

NIEYICHENG'S BLOG

- [ABOUT ME](#)
- [ARCHIVE](#)
- [FEEDS](#)

18 天 ago

[第七周周记](#)

这周一直在做网站的收尾工作，功能上基本没有更新，只是在修一些 BUG 和一些细节上的东西，虽然最后和想象中有很大的差距，但是也算是告一段落了，这一周没有之前那么焦虑，可能是因为要收尾了，心态反而更好了，这一周大家心态都比较好，没有出现什么争执。全栈培训即将结束，有开心，有失落，总之以良好的心态面对一切是最重要的。

- September 11, 2016 10:50
- [Permalink](#)

26 天 ago

[第六周最大的坑](#)

这周最大的坑是思想上的坑

被自己的想法束缚住了，直到后来才想清楚问题的本质，造成如今困局的还是自己，和别人没有任何关系，格局太小，不清楚问题的本质，过分在乎别人和自己的心情，现在看来就是纯粹的浪费时间，不仅浪费自己的时间，还浪费了很多人的时间，所以老师说，别他妈的想太多，就做！直到现在才醒悟，是有点太晚了，不过自己也终于清醒过来了，已经浪费了太多时间，算了，就别再浪费时间想了。

- September 02, 2016 17:14
- [Permalink](#)

26 天 ago

[第六周学到最棒的东西](#)

学到最棒的东西

这一周学到最棒的东西，应该就是明白了不畏难，不怕难。

从刚上课一直到自己做项目，路途中遇到很多困难，不过有老师的指引都一路跨过来了，但是自己做项目的时候，大部分时候是没有答案的，所以做一些困难功能的时候难免会产生畏惧感，所以一开始自己总是不太敢做难的东西，把任务分配给编程能力更强的班长，原因有两个，第一个不确定自己会不会，第二个怕自己做的慢拖垮整个项目的进度，所以就把方向倾向于自己更擅长的业务逻辑，花了大量的时间在上面，后来被老师及时呵斥才醒悟过来，我应该更专注于技术，所以当把重心转向技术的时候，发现只要把畏难心调整之后，一切根本没有想象的那么困难，做的过程中，你会发现，只要你想前进，全世界都会帮你。

- September 02, 2016 17:1

26 天 ago

[第六周周记](#)

这一周基本上没有做什么新功能，一直在修复和完善，周五 demo 的时候，发现其他团队的项目都很完善，而且很炫酷，我们比起来就逊色了很多，所以内心依然有一些失落感，但更多的其实是无奈，作为几次创业的人，深知一个好的团队的执行力比差的团队是强太多的，所以别的团队能做出好的产品我是毫不意外的，因为人家在讨论的怎么样让这个项目更好，为了一些细节吹毛求疵，而我们为了合并不合并谁的分支产生矛盾，真是懒得说。还有两个星期课程就结束了，我不想留下太多的遗憾，为了迁就，让一个项目最后没有做好。现在开始没有迁就，没有妥协，全力以赴，只对事不对人，只为了把项目做好。

- September 02, 2016 17:12

26 天 ago

[第五周最棒的概念](#)

第五周最棒的概念应该就是 onboarding，通过两部分建立起一套完善的用户体验

part1

- 1.Before start date
- 2.Aroud start date
- 3.start date
- 4.The first part
- 5.中间
- 6.closing & Folowup

part2

- 1.在 start date 前，客户会寄信来问你什么问题
- 2.在 start date 当天，客户会忘记什么让体验弄砸
- 3.他们最常做了什么“正确的事”达到很好的体验
- 4.他们最长做了什么“错误的事”结果收到很糟糕的体验
- 5.东西卖出后你如何检验他们“做了正确的事”或“做了错误的事”？
- 6.他们如何联络你修正问题
- 7.你怎么事后补偿的 instruction？
- 8.你希望他们如何事后帮你行销

