”，奇怪我记得我当时有在 Gemfile 添加 pg 啊！察看 Gemfile 发现是“pg”前面没有加 gem。

```
47 group :production do  
48   "pg"  
49 end
```

正确的写法应该是：

```
47 group :production do  
48   gem "pg"  
49 end
```

修改 Gemfile 后依次跑以下指令，就可以了。

1. bundle install
2. git add .
3. git commit -m "add gem before pg"
4. git push heroku master
5. heroku run rake db:migrate
6. heroku open



July 17, 2016

[Rails Heroku Gem](#)

[上传 Heroku](#)

命令行

1. bundle install
2. git add .
3. git commit -m "描述你修改了什么"
4. git push heroku master
5. heroku run rake db:migrate
6. heroku open

Step0: 登陆 Heroku (首次)

首次使用, 请先登入 [Heroku](#) 注册账号, 并安装[命令行工具](#)。之后在终端输入 heroku login , 使用账号密码登陆 heroku。接着输入 heroku keys:add 添加 SSH-key。这些操作只在首次使用 heroku 时进行, 后续再使用就无需做这些操作了。

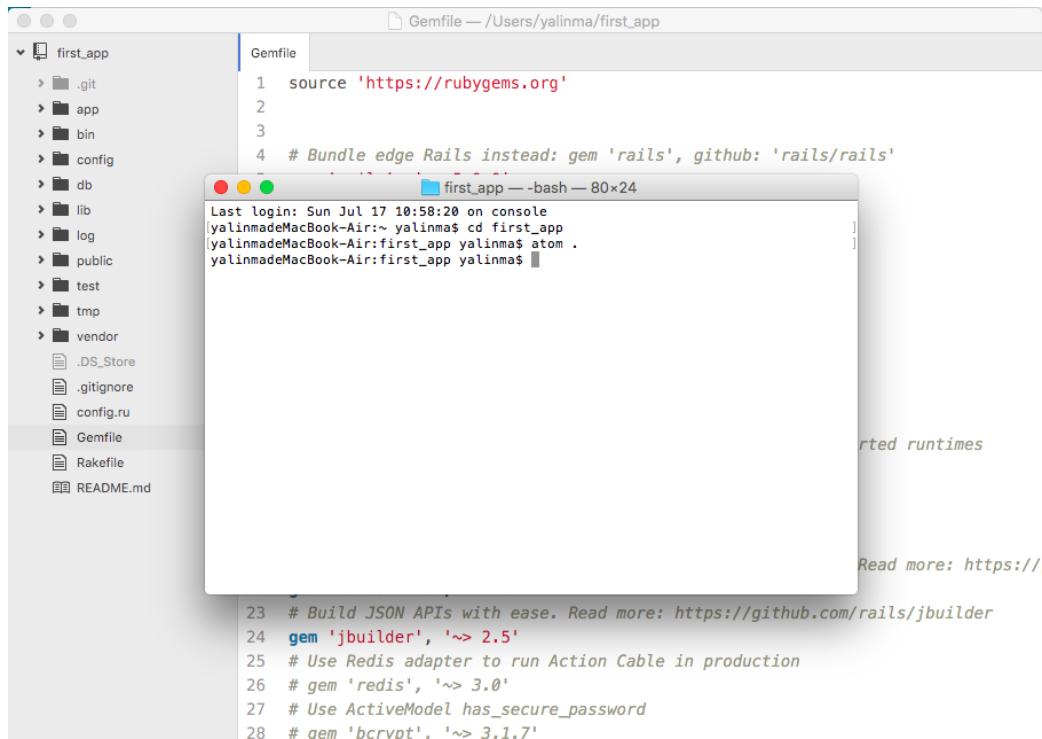
```
first_app — bash — 80x24
v2ProLtEiqpAsaaPjSRUEMNztN4n61lkG214/vTvmnntPyQyZlWKcC/63b+RiMDvCtGzyWfkQUVC8SBe
mr5ftxgD6r4daB0xzaNF3E1sCpebw9ZX/G4nrXCaCrvUMh3CrZ6DJPe5WQHTnXbj22dklw02vmFB2dej
Fi/ArxQM8Jx02mWorem1VDDl052XH6hF8YvQZtMvMLNz26gkoQMlcqXsY4h yalinma@yalinmadeMa
cBook-Air.local
yalinmadeMacBook-Air:first_app yalinma$ git remote add origin git@github.com:may
alin/first_app.git
[yalinmadeMacBook-Air:first_app yalinma$ git push -u origin master
Counting objects: 109, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (95/95), done.
Writing objects: 100% (109/109), 24.78 KiB | 0 bytes/s, done.
Total 109 (delta 2), reused 0 (delta 0)
To git@github.com:mayalin/first_app.git
 * [new branch] master -> master
Branch master set up to track remote branch master from origin.
[yalinmadeMacBook-Air:first_app yalinma$ heroku login
Enter your Heroku credentials.
Email: mayalin2012@gmail.com
>Password (typing will be hidden):
Logged in as mayalin2012@gmail.com
[yalinmadeMacBook-Air:first_app yalinma$ heroku keys:add
? Which SSH key would you like to upload? /Users/yalinma/.ssh/id_rsa.pub
Uploading /Users/yalinma/.ssh/id_rsa.pub SSH key... done
yalinmadeMacBook-Air:first_app yalinma$ ]
```

