全栈工程师成长记

编程重新定义人生

- Blog
- Archives
- About Me

主厨精选 9/23 日记

SEPTEMBER 24, 2016
Objective
完成：RubyConf China 2016 Day1
Reflective
情绪：
低点：没有
Interpretive
学到：

1. i18n 是 internationalization（国际化） 的约定缩写
2. 代码中不能含有中文，代码的国际化就是无需通过修改代码便可以适应不同国家和地区的需求
3. 使用 yml 文件作为词典配置文件
4. 互联网产品标配中英文语言是主流
5. 本地化的益处：改善用户体验、提高付费转化率、提高访客注册转化率、扩大市场占有
6. 如果不能做好本地化，宁可不要进入一个市场
7. gettext 可以把自然语言变成参数来传递（提取 msgid ）
8. gettext 负责提取 msgid, fast_gettext 负责翻译(缓存占用仅为 gettext 的 1/500)
9. 用 onesky 进行翻译协作
10. 对翻译人员的要求：以目标语言为母语、对互联网产品有所了解、了解产品中的特定翻译点
11. 自动计算工资系统的目的是帮助 HR 处理 90% 重复工作, 使其将精力留在“人”的部分
12. 人事行政不分家
13. 把签到记录从线下做到线上，每天凌晨用自动脚本批量抽取数据进行计算
14. 针对津补贴背后复杂的需求，使其可以添加不同的扣发规则（触发选项）
15. 一切皆可重算：原始数据 + 规则 = 核算结果
16. 原始数据不要包括结果计算，模型不要去计算，原始数据模型也不要去动
17. 以资源的视角进行开发（比如把注册表单当成资源来看，new 一个 registration）
18. 《Growing Rails Applications in Practice》
19. 创建更多的业务的所有的 action 都用默认的 7 个标准方法，然后做模版
20. 通过加密某些关键字段（如员工姓名、部门、职位），使其他字段的数据变得无意义
21. 所有的业务逻辑都在 routes.rb 里面
22. 真实世界里，审批通过之前不会创建一个新资源

23. 面向资源编程需要过的一个坎是，create、delete 可能不是真正的对数据进行操作
24. 大多数开发者都没有架构的能力
25. 微服务的好处：易于理解和维护，每一个子服务都不过分复杂；独立开发和测试，可选不同技术栈，易于扩展
26. MVC + ServiceObject，抽象业务层
27. 康威定律：组织形式等同系统设计
28. TGID （Thread group id）是真正意义上的进程 ID，是 get_pid 的结果(多个线程合并成一个线程组)
29. 解决内存泄漏问题的四个步骤： Survive，保证服务可用。企业级应用需要企业级的稳定，可以慢但不能崩。 Address，定位问题。到底是内存暴涨（某个时间段特别高）还是内存泄漏（平滑的曲线）？ Fix，解决问题。继续追踪，本例中发现是 Redis 问题，升级（或降级）版本号解决…… Lesson，总结经验。出了问题不能猜。
30. 同一个问题的会有不同解决思路（到最后各自的代码风格差异会变得很大），有时很难说哪个是最好的，但面对现实的业务状况，折中的选择一个也许是最快的途径
31. Rails API 和 Grape 应该根据不同的应用场景做选择
32. Basic Auth 等同于裸奔，Basic Auth + HTTPS 等同于在黑夜中裸奔，不要把所有的安全机制都寄托在 HTTPS 上。
33. 安全原则：1.不要乱用算法；2.不要乱改算法。
34. 使用 rails stats 查看项目统计信息，包括 class、method、代码行数统计等
35. 在 Atom 中使用 control + command + 上下键 可以将选中的代码进行移动上下移动
36. 在 Atom 中使用 command + R 可以快速定位并打开文件
37. 在 Atom 中使用 command + D 可以快速定位当前 controller 里面的某个 action
38. 使用 mkdir -p xxx/yyy/zzz 可以直接创建多重嵌套的文件夹结构（比如有一个 spec 文件夹，想创建一个 spec/support/page_objects/devise/session 这样的路径，使用 mkdir -p 参数就可以一步到位而不需逐层建立）
39. 产品的迭代是从自行车到摩托车到汽车的过车，而不是从汽车轮到汽车底盘到汽车的过程
40. 部署之前先把代码中的 binding.pry 去掉或者注释掉，否则……还能怎样，当然是会报错啦

领悟：

RubyConf China 2016 今天在成都举办第八届年会，虽然作为一个新手没能听懂太多大会主题演讲的技术细节，但作为刚加入 Ruby 社区的新成员，能够近距离的接触各路 Ruby 大神还是有点小激动。当然一天 8 个主题演讲听完，最大的感受是收获颇丰——很多的理念、方法论和技术实现是第一次了解到或者是之前听过但理解得不够多、不够深入，比如“主厨精选”、“约定大于配置”、“面向资源编程”、Turbolinks、gettext 等等。

当然来到成都，“吃吃吃”是一个躲不开的话题。

早上在会场附近的西月城谭豆花吃早餐，点了一份面条，服务员问我还要点哪一款豆花，我说不用了，然后她的表情像是在说："来我们店你不吃豆花？这可是我们的招牌……"。

RubyConf China 给大家准备了主会场二楼的自助餐，结果我硬是把价值 120 多元的自助餐吃成了 10 几块的快餐，这也是我平时不喜欢吃自助餐的原因：看上去有很多，但想吃的没几个。

Decisional

形容：过得非常充实的一天

努力：过更充实的一天

强迫症也有春天 9/20 日记

SEPTEMBER 20, 2016
Objective
完成：神秘的 RSpec 测试第二轮 ch1
Reflective
情绪：表示跟今天北京的天气一样晴朗
低点：没有
Interpretive
学到：
1.在 Atom 里想要要对一个文件的缩进排版优化，只要在该文件上点右键选“Beautify File”就可以了
（前提是先在 Atom 的 Packages 里安装 atom－beautify 插件）
2.rspec－rails：

- **describe**：用于定义一组事件 比如 describe "GET index" do 可以定义具体测试 index action 的多个行为
- **it**：用于定义一个具体的 example，比如 index action 的测试里可以包括 it "assigns @courses and render" do、it "render template" do
- **context**：用于描述内容，比如 context "when course doesn't have a title " do
- **panding**：用于存放尚未想好要如何测试的一个方法（先占坑）

领悟：
手打跟复制粘贴的效率差距果真不是一个量级的，原本计划今天做 rspec 测试练习第二遍，结果因为是一个字一个字的手打，一天下来只做了一个章节……
不知道北京从什么时候开始入秋，反正最近几天的最低气温已经越来越低。不过秋天来了，春天也就不远了，这不，今天我就发现自己的强迫症迎来了春天。
在做 rspec 测试的时候，如果测试通过，命令行会直接提示：
.....