通过自己思考设计一套流程，通过一系列的问答让自己能彻底了解一个用户可能的需求，然后再通过模拟用户反馈不断的修正这一系列的行为，构成了一套完整的培养用户习惯的方式，太厉害了。

- September 02, 2016 17:12

30 天 ago

8.29 日 time-log

8 : 10 - 9 : 00

与班长讨论项目进度，并思考今天要做任务

9:00 -10:00 思考并写下今天的任务以及思考每个任务的分配

10:00-10:30 开会叙说彼此进度，以及讨论分配任务

10:30-11:40 配置阿里云测试，网站搭上去能看 但是还有几个问题

1.图片无法上传，可能需要其他 gem 的支持

2.目前服务器处在开发环境中 需要转换到正式环境里

4.邮件无法在远端启用

3.现阶段 push 到 heroku 与阿里云有区别很大，尝试发现相同以及不同的地方

11:40-12:15 吃饭 扯淡

12.15-2:00

首页课程编辑与后台课程详情编辑区分，增加栏位 hero_title，修改后台标题为中文，更容易理解

排版用户个人中心与 admin 看到的用户中心使其一致

去除 views,body 里的 style 背景，统一放到 css 里的 body 里

去除 chapters,courses,faqs,posts,tasks,users,works 的重复代码，使其更整洁。

2:00-3:00

查看测试每个 push 上的 github 分支，没有问题，与班长讨论进行 merge

3:00-4:20

写周记，与班长讨论接下来要做的功能以及方向

4:20-6:00

写本周最坑，以及安排讨论工作，订饭。

6:00-7:00

写 After Action Review

7:00-8:00

吃饭，思考

8:00-9:00

开 Review 会议 检讨自己的过失

9:00-9:30

tower 上写 After Action Review

- August 29, 2016 23:32

- [Permalink](#)

大约 1 个月 ago

第五周最大的坑

我觉得这周最大的坑可能就是陷入的自己的思维陷阱里，忘记了当初学习的初衷，纠结于业务逻辑的实现，抛弃后端去折腾前端，于是老师安排后端之后，有一些傻眼，为了项目推进

只能让班长继续专注于后端功能的实现，我来负责一些小功能以及团队的任务安排协调，因为自身的局限导致项目进展缓慢，我负主要责任。

所以，无论什么时候都不能也不要忘记当初做这件事的初衷。

- August 29, 2016 16:59

- [Permalink](#)

大约 1 个月 ago

[第五周周记](#)

这一周进度较上一周进度有所提升，但是还是很慢，所以基本上每天都被老师骂。

这一周马师傅积极参与了进来，做出了一些比较重要的功能，所以我觉得团队不能分散，必须齐心才有可能做好一个项目，周五的时候和大家说了一下大概的业务逻辑与运营的方向，大家还是表示比较认同的，其实这两周大多数在思考这些东西，我觉得如果你把这个项目当做一个真实的项目去做，你必须有一个清楚的逻辑，不然大家跟着你东做一块，西做一块没有任何意义，说简单点，没有地图，你知道下一步怎么走？我不希望蒙着头做出的东西没有任何用处，那么我只能从未来的业务逻辑发展来判断什么是 must have。

可能因为花费了太多时间去思考业务逻辑以及协调团队，在真正的项目上进展缓慢，让老师对我们产生了质疑，我觉得老师可能更想让我们专注于技术而不是业务逻辑，这点我也很清楚，但是可能是之前做项目习惯了以运营或者产品的角度去思考，而不是以一个技术角度去思考这些问题。

说到本质还是因为个人能力的不足，所以我愿意接受 Xdite 老师所有的批评，以目的性的角度来看，没有做好就是没有做好，没有任何借口，我会尽力提升自己能力，以满足团队项目的快速推进。

- August 29, 2016 16:41

- [Permalink](#)

大约 1 个月 ago

[8/15 日记](#)