Step1: Heroku app (首次)

在终端输入 heroku create，创建 heroku app。一般情况下，每个项目对应的 heroku app 创建一个即可，后续只需要提交修改更新即可。

Step2: 修改 Gemfile

在 first_app 终端输入 atom . 打开项目文件库。这一步生效需要你首先已经安装了 [Atom](#) 编辑器。



The screenshot shows the Atom code editor with the Gemfile open. The file contains the following code:

```
source 'https://rubygems.org'
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder
gem 'jbuilder', '~> 2.5'
# Use Redis adapter to run Action Cable in production
# gem 'redis', '~> 3.0'
# Use ActiveModel has_secure_password
# gem 'bcrypt', '~> 3.1.7'

group :development, :test do
  # Call 'byebug' anywhere in the code to stop execution and inspect your application's state
  gem 'byebug', platform: :mri
  gem 'sqlite3'
end
```

A terminal window is also visible, showing the command `atom .` being run in the `first_app` directory.

由于 Heroku 使用 PostgreSQL 而不是 SQLite，我们需要修改 Gemfile 才能使数据库在 Heroku 上运行。在 Gemfile 里，剪切第 7 行的 gem 'sqlite3'。

```
4 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
5 gem 'rails', '~> 5.0.0'
6 # Use sqlite3 as the database for Active Record
7 gem 'sqlite3'
8 # Use Puma as the app server
9 gem 'puma', '~> 3.0'
10 # Use SCSS for stylesheets
```

把它放在 30 行左右的，development group 里面。

```
33 group :development, :test do
34   # Call 'byebug' anywhere in the code to stop execution and inspect your application's state
35   gem 'byebug', platform: :mri
36   gem 'sqlite3'
37 end
```

在 47 行新增一个 production group, 加上 pg 这个 gem。

```
47 group :production do
48   gem "pg"
49 end
```

在终端输入 bundle install, 安装修改后的 Gemfile。以后每当你修改 Gemfile 时, 你都要接着跑 bundle install, 这样修改才会真的生效。

```
[yalinmadeMacBook-Air:first_app yalinma$ bundle install
Fetching gem metadata from https://rubygems.org/
Fetching version metadata from https://rubygems.org/
Fetching dependency metadata from https://rubygems.org/
Resolving dependencies.....]
Using rake 11.2.2
Using concurrent-ruby 1.0.2
Using i18n 0.7.0
Using minitest 5.9.0
```

Step3: 提交修改到本地仓库

因为 heroku 只接受我们提交到本地仓库的程序, 所以每次修改文件都要提交。在终端输入 git add . 和 git commit -m "描述你修改了什么" 将 first_app 中修改过的文件提交到本地 git 仓库。

```
[yalinmadeMacBook-Air:first_app yalinma$ git add .
[yalinmadeMacBook-Air:first_app yalinma$ git commit -m "move sqlite3 to development group & add pg gem to production group"]
```