Finished in 0.26345 seconds (files took 1.74 seconds to load)
20 examples, 0 failures
类似的结果，只有 3 行文字，非常的清晰明了。
后来在写一个 show action 的测试时，虽然测试结果是通过了，但同时却会出现一大堆的 DEPRECATION WARNING 提示：
DEPRECATION WARNING: ActionController::TestCase HTTP request methods will accept only keyword arguments in future Rails versions.

Examples:

get :show, params: { id: 1 }, session: { user_id: 1 }
process :update, method: :post, params: { id: 1 }
(called from block (3 levels) in <top (required)> at /Users/nfreeness/Project/classrom/spec/controllers/courses_controller_spec.rb:26)
DEPRECATION WARNING: ActionController::TestCase HTTP request methods will accept only

keyword arguments in future Rails versions.
只要有一行相关的错误代码，那个 Examples 的提示就会出现两次，随着类似代码的增加，每运行一次测试，命令行的 log 就会因为这些提示而疯狂的滚动，看着非常碍眼（平时 log 一多我都要直接 command + k 清除），于是决心把它摘掉。
原本的测试写法是：
get :show, :id => course.id
这种情况它就会提示写法有问题。
改成：
get :show, params: { id: course.id }
就 ok 了。而且之后再写其他 action 也会小心留意这个用法，从此告别这些烦人的 DEPRECATION WARNING 提示。

- 我在另外一篇博客《rspec-rails 常见错误提示及解决办法》里有写更多的 params 修改方法参考。
- 附上 rails github 上的 Pull requests 描述。

As @dhh suggested, controller HTTP methods are a prime candidate for kwargs.
post url, nil, nil, { a: 'b' } doesn't make sense.
More explicit way would be: post url, params: { y: x }, session: { a: 'b' }.
Decisional
形容：通过今天的测试练习再次印证了拼图学习和肌肉记忆的强大威力。
努力：完成神秘的 RSpec 测试第二轮？

rspec-rails 常见错误提示及解决办法

SEPTEMBER 20, 2016
由于使用 TDD 开发，有些报错是尚未做到某个步骤必然发生的结果，而有些则是本身编写的代码或者测试代码有错。本文将这些常见的错误提示都记录并提供相应的解决办法（有特定内容——比如某个 class 的名称——的位置请自行替换）。
NameError
错误提示：

```
 Failure/Error: course1 = Course.create(title: "foo", description: "bar")

       NameError:
         uninitialized constant Course
```

原因及解法：
没有 course 这个 class，因为还没有建立这个 model
运行 rails g model course title:string description:text
运行 rake db:migrate

ActionController::UrlGenerationError
错误提示：

```
 Failure/Error: get :index

       ActionController::UrlGenerationError:
         No route matches {:action=>"index", :controller=>"courses"}
```

原因及解法：

尚未建立 routing
修改 routes.rb
routes.rb
```
  Rails.application.routes.draw do
+       resources :courses
  end
```

AbstractController::ActionNotFound
错误提示：
```
Failure/Error: get :index

       AbstractController::ActionNotFound:
         The action 'index' could not be found for CoursesController
```
原因及解法：
尚未建立 index action
修改 courses_controller.rb
courses_controller.rb
```
class CoursesController < ApplicationController

+          def index
+             @courses = Course.all
+          end

end
```

ActionController::UnknownFormat
错误提示：
```
Failure/Error: get :index

       ActionController::UnknownFormat:
         CoursesController#index is missing a template for this request format and variant.


         request.formats: ["text/html"]
         request.variant: []
```
原因及解法：
尚未建立 index view
新增 app/views/course/index.html.erb 并填入页面内容即可

NoMethodError
错误提示：
```
 Failure/Error: expect(assigns[:courses]).to eq([course1,course2])

       NoMethodError:
```

assigns has been extracted to a gem. To continue using it,
add `gem 'rails-controller-testing'` to your Gemfile.

原因及解法：

没有安装 rails-controller-testing 这个 gem

修改 gemfile

gemfile

```
group :development, :test do
  gem 'byebug', platform: :mri
  gem 'rspec-rails'
+       gem 'rails-controller-testing'
end
```

---

got: nil

错误提示：

Failure/Error: expect(assigns[:courses]).to eq([course1,course2])

expected: [#<Course id: 1, title: "foo", description: "bar", created_at: "2016-09-20 04:35:06", updated_at: "20...e: "bar", description: "foo", created_at: "2016-09-20 04:35:06", updated_at: "2016-09-20 04:35:06">]

got: nil

(compared using ==)

原因及解法：

controller 里面的变量命名错误（手动拼写的坑）

修改 courses_controller.rb

courses_controller.rb

```
  def index
-     @couses = Course.all
+     @courses = Course.all
  end
```

---

spec/support/page_objects/base (LoadError)

错误提示：

rspec spec/features/homepage_spec.rb

spec/support/page_objects/pages/home.rb:1:in `require_relative': cannot load such file spec/support/page_objects/base (LoadError)

from /Users/nfreeness/Project/classrom/spec/support/page_objects/pages/home.rb:1:in `<top (required)>'

原因及解法：

尚未建立 base.rb

新增 spec/support/page_objects/base.rb

```
module PageObjects
  class Base
```

```
    include Capybara::DSL
    include Rails.application.routes.url_helpers
  end
end
```

`id` is not available from within an example

错误提示：

```
 Failure/Error: get :show, id => course.id
          `id` is not available from within an example (e.g. an `it` block) or from constructs that run in the scope of an example (e.g. `before`, `let`, etc).
          It is only available on an example group (e.g. a `describe` or `context` block).
```

原因及解法：

测试代码拼写错误

修改 spec/support/page_objects/base.rb

```
-       get :show, id => course.id
+       get :show, :id => course.id
```

NameError

错误提示：

```
 Failure/Error: get :show, id => course.id
       Failure/Error: expect(response).to redirect_to coures_path

       NameError:
         undefined local variable or method `coures_path' for #<RSpec::ExampleGroups::CoursesController::GETCreate:0x007f9adf8b0200>
```

原因及解法：

测试代码拼写错误

修改 spec/controllers/courses_controller_spec.rb

spec/controllers/courses_controller_spec.rb

```
-       expect(response).to redirect_to coures_path
+       expect(response).to redirect_to courses_path
```

ActionController::UrlGenerationError

错误提示：

```
Failure/Error: post :careate, course: {:description => "bar"}

       ActionController::UrlGenerationError:
         No route matches {:action=>"careate", :controller=>"courses", :course=>{:description=>"bar"}}
```

原因及解法：

controllers/courses_controller.rb 里的 create action 并没错

是 spec/controllers/courses_controller_spec.rb 里拼写错误

PS：从上面开始纪录了这么多个错误，到这里的时候已经能很快的定位到问题所在了。所以纪录错误真的太重要了。

spec/controllers/courses_controller_spec.rb

```
-         post :careate, course: {:description => "bar"}
+         post :create, course: {:description => "bar"}
```

DEPRECATION WARNING

错误提示：

spec/controllers/courses_controller_spec.rb

DEPRECATION WARNING: ActionController::TestCase HTTP request methods will accept only keyword arguments in future Rails versions.