今天老师要求我们做了 landingpage，每个人都得想一个项目然后上台 demo，然后从 19 个项目里选 3-4 个开始做，最后老师定了四个项目，我有幸被选上，小组被重新打乱排序。

之前一个人的奋斗，变成了一组人一起做一件事，虽然之前创过业，但是我并不擅长管理团队，所以虽然我们团队人很少，可是还是有一定的压力，而且我们做的项目其实是老师的项目，所以压力更大，可是压力也是一个锻炼的机会，我们一定尽力不把老师的牌子弄砸了。

- August 16, 2016 09:45

- [Permalink](#)

大约 1 个月 ago

用户调研

提供线上的以目的性为导向的编程教学

feature

1. 快速完成作品达成成就感
2. 让你在线上也能体验到类似线下的班级氛围
3. 帮你躲避大多数新手坑
4. 帮助你迅速达成目标横空出世
5. 几十人亲身验证 0 基础如何成为全栈工程师的教学理念
6. 真正针对 0 基础学员提供 step by step 的教学模式

How it works

1. 选择你想学的语言，选择你想学习语言达成的目标（购物网站，JD,APP）
2. 选择学习方式，1.最小可行性方案（只学习必要知识大达成方案）2.相关知识的全面拓展
3. 使用实战派教学理念，从实战入手而非学院派的理论知识，提供每个步骤的目标源码给用户，每完成一个 step 才可以进阶到下一个 step 去学习，并且每一个 step 都有时间期限。
4. 班级教学制度，每个星期通过网络进行 2 小时的授课，提供在线录音，讲解精华部分
5. 当完成所有步骤并且成功建立起网站成品之后，进行必要部分理论教学，从实战切入理论，让你明白自己在干什么。
6. 每个人都可以在班级的讨论版里提交遇到的问题，必须符合规则和不能是重复的问题，所有的问题以及回答必须公开，我们的助教会 24 小时之内答复学员的问题，当然同学帮助解答问题，是有奖励制度。
7. 每一期课程优秀的学员可以获得被推荐面试的机会。和别的在线编程教学网站的区别是什么？
 1. 我们是以目的性为导向，根据你最终想做出的产品来制定你的学习课程，让你以最快速度达成目标。
 2. 新手自学编程遇到最大的坑就是，遇到问题无法解决，首先我们课程是 0 基础的会 step by step 教你从配置环境到一步步构建出网站，并且会列出很多新手会遇到问题的答案，其次就算遇到没有列出的问题，我们也会有专门的提问板块，你可以按照格式上去提问，也可以看到很多你这一班同学所遇到的问题，也许就有你要的答案，如果实在没有的话，我们助教会尽速答复。