Step4: 上传 Heroku

在终端输入 git push heroku master, 把我们提交到本地仓库的程序, 上传到 heroku。

```
[yalinmadeMacBook-Air:first_app yalinma$ git push heroku master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 283 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Using set buildpack heroku/ruby
remote: -----> Ruby app detected
remote: -----> Compiling Ruby/Rails
remote: -----> Using Ruby version: ruby-2.2.4
remote: -----> Installing dependencies using bundler 1.11.2
remote:           Running: bundle install --without development:test --path vendor/bundle --binstubs vendor/bundle/bin -j4 --deployment
[
```

```

remote: ##### WARNING:
remote:      No Procfile detected, using the default web server.
remote:      We recommend explicitly declaring how to boot your server process
remote: via a Procfile.
remote:      https://devcenter.heroku.com/articles/ruby-default-web-server
remote:
remote:
remote: -----> Discovering process types
remote:      Procfile declares types      -> (none)
remote:      Default types for buildpack -> console, rake, web, worker
remote:
remote: -----> Compressing...
remote:      Done: 27.7M
remote: -----> Launching...
remote:      Released v7
remote:      https://safe-dusk-58912.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/safe-dusk-58912.git
  03080dd..cd04d53  master -> master

```

Step5: 迁移数据库

Heroku 的数据库和你本地的数据库是分开的，也就是说每次你更新了本地的数据库，你都要对应的更新 heroku 的数据库。在终端输入 heroku run rake db:migrate，把本地数据库迁移到 heroku 的数据库。

```

[yalinmadeMacBook-Air:first_app yalinma$ heroku run rake db:migrate
Running rake db:migrate on ⚡ safe-dusk-58912... up, run.3925
D, [2016-07-17T05:02:05.663396 #3] DEBUG -- :      (1.7ms)  SELECT pg_try_advisory_lock(3617015510247687660);
D, [2016-07-17T05:02:05.673603 #3] DEBUG -- :      ActiveRecord::SchemaMigration Load (1.9ms)  SELECT "schema_migrations".* FROM "schema_migrations"
D, [2016-07-17T05:02:05.694322 #3] DEBUG -- :      ActiveRecord::InternalMetadata Load (1.9ms)  SELECT "ar_internal_metadata".* FROM "ar_internal_metadata" WHERE "ar_internal_metadata"."key" = $1 LIMIT $2  [{"key": :environment}, {"LIMIT": 1}]
D, [2016-07-17T05:02:05.700981 #3] DEBUG -- :      (1.6ms)  BEGIN
D, [2016-07-17T05:02:05.703921 #3] DEBUG -- :      (1.6ms)  COMMIT
D, [2016-07-17T05:02:05.705761 #3] DEBUG -- :      (1.7ms)  SELECT pg_advisory_unlock(3617015510247687660)

```

Step6: 打开网页

在终端输入 heroku open，在浏览器打开你上传到 heroku 的程序。你也可以[点击链接](#)，查看我上传的一个项目。

```
[yalinmadeMacBook-Air:first_app yalinma$ heroku open
yalinmadeMacBook-Air:first_app yalinma$ ]
```



总结

一般在实际项目是中 Step3-6 是不停重复进行的，要实现的核心就是：修改程序 > 提交到本地仓库 > 上传 Heroku，反复进行修改上传。

参考

1. 利用 Heroku 将你的程序部署到网络上

July 17, 2016

[Rails Git Github](#)

[上传 Github](#)

命令行

1. git init
2. git add .
3. git commit -m "add all files"
4. git remote add origin git@github.com:mayalin/first_app.git
5. git push -u origin master

Step0: Github 用户名和邮箱 (首次)

首次使用 Git 命令上传 Github 时，需要输入你的用户名和邮箱，后续再上传就无需执行这 2 条命令了。

git config --global user.name "mayalin"

git config --global user.email mayalin2012@gmail

```
invoke test_unit
create     test/controllers/topics_controller_test.rb
invoke helper
create     app/helpers/topics_helper.rb
invoke test_unit
invoke jbuilder
create     app/views/topics/index.json.jbuilder
create     app/views/topics/show.json.jbuilder
invoke assets
invoke coffee
create     app/assets/javascripts/topics.coffee
invoke scss
create     app/assets/stylesheets/topics.scss
invoke scss
create     app/assets/stylesheets/scaffolds.scss
[yalinmadeMacBook-Air:first_app yalinma$ rake db:migrate
== 20160713023819 CreateTopics: migrating =====
-- create_table(:topics)
 -> 0.0014s
== 20160713023819 CreateTopics: migrated (0.0015s) =====
[yalinmadeMacBook-Air:first_app yalinma$ git config --global user.name "mayalin" ]
[yalinmadeMacBook-Air:first_app yalinma$ git config --global user.email mayalin20
12@gmail.com ]
```