原因及解法：

这个算不上是错误，而是提示。由于使用了旧版的语法结构，每次运行测试都会出一大堆的版本兼容提示，很是烦人。rials github 上有相关说明。

具体的修改是用 params 将 http 动作后面的方法包起来，举例如下：

```
-    get :show, :id => course.id
+    get :show, params: { id: course.id }

-    expect { post :create, course: { :description => "bar"}}.to change{Course.count}.by(0)
+    expect { post :create, params: { course: {description: "bar"}}}.to change{Course.count}.by(0)

-    post :create, course: { :description => "bar" }
+    post :create, params: { course: {description: "bar"}}

-          expect{ post :create, :course => FactoryGirl.attributes_for(:course)}.to change{ Course.count}.by(1)
+      expect{ post :create, params: { course: FactoryGirl.attributes_for(:course)}}  .to change{ Course.count}.by(1)

-    post :create, :course => FactoryGirl.attributes_for(:course)
+    post :create, params: { course: FactoryGirl.attributes_for(:course)}
```

ArgumentError

错误提示：

```
 Failure/Error: @course.update

       ArgumentError:
         wrong number of arguments (given 0, expected 1)
```

原因及解法：

update action 没写对，少了 course_params 导致无法修改文件，所以期待为 1 但输出为 0

controllers/courses_controller.rb

```
  def update
```

```
      @course = Course.find(params[:id])
-     @course.update
+      @course.update(course_params)
      redirect_to course_path(@course)
  end
```

Devise::MissingWarden

错误提示：

Failure/Error: get :new

Devise::MissingWarden:

Devise could not find the `Warden::Proxy` instance on your request environment.

Make sure that your application is loading Devise and Warden as expected and that the `Warden::Manager` middleware is present in your middleware stack.

If you are seeing this on one of your tests, ensure that your tests are either executing the Rails middleware stack or that your tests are using the `Devise::Test::ControllerHelpers` module to inject the `request.env['warden']` object for you.

原因及解法：

在 rspec 的配置中尚未包含 devise 的测试

新增 spec/support/devise.rb，将 Devise 包含在所有的 controller 测试里。

spec/support/devise.rb

```
+     RSpec.configure do |config|
+       config.include Devise::TestHelpers, type: :controller
+     end
```

值得注意的是，对扩展 gem 的支持都是在 spec/support 下面进行配置

比如我们使用 FactoryGirl 来产生假的测试资料，那我们需要新增 spec/support/factory_girl.rb

spec/support/factory_girl.rb

```
+  RSpec.configure do |config|
+    config.include FactoryGirl::Syntax::Methods
+  end
```

DEPRECATION WARNING

错误提示：

DEPRECATION WARNING: [Devise] including `Devise::TestHelpers` is deprecated and will be removed from Devise.

For controller tests, please include `Devise::Test::ControllerHelpers` instead.

(called from <top (required)> at /Users/nfreeness/Project/classroom-2/spec/controllers/courses_controller_spec.rb:3)

原因及解法：

Devise 的测试用法不规范

修改 spec/support/devise.rb

spec/support/devise.rb

```
RSpec.configure do |config|
```

```
-           config.include Devise::TestHelpers, type: :controller
+           config.include Devise::Test::ControllerHelpers, type: :controller
end
```

Trait not registered: email

错误提示：

```
 Failure/Error: user = FactoryGirl.create(:user)

       ArgumentError:
         Trait not registered: email
```

原因及解法：

尚未配置 FactoryGirl 来产生 Devise 用户 email

修改 spec/factories.rb

spec/support/devise.rb

```
  FactoryGirl.define do

+     sequence(:email) { |n| "user#{n}@example.com"}

  end
```

NoMethodError

错误提示：

```
Failures:

  1) CoursesController POST create create a course for user
     Failure/Error: expect(Course.last.user).to eq(user)

       NoMethodError:
         undefined method `user' for nil:NilClass
```

原因及解法：

尚未建立 course 和 user 的从属关系

1. 为 course 添加 user_id 栏位，add_column :courses, :user_id, :integer
2. 修改 app/models/course.rb，增加 belongs_to :user
3. 修改 app/controller/courses_controller.rb，指定 course 的 user

app/controller/courses_controller.rb

```
  def create
    @course = Course.new(course_params)
+     @course.user = current_user
```

4.修改 spec/modes/course_spec.rb，增加关系验证测试

spec/modes/course_spec.rb

```
  RSpec.describe Course, type: :model do
    it { is_expected.to validate_presence_of(:title) }
+     it { is_expected.to belong_to(:user) }
  end
```

NoMethodError

错误提示：

Failures:

1) CoursesController POST create create a course for user
Failure/Error: expect(Course.last.user).to eq(user)

NoMethodError:
undefined method `user' for nil:NilClass
./spec/controllers/courses_controller_spec.rb:102:in `block (3 levels) in <top (required)>'

原因及解法：

context "when course have a title " do 里面定义了 before { sign_in_user}，而 create a course for user 这个测试并没有包含在 context 里面，所以会找不到用户。

修改 spec/controllers/courses_controller_spec.rb

spec/controllers/courses_controller_spec.rb

```
  context "when course have a title " do
    before { sign_in_user}
    it "create a new course record" do
      ……略
    end

    it "redirect to courses_path" do
      ……略
    end
-   end

    it "create a course for user" do
      course = FactoryGirl.build(:course)
      post :create, params: { course: FactoryGirl.attributes_for(:course)}
      expect(Course.last.user).to eq(user)
    end
+ end
```

RuntimeError

错误提示：

1) CoursesController PUT update when course have title assign @course
Failure/Error: sign_in user

RuntimeError:
Could not find a valid mapping for #<Course id: 1, title: "Course Title", description: "Couese Description.", created_at: "2016-09-21 04:20:00", updated_at: "2016-09-21 04:20:00", user_id: 1>

/.rvm/gems/ruby-2.3.1/gems/devise-4.2.0/lib/devise/mapping.rb:44:in `find_scope!'

/.rvm/gems/ruby-2.3.1/gems/devise-4.2.0/lib/devise/test/controller_helpers.rb:74:in `sign_in'

./spec/support/macros.rb:2:in `sign_in_user'

./spec/controllers/courses_controller_spec.rb:148:in `block (3 levels) in <top (required)>'

原因及解法：
创建了错误的数据类型。
修改 spec/controllers/courses_controller_spec.rb
spec/controllers/courses_controller_spec.rb

```
  describe "PUT update" do

    let (:user) { FactoryGirl.create(:user)}
-   let (:user) { FactoryGirl.create(:course)}
+    let (:course) { FactoryGirl.create(:course)}
    before { sign_in_user }
```

NoMethodError
错误提示：
Failures:

```
  1) CoursesController DELETE destroy behaves like require_sign_in redirects to login page
     Failure/Error: @course = current_user.courses.find(params[:id])

     NoMethodError:
       undefined method `courses' for nil:NilClass
     Shared Example Group: "require_sign_in" called from ./spec/controllers/courses_controller_spec.rb:222
     # ./app/controllers/courses_controller.rb:40:in `destroy'
```

原因及解法：
controller 的限制尚未将 destroy action 添加进来
修改 app/controllers/courses_controller.rb
app/controllers/courses_controller.rb

```
  class CoursesController < ApplicationController

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

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

ActiveRecord::RecordNotFound

错误提示：

```
 Failures:

   1) CoursesController DELETE destroy assigns @course
        Failure/Error: @course = current_user.courses.find(params[:id])

        ActiveRecord::RecordNotFound:
          Couldn't find Course with 'id'=1 [WHERE "courses"."user_id" = ?]
```

原因及解法：

使用 FactoryGirl 创建 course 的时候没有为其指定 user。

修改 spec/controllers/courses_controller_spec.rb

spec/controllers/courses_controller_spec.rb

```
   describe "DELETE destroy" do

         let(:user) { FactoryGirl.create(:user)}
-        let!(:course) { FactoryGirl.create(:course)}
+         let!(:course) { FactoryGirl.create(:course, user: user )}
```

NoMethodError

错误提示：

```
Failures:

   1) user sign out sign_out the user
        Failure/Error: navbar.sign_in_user user.email

        NoMethodError:
          private method `sign_in_user' called for
#<PageObjects::Application::Navbar:0x007fe48deb8448 @_routes=nil>
          Did you mean?  sign_in
                         sign_out_user
        # ./spec/features/user_sign_out_spec.rb:8:in `block (2 levels) in <top (required)>'
```

原因及解法：

spec/support/page_objects/application/navbar.rb 里的 def sign_out 定义有错

修改 spec/support/page_objects/application/navbar.rb

spec/controllers/courses_controller_spec.rb

```
-         def user_sign_out(email)
+          def sign_out(email)
            user_dropdown(email).click_on "Logout"
          end
```

---

wrong number of arguments

错误提示：

Failures:

1) user create course valid
Failure/Error: course_form.create course.title course.description

ArgumentError:
wrong number of arguments (given 1, expected 0)
./spec/features/user_create_course_spec.rb:6:in `block (2 levels) in <top (required)>'

原因及解法：

在创建 course 的时候代码少了一个逗号分隔符

修改 spec/features/user_create_course_spec.rb

spec/features/user_create_course_spec.rb

```
  scenario "valid" do
    course = build_stubbed(:course)
-   course_form.create course.title course.description
+    course_form.create(course.title, course.description)
    expect(page).to have_text(course.title)
  end
```

Could not find shared examples

错误提示：

/.rvm/gems/ruby-2.3.1/gems/rspec-core-3.5.3/lib/rspec/core/example_group.rb:370:in `find_and_eval_shared': Could not find shared examples "require_sign_in" (ArgumentError)

原因及解法：

因为尚未建立该 example

新增 spec/support/shard_examples.rb

spec/support/shard_examples.rb

```
shared_examples "require_sign_in" do
  it "redirects to login page" do
    sign_out_user
    action
    expect(response).to redirect_to new_user_session_path
  end
end
```

---

got: nil

错误提示：

Failures:

1) CoursesController GET show assigns @course
Failure/Error: expect(assigns[:courses]).to eq(course)

expected: #<Course id: 1, title: "Course Title", description: "Couese Description.", created_at: "2016-09-21 12:12:29", updated_at: "2016-09-21 12:12:29", user_id: 1>

got: nil

(compared using ==)

原因及解法：

单复数使用错误

修改 spec/controllers/courses_controller_spec.rb

spec/controllers/courses_controller_spec.rb

```
  describe "GET show" do
    it "assigns @course" do
      course = FactoryGirl.create(:course)
      get :show, params: { id: course.id }
-       expect(assigns[:courses]).to eq(course)
+        expect(assigns[:course]).to eq(course)

    end
```

TypeError

错误提示：

Failures:

```
  1) CoursesController GET new when user login    assign @course
     Failure/Error: expect(assigns(:course)).to be_instance_of(course)

     TypeError:
       class or module required
     #    ./spec/controllers/courses_controller_spec.rb:46:in    `block    (4    levels)    in    <top
(required)>'
```

原因及解法：

类名定义错误，首字母应该大写

修改 spec/controllers/courses_controller_spec.rb

spec/controllers/courses_controller_spec.rb