Step1: 生成本地.git 文件

在 first_app 文件夹下，执行 git init，生成 first_app 的本地.git 文件。

```

first_app — bash — 80x24
create      app/helpers/topics_helper.rb
invoke      test_unit
invoke      jbuilder
create      app/views/topics/index.json.jbuilder
create      app/views/topics/show.json.jbuilder
invoke      assets
invoke      coffee
create      app/assets/javascripts/topics.coffee
invoke      scss
create      app/assets/stylesheets/topics.scss
invoke      scss
create      app/assets/stylesheets/scaffolds.scss
[yalinmadeMacBook-Air:first_app yalinma$ rake db:migrate
== 20160713023819 CreateTopics: migrating =====
-- create_table(:topics)
-> 0.0014s
== 20160713023819 CreateTopics: migrated (0.0015s) =====

[yalinmadeMacBook-Air:first_app yalinma$ git config --global user.name "mayalin" ]
[yalinmadeMacBook-Air:first_app yalinma$ git config --global user.email mayalin20
12@gmail.com
[yalinmadeMacBook-Air:first_app yalinma$ git init
Initialized empty Git repository in /Users/yalinma/first_app/.git/
yalinmadeMacBook-Air:first_app yalinma$ ]

```

如果有什么异常，你可以在终端中输入 rm -rf .git/，来删除这个.git 文件，再重新生成。

Step2: 提交文件到本地仓库

上一步生成的.git 文件，就是 first_app 的本地 git 仓库。使用命令 git add . 和 git commit -m "add all files" 将 first_app 下所有的文件提交到本地 git 仓库。

```
[yalinmadeMacBook-Air:first_app yalinma$ git add .
[yalinmadeMacBook-Air:first_app yalinma$ git commit -m "add all files"
]
```

Step3: 上传 Github

登入你的 [Github](#) 账号，创建一个名为 first_app 的项目。

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner	Repository name
 mayalin	/ first_app 

Great repository names are short and memorable. Need inspiration? How about [crispy-broccoli](#).

Description (optional)

 Public
Anyone can see this repository. You choose who can commit.

 Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None



Create repository

在终端输入新项目提供给你的 2 条上传命令：

```
git remote add origin git@github.com:mayalin/first_app.git  
git push -u origin master
```

The screenshot shows the GitHub repository setup page for 'mayalin / first_app'. It includes sections for quick setup, command-line instructions for creating and pushing repositories, and options for importing code from other repos.

```
Quick setup — if you've done this kind of thing before  
Set up in Desktop or HTTPS SSH git@github.com:mayalin/first_app.git  
We recommend every repository include a README, LICENSE, and .gitignore.  
...or create a new repository on the command line  
echo "# first_app" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin git@github.com:mayalin/first_app.git  
git push -u origin master  
...or push an existing repository from the command line  
git remote add origin git@github.com:mayalin/first_app.git  
git push -u origin master  
...or import code from another repository  
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.  
Import code
```

上传成功后画面，你也可以[点击链接](#)查看我刚上传到 Github 的文件。

The screenshot shows the GitHub repository page for 'mayalin / first_app'. It displays the commit history, showing 2 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was made 16 minutes ago.

File	Message	Time
app	add rest files	16 minutes ago
bin	add rest files	16 minutes ago
config	add rest files	16 minutes ago
db	add rest files	16 minutes ago
lib	add rest files	16 minutes ago
log	add rest files	16 minutes ago
public	add rest files	16 minutes ago
test	add rest files	16 minutes ago
tmp	add rest files	16 minutes ago
vendor/assets	add rest files	16 minutes ago
.gitignore	add rest files	16 minutes ago

July 17, 2016

[Rails](#) [Git](#) [Github](#)

留言板功能

新增留言板功能

接上两篇，保持 rails s 运行页面打开状态，这样才能正常看到 <http://localhost:3000/> 页面。

```
[yalinmadeMacBook-Air:~ yalinma$ cd first_app
[yalinmadeMacBook-Air:first_app yalinma$ rails s
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.4.0 (ruby 2.3.1-p112), codename: Owl Bowl Brawl
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop
```

新开一个终端(快捷键： + N)，专门跑指令。输入 cd first_app 在 first_app 下跑 rails g scaffold topic title:string description:text。

```
[yalinmadeMacBook-Air:~ yalinma$ cd first_app
[yalinmadeMacBook-Air:first_app yalinma$ rails g scaffold topic title:string description:text
Running via Spring preloader in process 4524
  invoke  active_record
    create    db/migrate/20160713023819_create_topics.rb
    create    app/models/topic.rb
  invoke  test_unit
    create    test/models/topic_test.rb
    create    test/fixtures/topics.yml
  invoke  resource_route
    route    resources :topics
  invoke  scaffold_controller
    create    app/controllers/topics_controller.rb
  invoke  erb
    create    app/views/topics
```

rake db:migrate

```
[yalinmadeMacBook-Air:first_app yalinma$ rake db:migrate
  create    app/views/topics/new.html.erb
  create    app/views/topics/_form.html.erb
  invoke  test_unit
    create    test/controllers/topics_controller_test.rb
  invoke  helper
    create    app/helpers/topics_helper.rb
  invoke  test_unit
  invoke  jbuilder
    create    app/views/topics/index.json.jbuilder
    create    app/views/topics/show.json.jbuilder
  invoke  assets
  invoke  coffee
    create    app/assets/javascripts/topics.coffee
  invoke  scss
    create    app/assets/stylesheets/topics.scss
  invoke  scss
    create    app/assets/stylesheets/scaffolds.scss
[yalinmadeMacBook-Air:first_app yalinma$ rake db:migrate
== 20160713023819 CreateTopics: migrating =====
-- create_table(:topics)
  -> 0.0014s
== 20160713023819 CreateTopics: migrated (0.0015s) =====
```

前往 <http://localhost:3000/topics> 即可进到留言板。注意这里 rails s 运行页面也必须保存打开状态，页面才能正常显示。



July 17, 2016

Rails 初级 全栈

Rails server

开启 server

输入 rails s, 开启 server 模式。

```
[yalinmadeMacBook-Air:first_app yalinma$ rails s
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.4.0 (ruby 2.3.1-p112), codename: Owl Bowl Brawl
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop]
```

打开 <http://localhost:3000/>, 即可进入 first_app 首页, 也即默认的 rails 欢迎页。



关闭 server

按 Ctrl-C 关闭 server, 提示^CExiting。如果不手动关停, sever 会一直运行。

```
[yalinmadeMacBook-Air:first_app yalinma$ rails s
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.4.0 (ruby 2.3.1-p112), codename: Owl Bowl Brawl
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop
^CExiting
yalinmadeMacBook-Air:first_app yalinma$ ]
```

删除 server

1. 查找 lsof -i tcp:3000
2. 删除 kill -9 <PID>

当页面 <http://localhost:3000/> 打开空白或提示已有 server 在运行。

```
[yalinmadeMacBook-Air:first_app yalinma$ rails s
=> Booting Puma
=> Rails 5.0.0 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
A server is already running. Check /Users/yalinma/first_app/tmp/pids/server.pid.
Exiting
yalinmadeMacBook-Air:first_app yalinma$ ]
```

输入 lsof -i tcp:3000 查找已有 server。

```
[yalinmadeMacBook-Air:first_app yalinma$ lsof -i tcp:3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
ruby 4116 yalinma 31u IPv4 0xda16aa36a0e775b3 0t0 TCP localhost:hbc
(Listen)
yalinmadeMacBook-Air:first_app yalinma$ ]
```

运行 kill -9 <PID> PID 换成你找到的号码，就可以删掉该 server。

```
[yalinmadeMacBook-Air:first_app yalinma$ lsof -i tcp:3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
ruby 4116 yalinma 31u IPv4 0xda16aa36a0e775b3 0t0 TCP localhost:hbc
(Listen)
yalinmadeMacBook-Air:first_app yalinma$ kill -9 4116
yalinmadeMacBook-Air:first_app yalinma$ ]
```