```
  describe "GET new" do
      ……略
      it "assign @course" do
          expect(assigns(:course)).to be_new_record
-         expect(assigns(:course)).to be_instance_of(course)
+          expect(assigns(:course)).to be_instance_of(Course)
      end
```

当我不睡觉时我在干什么 9/19 日记

SEPTEMBER 19, 2016

Objective

完成：神秘的 RSpec 测试 ch1-ch3

Reflective

情绪：整体 ok

低点：早上又睡过头

Interpretive

学到：

1. 药师的一个工作职责是 check 医师开出的药方是否正确并对此承担一定的责任
2. 设计的几个主要方面：颜色、内容、形式
3. 颜色可以影响人的感受，因此品牌的视觉设计可以通过特定的颜色传递品牌理念
4. 在网站上使用图片的一个原则是图片能传递你想要传递的内容，而不仅仅是图片本身好看就用它
5. 在正式商业应用中，为了避免使用网络上的一些图片造成侵权，直接请摄影师拍摄相关的主题图片是更好更快的方式(推展开来，还可以请设计师画一些图标之类的)
6. 像设计师一样思考——永远从解决问题的角度去思考。去改变那些理应被改变的东西，即便它们已经成为了人们的"习惯"
7. 没有设计感的用户不会知道你的产品设计具体好在哪里，但重点是你要通过设计让用户在使用产品中感到舒服、自然、没有违和感
8. 学习新知识要把自己的左脑模式关闭，避免使用逻辑去理解、去匹配已有的认知。要使用右脑模式，接受所有的新东西，然后到一定的程度它们会自己融合
9. 人会拖延的原因很简单：害怕做了这件事，但又做得不完美。所以解决拖延症的办法就是骗自己的大脑说自己并不是在做这件事，比如不要说我在练习写作，而是说我在随便记录一些东西而已。
10. 虽然测试很有用，但这只是对代码而言，在商业环境中，更重要的是抢占商机

领悟：

自打上周四结束了全栈课程之后，每天的睡眠时间都长得有点难以接受（每天睡 9 个小时这样），常常是听到闹铃声音（定 7 个小时）会顺手关掉，等再次醒来的时候会马上感觉不对劲——觉得自己睡了太久然后跟这个世界脱节了——带着懊恼和自责。

虽然我知道"若有恒，又何须，三更眠，五更起"，但一天睡 7 个小时真的很 ok 了，封顶 8 个小时，多余的真的太浪费时间了，况且睡太久了起来会头疼——这种费时还无效的事情真的要少做。1 个小时可以干什么呢？太多事情了，写一篇博客、反馈练习 20 个英文单词、听一节课……

一个感受就是，你知道的越多，你想知道的就越多，所以时间再多都是不够用。怎么解决时间这个问题呢？我的脑子里有一个科学的路径，可现阶段只能是大多数人眼中的"科幻"，而我的终极目标（ xdite 所说的 "社会天命"）就是把这个科幻变成科学。

Decisional

形容：收获超多的一天：设计、测试、读书经验三节课。

努力：完成神秘的 RSpec 测试第一轮并再做第二轮

流逝的是时光，收获的是成长——第八周周记

SEPTEMBER 18, 2016

标题注解（不是正文的正文）：这大概是最短但却最意味深长的一篇周记了，虽然原本写了1000 多字，可写着写着忽然发现该说的全都在标题里了，所幸就删到只剩标题……以上。

第八周学到的概念和工具

SEPTEMBER 17, 2016

概念——“为你做的事赋予意义”

当你想要做一件相对长期的事——比如学习一门技能，你不需要很快的下决定，而是先花一些时间来为这件事情赋予意义，这会使得你在做事的过程中不再需要“坚持”或者“努力”。甚至在你开始做事之后，还要不断的为它赋予一层又一层更伟大的意义。

意义之所以如此重要和有效，是因为你在给这件事赋予意义的时候，就已经打开了自己的内在动力（自驱动）——因为你觉得自己做的事情非常有意义，以至于别人不让你做你会觉得难受。当你觉得自己要“努力”做事的时候，这背后其实是有一种“不得已而为之”的情绪在作祟，而这种情绪积累到一定的程度之后，你就很容易爆发——我这么“努力”了为什么还是没有好的结果、我这么“努力”了为什么他们都不能体谅一下……但，这种宣泄于事无补。所以，从此把“努力”这个字眼从自己的字典中拿掉吧，与其事后抱怨，不如事前“赋予意义”。

工具——Typora

Typora 是一个非常简洁的**所见即所得** MarkDown 编辑器。

市面上的 MarkDown 编辑器非常多，也各有千秋。之前我一直用的是 MWeb，但在写字的过程中我发现有个一非常不方便的地方，因为写字和写代码多少有些区别，我在写字的过程不希望看到各种标题、超链接等格式的源码，而是只看到最终的输出样式就可以了，这可以最大程度的聚焦在内容而不是格式。

Typora 很好的解决了这个问题，不同于其它编辑器，它只有一个窗口，这个窗口既是编辑区也是预览区——真正的实现了所见即所得。比如你输### 三号标题并回车，###的标记符号会被自动隐藏，三号标题这几个字会自动转化成最终输出的样式。而如果想要看到源码，通过 command + /快捷键可以切换到源码视图。

在编辑中文的状态下要频繁输入各种（半角）标记符号也是一场噩梦，因为中英文状态的切换很是繁琐（即便是仅用 caps lock 键）。而 Typora 的快捷键这时就可以派上用场了。比如将光标定位在标题行（不需要选中文本），利用 command + 1……5 快捷键可以快速的在一号标题到五号标题之间切换而不需要手动输入标记符号。再比如，在本章节的开头我想要把 Typora 这个单词变成超链接，我只需要在浏览器里复制好对应的网址，然后将光标放在 Typora 这个单词前面，按下 command + k 快捷键，一个超链接就瞬间生成了（原理是 Typora 会自动定位并选中光标之后的单词，将它作为链接文本，将粘贴板里的网址作为链接地址）。

当然 Typora 也不是完全没有缺点，比如每打开或新建一个文件，它就要开出一个新的窗口而不是使用标签页的形式，在需要同时打开多个文件的情况下，这种做法对于文件之间切换并不是最优的。同时由于缺乏了目录树的文件管理功能，想要在多个素材源查找资料也变得很麻烦，如果是在 MWeb 里，可以全局搜索存在 MWeb 档案库里的所有文件（标题、正文匹配）。

就我个人而言，Typora 的应用场景是不需要输入太多代码、图片类的写作场景。目前 Typora 还没有发布正式版，而 bate 版能做到如此好用并且免费使用，绝对是业界良心且值得推荐。

一样的月光，不一样的中秋 9/15 日记

SEPTEMBER 16, 2016

Objective

完成：1.浓缩书项目：Onboarding 细节打磨、项目 Demo；2.全栈班毕业典礼

Reflective

情绪：从紧张刺激到轻松活泼

Interpretive

学到：

1. 技术很重要，也很不重要；不要陷入争论，也不要抢着去追求新技术
2. 不要对学习感到饥饿，而是对解决问题感到饥饿