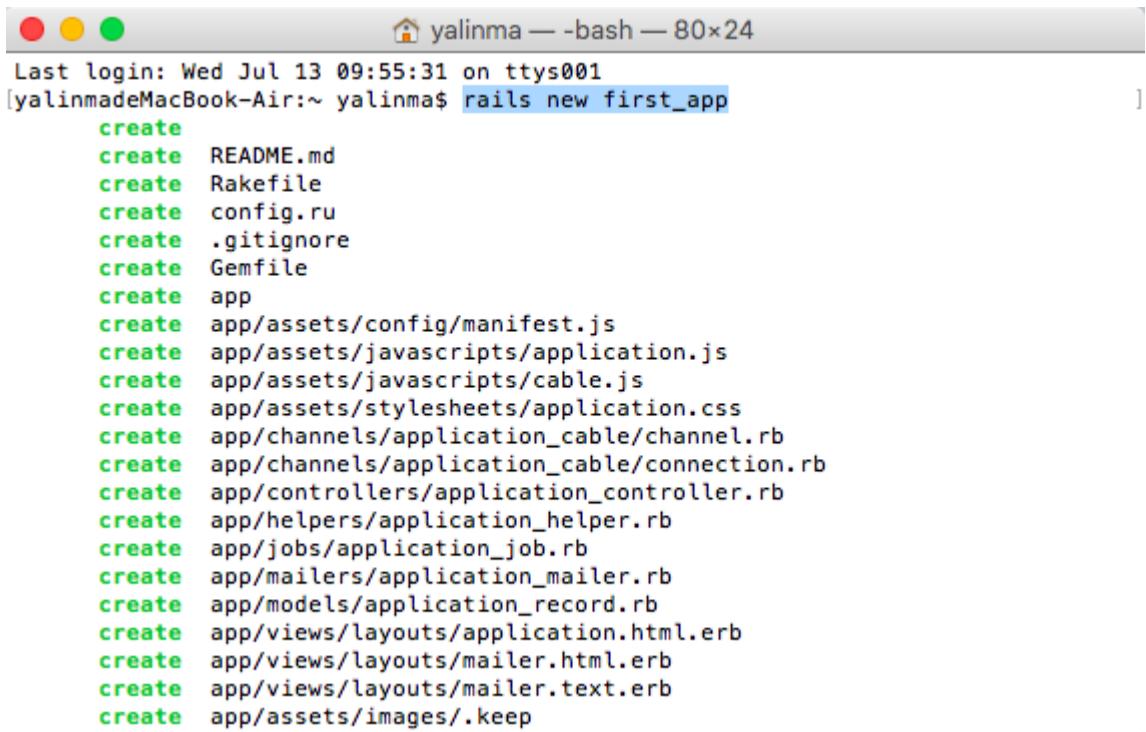
July 16, 2016

[Rails 初级 全栈](#)

[Rails app 创建](#)

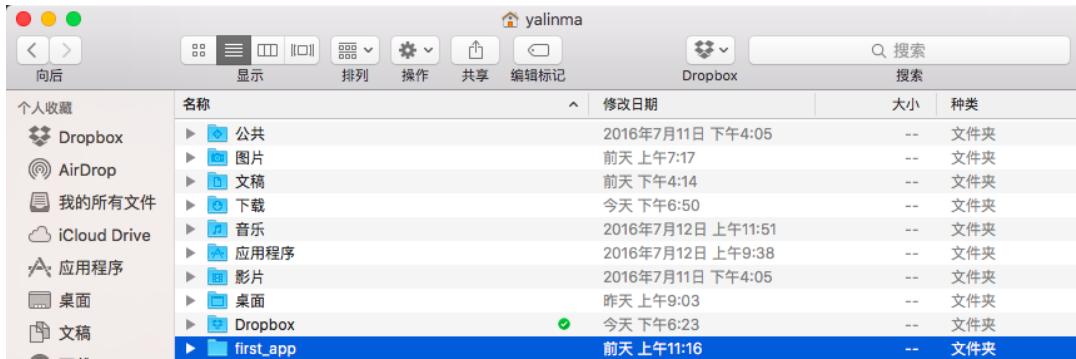
Rails app 创建

在终端输入 rails new first_app，建立新的 rails 项目。



```
Last login: Wed Jul 13 09:55:31 on ttys001
[yalinmadeMacBook-Air:~ yalinma$ rails new first_app
  create
  create README.md
  create Rakefile
  create config.ru
  create .gitignore
  create Gemfile
  create app
  create app/assets/config/manifest.js
  create app/assets/javascripts/application.js
  create app/assets/javascripts/cable.js
  create app/assets/stylesheets/application.css
  create app/channels/application_cable/channel.rb
  create app/channels/application_cable/connection.rb
  create app/controllers/application_controller.rb
  create app/helpers/application_helper.rb
  create app/jobs/application_job.rb
  create app/mailers/application_mailer.rb
  create app/models/application_record.rb
  create app/views/layouts/application.html.erb
  create app/views/layouts/mailер.html.erb
  create app/views/layouts/mailер.text.erb
  create app/assets/images/.keep
```

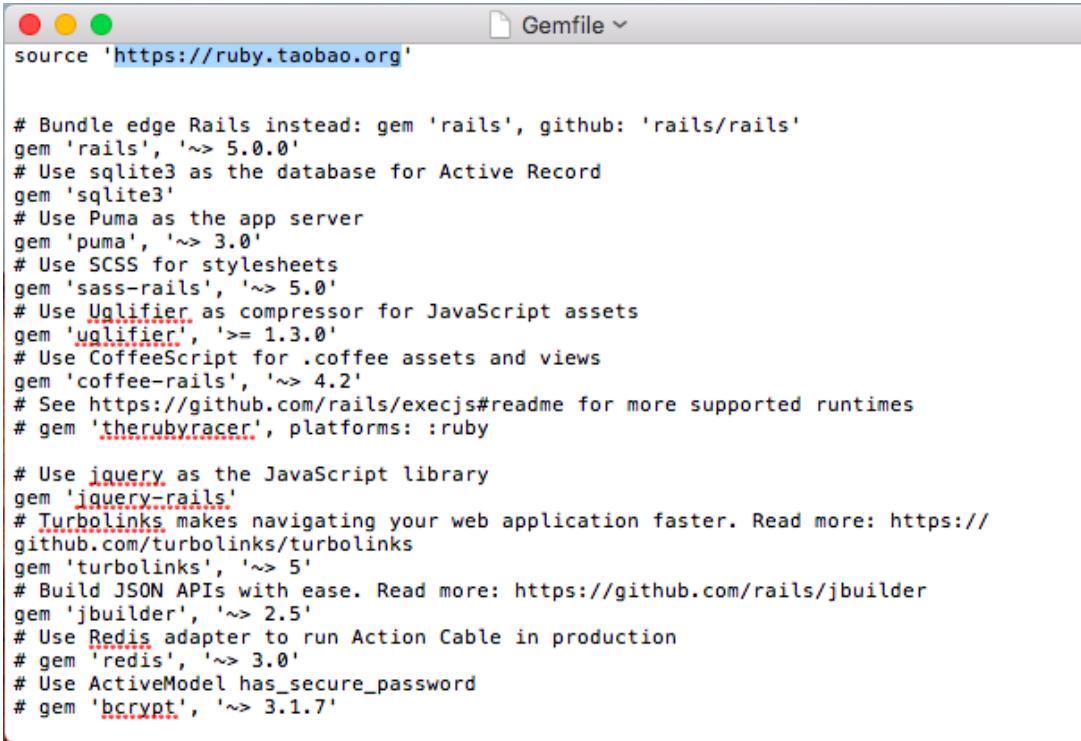
刚用命令行创建的 app，路径：Finder > 个人 > first_app。



相关：RubyGems 安装

一般在配置开发环境时，已经挂了 VPN，保持来源为官方 <https://rubygems.org> 即可，终端会自动安装官方来源的 rubygems，你不需要做什么操作。有些教程会建议中国大陆用户把来源直接改为中国大陆镜像 <https://ruby.taobao.org>，但是当你已经挂了 VPN 的情况下，再使用国内镜像反而会出现异常。

中国大陆用户，未使用 VPN 的情况下，需要(1)打开 first_app 文件夹下的 Gemfile，将来源从官方 <https://rubygems.org> 改为国内淘宝镜像 <https://ruby.taobao.org>。