3. 先去“解决问题”，再去思考怎样才“不用解决问题”
4. 项目管理的问题不需要解决，团队有进步就是
5. 全栈思维很重要，思索全局却更重要
6. 不需要什么都会，花钱请专业的人来做
7. 先想清楚产品对他人的价值（从 Landing Page 找重点），再开始写代码
8. 永远问自己什么最重要
9. 学习所有的技术都是拼图而不是登山，找优秀的人带路
10. 永远要学会协作（哪怕是跟过去的自己、未来的自己），协作是进步的最快方式
11. 养成记录的习惯，特别是记录错误——下次就会绕开这个坑
12. 检讨错误，不陷入重复的坑就是一种进步——使用 After Action Review
13. 写作是总结自己的一种方式
14. 泛读的“泛”是“广泛”的读而不是“泛泛”的读
15. 一念一世界，你做的每一个决策，都要花时间去想，并不断的给它赋予一层又一层的伟大意义
16. 经过决策之后要长期做的事情，是不需要“坚持”或者“努力”的，因为意义本身已经是内在推动力（自驱动）——根本停不下来
17. 做自己精神世界（或者说平行时空）的巨人
18. 如果你把一件事情当成一份工作来做，久而久之就会产生”不得以“的情绪
19. 学会正确的预期，预演的能力、复杂情况的判断能力
20. Coding 就像是会识字，光会识字想写文章还不行，还得有思想
21. 十八般武艺是指学习的方法、思考方法、跟人正常沟通的方法
22. 一把锁头在那里，想要打开它，就去找钥匙，而不是盯着锁头看
23. 通过不断的升级来解决问题
24. 完整的作品非常重要，为了完成，就不能卡在一个环节停步不前，先跳过它，做其他的部分
25. 思考产品的核心商业模式，用户的哪些行为会对产品产生价值
26. 人生不会处处让你满意即便你已经殚精竭虑

领悟：

今天是中秋节，也是全栈班课程的最后一天，但依然是收获颇丰的一天。

第一、午饭后为了完成下午的浓缩书项目 Demo，开始了 3 个小时的 hackathon，整个过程大脑持续高速运转，几乎达到忘我的状态，感觉非常的好，以后多给自己找这样的状态。

这个项目也算是一个毕业作品，之前大家是 4～5 个人用了 4 个星期做出一个项目，而这次则是大约 8 个人用 2 天做了一个项目（从星期二下午开始，前两天忙到根本没时间写日

记），这次更高效的协作让我感到惊喜。
第二、xdite 老师和笑来老师的毕业典礼发言又是一大波干货袭来的即视感。之前说全栈班是对未来世界软件教育的一次实验，我真的很庆幸自己能加入这次实验——实验还非常成功。
第三、xdite 老师特地请来台北最厉害的一间居酒屋的老板娘过来给大家做日本料理，因为太好吃，每上一道菜都是瞬间被清空，因为太好吃，另外订的几百元外卖烧卤沦为陪衬——基本没人吃。现场看居酒屋老板娘做料理，最大的一个感受就是能把事情做做到极致的人，做事的态度都非常的认真。
Decisional
形容：感慨与收获非常多的一天，跟开学第一天一样，但已经比两个月前成长太多倍
努力：好好休息一天

Beyond Coding —— 我在新生大学全栈工程师训练营收获成长

SEPTEMBER 14, 2016
开宗明义，参加新生大学全栈工程师训练营（我习惯简称为全栈班）是目前为止我所有人生重大决策中最正确的一次选择——两个月的高强度训练，大量关于编程和非编程的概念、知识、价值观使我获得了前所未有的高速成长，而我的人生旅程新篇章也由此开启。

全栈班如何使我 Beyond Coding ？
新生大学全栈工程师训练营由李笑来老师发起、郑伊廷（Xdite）老师主讲，但课程内容可不止 Ruby on Rails 快速进阶这么简单，这里面包含了 Xdite 老师所擅长的三个知识领域：Ruby on Rails + 敏捷项目管理 + Growth Hack 。
这是一次成长蜕变之旅，也是我对《新生——七年就是一辈子》一次的践行——持续主动的升级自己的操作系统。我从《Beyond Feeling》的书名引申出“ Beyond Coding ”这个概念，并将这个概念总结为全栈班的一个核心理念——教与学都围绕着它进行，它可以很好的解释我在编程的世界里从零基础到可以和团队合作完成真实项目的过程和原理。
在我看来，Beyond Coding 就是跳出常见的编程学习方法和思维——大多数是低效甚至无效的，然后用全新的角度去理解编程的意义、用正确的方法学习编程以及更多与之相关的东西。这些会持续提升思考能力和个人进步空间的东西是无法粗暴的用金钱直接买到的——第一是必须要有很厉害的老师用正确的方法传授给你，第二是你必须主动进入到接收的频率才能捕获 。
1. 编程只占 20%
开学第一天，笑来老师就告诉大家一个一开始无法理解到最后慢慢领悟的结论：编程只占 20% 。
原因在于，编程对于新手而言是一道看上去非常高的门槛（实际上并不高）以至于无法看到门后的世界，事实却是一旦跨过这个门槛之后，编程本身就不再那么重要，而编程之外的 80% ——产品要实现什么价值，它的商业模式如何……？快速的把代码写好，然后花更多的时间去思考更多的这类问题。
2. 最小必要知识
很多人一想到全栈工程师，就觉得这人应该是十八般武艺样样精通的高手，但那其实是多年之后的成长结果。2/8 定律对于全栈的意义即 20% 的核心知识覆盖了 80% 的应用范畴，在最开始的阶段，你需要的只是运用最少必要知识让产品可以运行得起来就 ok 了——重要的是“用”，把学到的东西用起来，然后再去不断精进。
3. 编程的学习方法

a. 拼图理论

编程学习其实是有套路可循的，不要高估编程的难度然后把自己吓跑，也不要用线性增长的思维去想象成长的路径——否则遇到多几个挫折就觉得自己不进步了。记住类比影响思维，这一点很重要，错误的类比会使自己始终不得要领。

学习编程的方法就像拼图，是一个将不同的知识点逐渐拼凑完整的过程，所以最高效的方法就是先把外面的框拼好——对于 Rails 学习来说 Rails101 就是画框，好让自己知道应该在什么样的范围之内进行活动。

b. 肌肉记忆

在全栈班开始之前的课前作业里，Xdite 老师要求我们把 Rails101 重复练习至少 3 次，如果能重复到 5 次会更好。起初我无法理解这其中的用意，我只知道不要试着去问为什么，先照着老师说的去做就好了。

在之后的课程里我才逐渐明白，原来人能学会新的知识不是靠思考，而是记忆。学习编程的过程中要放下自己的傲慢，反复跟着练习去做。千万不要在一开始就试图去理解，因为你无法用未知解释未知，并且，现实生活中没有多少能跟编程概念互相匹配的经验。经过不断重复的练习，把最基本的东西练成肌肉记忆，进而大脑才有精力去捕捉到肌肉记忆之外的更多细节，这时很多关键点已经不言自明，而如果还有不明白的，再去找解答也变得容易理解。

c. 正向回馈

没有多少人会喜欢挫折感，因此新手学编程需要不断的做出能够动得起来的程序，这会比较容易获得成就感——还可以抵消因为遇到无数个 Bug 而产生的挫折感。在全栈班里，不论是个人项目还是团队合作，每个阶段做出的项目都可以获得极大的成就感——且持续增强。如今回想两个月前第一次做的 First-App ，那个页面简单得不忍直视，但当时的心情却是超激动的，因为它已经可以新增、删除文章了，并且是写在数据库里的！看着终端机里不断滚动的 log ，一种我是天才的感觉油然而生。很难想象，如果没有一步一步的成果激励，我如何能够快速成长——到现在已经有能力在团队里合作功能更复杂的真实项目。

4. 编程之外

a. 敏捷项目管理

经历了课程前 3 个星期的个人独立开发项目之后，我们在之后的时间里进行产品小组合作开发。在敏捷项目管理上面，包括但不仅限于使用 User Story 快速筛选功能实现的优先级；After Action Review 发现问题、提出改进建议并执行以改善结果；每天进行 Startup meeting 快速总结前一天的进展并分配当天任务；用 Github 管理代码以及 merge 功能分支；用 Tower（或其他项目管理工具）分配、管理任务进度。

b. Growth Hack

项目管理在于提高工作效率、确保产品如期交付，但对业绩增长却不是直接正相关。因此根据项目进度 Xdite 老师指导我们该引入 Growth Hack 来改善产品，包括使用 Landing Page 提高转化率，用 Onboarding 提高留存率，用 NPS 提高转介率等。

c. 持续成长

比你优秀的人比你还努力是什么样的体验？Xdite 老师在 Rails 开发、敏捷专案管理、Growth Hack 等方面的技术水平和教学能力都已经非常优秀，但她从来不曾停下自己的进步。每周她都会从网上买来 10 ~ 30 本某个领域的书，并用自己的中文速读能力去快速吸收，然后每周（甚至只隔个两三天）就兴奋的跟大家分享她学到的新知识，这种拼劲的的确确让我们感到震撼。

此外，班上来自各行各业的同学们也都非常的积极向上，在这里没人愿意有一丝怠慢——因为不进步就是在浪费自己的生命，这些的良好心态和行事风格对于个人的持续成长也是非常有帮助的。

我为什么选择全栈班？

在此之前我知道自己的很多想法可以通过编程实现或者实现的过程里少不了编程的那一部分，但却不知道如何开始。就在 6 月 29 日我还有一个可以通过编程解决的问题想得特别强烈，无巧不成书，第二天早上就看到了学习学习再学习公众号的全栈班招生公告，看完推送我就坐不住了，到现在我都还能想起当时的感受——我内心告诉自己，这个课一定要上！

1. 选择比努力更重要

2004 年前后我曾有过一段自学编程的经历，由于当时遇到了无法突破的瓶颈身边又无人指导，最后只好放弃。10 多年来的经历让我对自己个人成长多了一些领悟：如果想要在一个领域从零做起并逐步做到优秀，有一个经验很好的前辈指导一定是获得进步的最佳方式——比如我很少看到有几个运动员是自学成才的。

在这样的前提下，选择比努力重要的原因是，找名师指导对我而言是比较靠谱的做法。因为如果我自己捣鼓的话，也许能成功，但也有可能第一步“努力”的方向就直接搞错了——这种大块时间被浪费的事情要尽量避免。

2. 相信的力量

比较了解笑来老师的人都知道，他要经过长期思考并践行之后才会把总结到的概念分享给大家，他坚信先做到了再说。因此当笑来说“一年可以成为全栈工程师”，我就惊叹自己之前对编程难度的错误高估。而 Xdite 老师以知名 Rails 教练的身份补充说不用一年，两个月就可以了！有这么高的学习效率，为什么我要浪费自己宝贵的生命去做无用功的“努力”？为什么要做那些用战术上的勤奋掩盖战略上的懒惰的事情？既然我想要学编程，既然我想升级自己的操作系统，那我只需坚定的相信自己的选择，相信自己有能力比昨天更进一步，然后笃定的去践行就好了。

迈出这一步

我曾经错误的把“学习”和“上课”联系在一起，错误的以为从离开学校之后就是”脱离苦海“——终于不再需要”学习“了。后来我才逐渐明白，离开学校，才是“学习”的真正开始——象牙塔之外的世界有太多需要持续精进的地方。

两个月的全栈班课程，带我迈入了一个广阔的编程世界。随着课程的结束，属于我的探索之旅却由此正式开始，我想对这个世界说一声：嘿，我来了 ~

第七周学到的最棒概念——TEST

SEPTEMBER 11, 2016

第七周学到的最棒概念——TEST

经过四个星期的团队协作，随着新功能的不断加入，每个小组的代码量都增长到了一定程度，因此在这周 xdite 老师适时给大家引入了 测试 （Test） 的基本概念。在这之前，我对测试的理解是很模糊的，我只知道如果新增了一个功能，就要去走一下流程，看看会不会出现报错、有没有执行了正确的操作之类的，但我不知道竟然还有一个可以独立出来的“测试”概念——之前我会搞不懂 config/environments 里 development.rb 和 test.rb 这两个环境配置文件有什么区别。可以说了解到测试这个概念在很大程度上又是完成了一块拼图。本文主要根据前几天的日记重新梳理一下对测试的基本理解。

1. Rails 环境分为 Development（开发）、Test（测试）、Production（生产）共三种环境。
2. 测试（Test）是指写 ［测试代码］ 去**自动**测试 ［整体代码］ 是否正确。

3. 测试通常有 3 种：
   a. 单元测试（Unit test）。测试 model 或 service 里的某个 method 是否正确;
   b. 功能测试（Functional Test）。测试 action 是否能输出预期的 view 或者 redirect_to;
   c. 整合测试（Integration Test ）。测试一个真实情景的操作流程，比如从添加购物车到生成订单再到付款。
4. 通常用“绿灯”、“红灯”表示测试是否通过。
5. 当代码量比较少的时候，可以**手动**测试修改的代码是否正常 ;而当程序开始复杂后，有必要使用自动测试（Test）的减少重复劳动且保证程序可以随时正常运行。
6. TDD（Test-Driven Development）方法是先写测试程序预期结果，然后再写代码直到符合测试程序的要求。
7. 写 1 行代码通常需要再写 3 行代码来测试。
8. 初阶程序员不太建议写测试的原因是，你不知道它报的错是你的程序错了还是你写的测试程序本身就错了。
9. 不建议新手写测试的另一个原因就是写测试的时间比写 code 的时间还要耗费更多。
10. “测试工程师”是专业分工的体现。
11. RSpec 是一个 Ruby 的测试工具，在 Rails 中常用的是叫做 rspec－rails 的 gem（虽然 Rials 也有内建的 Test 功能，但……）。
12. Github 可以跟测试自动整合，绿灯后还可以自动 deploy 到 heroku 。
13. 测试的目的是验证代码是否正确而不是验证商业机会是否正确，商业的机会是不允许错过的，因此测试虽然重要，也要注重两者的平衡。

第七周遇到的“CSS”坑

SEPTEMBER 10, 2016

其实从第二周开始就一直有遇到 CSS 的坑，因为在第一周做 Joblisting 项目的时候，并没有太多的排版要求，基本放几个 Table 标签就算是可以了。而从第二周的 JDstore 项目开始，随着页面数量及内容的增加，不用到 CSS （甚至还要包括 JS、JQ）就会很难合理的呈现内容，更不用谈及 Onboarding 或者用户体验之类。