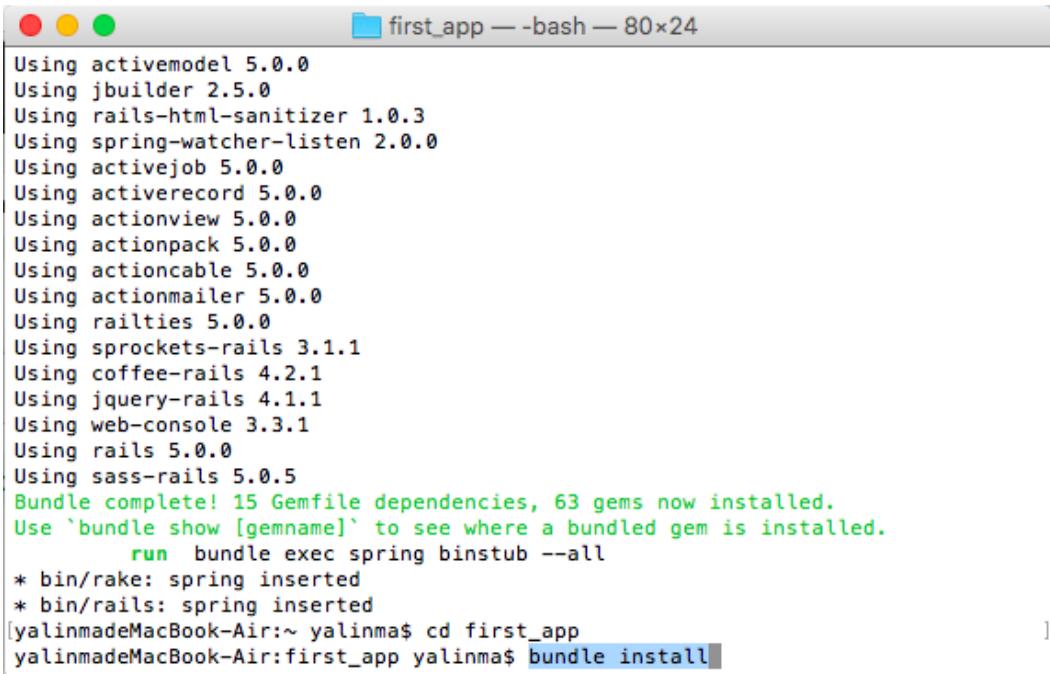


```
source 'https://ruby.taobao.org'

# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '~> 5.0.0'
# Use sqlite3 as the database for Active Record
gem 'sqlite3'
# Use Puma as the app server
gem 'puma', '~> 3.0'
# Use SCSS for stylesheets
gem 'sass-rails', '~> 5.0'
# Use Uglifier as compressor for JavaScript assets
gem 'uglifier', '>= 1.3.0'
# Use CoffeeScript for .coffee assets and views
gem 'coffee-rails', '~> 4.2'
# See https://github.com/rails/execjs#readme for more supported runtimes
# gem 'therubyracer', platforms: :ruby

# Use jquery as the JavaScript library
gem 'jquery-rails'
# Turbolinks makes navigating your web application faster. Read more: https://github.com/turbolinks/turbolinks
gem 'turbolinks', '~> 5'
# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder
gem 'jbuilder', '~> 2.5'
# Use Redis adapter to run Action Cable in production
# gem 'redis', '~> 3.0'
# Use ActiveModel has_secure_password
# gem 'bcrypt', '~> 3.1.7'
```

(2)接着在终端输入 cd first_app, 进入 first_app 后 (3)输入 bundle install, 用国内镜像重新安装 rubygems。



```
first_app — bash — 80x24

Using activemodel 5.0.0
Using jbuilder 2.5.0
Using rails-html-sanitizer 1.0.3
Using spring-watcher-listen 2.0.0
Using activejob 5.0.0
Using activerecord 5.0.0
Using actionview 5.0.0
Using actionpack 5.0.0
Using actioncable 5.0.0
Using actionmailer 5.0.0
Using railties 5.0.0
Using sprockets-rails 3.1.1
Using coffee-rails 4.2.1
Using jquery-rails 4.1.1
Using web-console 3.3.1
Using rails 5.0.0
Using sass-rails 5.0.5
Bundle complete! 15 Gemfile dependencies, 63 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
  run  bundle exec spring binstub --all
* bin/rake: spring inserted
* bin/rails: spring inserted
[yalinmadeMacBook-Air:~ yalinma$ cd first_app
[yalinmadeMacBook-Air:first_app yalinma$ bundle install]
```

参考

1. [RubyGems 镜像- 淘宝网](#)