本周三在做 project index 的 Tab 切换效果时就再次遇到 CSS 大坑。我想要实现的效果是点击切换”最新项目“或者”往期项目“之后，当前标签页的选项卡可以高亮显示——以任何一种样式提示用户正在浏览的页面。综合分析下来有三种方案：

1. 使用 Gem active_link_to。但这个在用户后台已经使用，由于具体样式还不知道怎修改，所以再用在首页就不太适合了。
2. JS + CSS。JS 也是一个坑，先绕行……
3. 纯 CSS。似乎比较容易实现，就只能选它了。

然而在实现的过程中，由于对每个 class 应该用到的属性不甚了解，各个 div 的嵌套关系也在相互影响，还有不少地方涉及到之前已有 class 的更改，真有牵一发而动全身的感觉，所以最后花了几个小时还是很难调试出自己想要的效果。

总结起来，还是要把基本功练好，这样不至于盲目的改各个属性却不得要领。

附上两个链接

- 基本的 CSS 入门： Code School CSS Cross-Country
- 相应的读书笔记： CSS 基礎技巧懶人包

选择真的比努力重要 7/25 日记

JULY 25, 2016

Objective

记得：Computational Thinking、User Stroy、全栈工程师的定义、人生三大坑、人生最重要的东西

完成：给 job – listing 挂 bootstrap、simple_form、devise

Reflective

情绪：激动（哈哈，笑来老师和 xdite 老师在我看来都属于名人，而我从来没有这么近距离的接触过名人）

高峰：两位老师的演讲（都是高级干货，受益匪浅）

低点：没有

Interpretive

学到：

xdite 老师讲的 Computational Thinking 和 User Strory，尤其是实际做 Must Have、Should Have 的两次分组练习，一下子对程序开发的流程就豁然开朗了。新手提问存在的最大问题是根本不知道自己遇到什么问题，如果你的问题是“我该怎么建立一个招聘网站”，高手们当然不知道从何回答。

学会 Computational Thinking 和 User Stroy，你就知道该如何把大问题解构成小问题，如何把未知问题结构成已知问题，在这之后，你的问题大概会是“我该如何建立招聘网站的登陆系统”、“我该如何增加一个删除招聘信息的按钮”之类的具体问题。

笑来老师讲的人生三大坑和人生最重要的东西，感觉干货十足，很多东西如果不是旁人指点，自己很可能一辈子都不知道。在我看来笑来老师最大的特点就是善于总结，这种从事物的本质去反推的“第一原理思考”，必须学习并践行。

领悟：

开学前 A 朋友转告我说他的一个同学是做网站的，那个人觉得我花 5 万块去学做网站太贵了，我当然没有见过那个人，我只告诉 A 朋友说我不是去“学做网站”这么简单的事情。

去年我听一个销售课程，里面有两个观点从某些层面来说强化了我要参加 Rails 全栈班的决心。

观点一：你没钱？正是因为你没钱才更要借钱去学习，不然你就会永远掉入“等我有钱了”的恶性循环且很难解开。

观点二：教练的级别决定选手的表现。

我跟 B 朋友借钱参加课程，B 朋友很赞成我花钱投资头脑，但是对于这么大的一笔费用，他还是有点无法理解，我告诉他我这是要从顶尖的人那里学习顶尖的思维，就像有人愿意花“天价”去和巴菲特共进午餐那样，然后他终于明白自己为什么没有太多的想法，因为身边的人“都没什么想法”。

我只想说，光是今天一天的课程就已经值回票价了，至于后面的两个月，那就是赚到了。比如 User Stroy，如果你是一个“野生程序员”，你要花多少的时间、经历多少的磨难才能领悟这个最基本却最重要的概念（还记得 2/8 定律吗）？比如注意力，你要花多少的时间才能明白时间不能管理，该管理的是自己的注意力？你甚至可能一辈子都领悟不到这个东西！我想选择比努力重要的说法也是由来于此，你的选择决定了你努力的方向是否是正确的或者值得的。很多人不加思索的觉得既然选择重要，那就不用努力了，拜托这些人看清楚一点，是选择比努力重要，不是说努力就因此不重要了，这不是 if else。

Decisional

形容：漫长的一天，却是前所未有的受益匪浅的一天。
努力：完成当天的任务，并完成课前中级练习第三遍，从拖延的死循环中跳出不致于掉队。
Posted by nfreeness July 25, 2016
Tweet

这么简单（那么困难）——全栈班第五周周记

AUGUST 27, 2016
这两天跟着几位同学和 xdite 老师闲聊，她有提到的一个观点是所有的编程学习过程其实都有套路可循，既没有想象的那么困难，也没有想象的那么简单，但一年时间还是可以达到 master 水平。
可能在很多人看来这有点像是正确的废话，但，正所谓大道至简，不然哪来的那么多人会觉得“听说过很多道理，依然过不好这一生”。
不管是困难还是简单，我想班里的不少同学都和我一样深有感悟，在 7 月初的时候，我们甚至连 Ruby on rails 是什么都不懂，而到了现在，大家能自己写 CRUD、自己做功能、自己 debug 、自己做出小型购物网站，甚至开始团队合作开发项目，这一切，真的没那么困难。很多的知识，一旦入了门，了解了概念和原理，至少从心理层面会减少很多的恐惧情绪，所需要的无非就是不断充实自己的方法论，同时还可以通过不断获得的成就感给予自己正面反馈，形成良性循环。
在写代码的过程里，当然也有很多难倒自己的地方，可能以为自己懂了、以为自己写对了，但是报错的页面还是不可避免的随时出现。又或者觉得这个功能“应该”就是这么写，但其实还有很多没有考虑的多个状态、用户权限的相互牵扯，等到认真分析背后的运行逻辑之后，会发现存在很多的盲点和漏洞，这些都是事实上的“没那么简单”，当然还有更多。
回顾本周，原计划周一周二完成功能，周三周四调整页面，周五达到上线要求。不过真正执行起来还是有一定的难度，首先是功能基本到周三才完成，周四则是把所有的 bug 和所有需要改进的地方提出来并进行修改，结果提出来的改进和 bug 有 7、80 条，即便 xdite 老师帮我们进行优先级筛选，也还有是一半的任务要完成，而周四一天根本没法做这么多。
于本周而言，“简单”的是我们完成了大部分必要的功能——比如手机短信验证、站内搜索、资金流水、project 审核机制、站外视频播放、project 动态发布管理等等，似乎写代码是大家更擅长的事。“困难”的是页面的样式除了 project show 页面进行了一些初步的调整，其他的页面都没有太大进展，整体的用户体验流畅度和页面的样式都还没达到上线水准。这些有待提升的点在于 css 必要知识、页面样式如何与功能结合、色彩搭配、基本的设计理念等。
虽然五个星期的实战看上去还剩三个星期的“一大半”时间，但是按照上线为最终目的来看的话，其实时间是很紧的，下周的大致方向就是不再需要增加功能，聚焦在完善 Landing Page、Onboarding 这两个主要内容上。借用一句话来总结——“革命尚未成功，同志仍需努力”。
Posted by nfreeness August 27, 2016