客户调查

1.当你在线学习你想学习的编程语言的时候，你最讨厌的事情是什么？

沒有時間學習，一旦中斷很難繼續。

首先是線上課程資源通常不會寫大約需要的時間，在學習的過程中，如果因為其他事情突然中斷了，很難再回到上次的階段。

2.当你知道有一种在线编程课程可以从 0 基础教你如何迅速做出一个你想做的网站（APP），

你会怎么想？

多少錢？

需要多少時間？

學會了這個技巧，還可以幫助我達到什麼目標？

中小企業主、個人企業主

無法或不想購買技術資源，沒有能力養一個編程團隊，所以想要自己製作一個網站，可以簡單的修改上新，獲得技術諮詢/支援。

3.当你通过在线编程课程做出你想做的网站（APP）后，你最想达成什么目标？

幫一個非營利性質的網站翻修更新。

1.当你在线学习你想学习的编程语言的时候，你最讨厌的事情是什么？

寫編程寫到卡關找不到助教幫忙

自己又看不懂出錯在哪沒辦法自己解決

又或是自己不是編程出身

搞不懂簡單的程式語法

2.当你知道有一种在线编程课程可以从 0 基础教你如何迅速做出一个你想做的网站（APP），你会怎么想？

像我們這些不是從編程出身對程式語言完全一竅不通，真的可以做出我想要的網站嗎？

3.当你通过在线编程课程做出你想做的网站（APP）后，你最想达成什么目标？

作為應徵工作或是假使產品做的出色找人投資或創業

第三周周记

这一周一直在写代码，每天沉浸在代码里，花很少的时间停留在思考里，基本上 xdite 老师要求完成任务都完成了。

但是我在写代码的时候，已经习惯性去参考之前的代码，如果突然让我不看参考做一个功能，我甚至不知道第一步是什么第二步是什么，我发现我可能已经走偏了，笑来老师说，20%的时间是用来写代码 80%的时间用来思考，没想到我却反其道而行之了。

很多人讨厌思考，曾经的我也是这样，但是不思考根本无法进化成人，对我来说旧有的经验可能过于根深蒂固，再次让我把思考交给了老师 甚至其他人，因为放弃思考，导致我竟然以为自己学会了，所以内心充斥着两种声音，一个告诉我，这些我都会，另一个告诉我，你会个屁。

我发现因为不愿意思考，导致我每天写日记都很敷衍，因为一天的行为都是下意识行为完成的，我甚至不知道我学了什么，看着别的同学写的东西，我觉得自己都被碾压成渣了，既然知道了，我就不能让这种情况继续下去，其实最可怕的是你不知道你自己不知道，好歹我现在知道了，也算是一种进步吧，加油！

- August 14, 2016 22:18

- [Permalink](#)

大约 2 个月 ago

第三周学到最棒的概念

这周最棒的概念

Growth Hack

Growth Hack 的核心其实就是把产品做好。

Growth Hack 就是如何把好的产品营销出去。

- 寻找传统营销漏洞并改进
- Landing page 打造有影响力的品牌
- Onboarding 引导用户发现价值
- Customer Support 客服就是金矿
- Email List 留住你的销售机会
- Referral 与 NPS,旧客户是最好的营销
- August 14, 2016 10:50

- [Permalink](#)

大约 2 个月 ago

第三周遇到最大的坑

这周遇到最大的坑

学习到的东西感觉都会，但是要仔细想的话又发现什么都不会，我发现是没有认真的整理所学过的知识所以导致东西比较混乱，突然想做一个功能的时候不知道该如何下手，所以把思路弄清楚很重要，初学的时候可以像无头苍蝇一样乱撞，但是学到一定程度的时候一定要及时的整理，不然大脑会一片混乱,因为知识就像一个一个记忆孤岛一样，你不想办法把他们连接成大陆，他们就会慢慢的沉下去。

- August 13, 2016 23:35

- [Permalink](#)

大约 2 个月 ago

delete_all 與 destroy_all 的區別與用法

If you want to delete the User and all associated objects -> destroy_all However, if you just want to delete the User without suppressing all associated objects -> delete_all

According to this post : Rails :dependent => :destroy VS :dependent => :delete_all

destroy / destroy_all: The associated objects are destroyed alongside this object by calling their destroy method

delete / delete_all: All associated objects are destroyed immediately without calling their :destroy method.

delete will only delete current object record from db but not its associated children records

from db.

destroy will delete current object record from db and also its associated children record from db.

Their use really matters:

If your multiple parent objects share common children objects, then calling destroy on specific parent object will delete children objects which are shared among other multiple parents.

- August 12, 2016 14:38

- [Permalink](#)

大约 2 个月 ago

8/11 日记

1.学到了什么

1.一个字段(栏位)只能有一种状态

2.aasm 的使用

3.case when 的使用

4.if elsif 的用法。

5.gem"awesome_rails_console" 能让后台变的 awesome

2.Q&A

如何在不同的 views 页面里引用非对应 controller 里的实例变量

3.感悟

学无止境。

4.有什么需要改进的

今天在一个地方卡壳太久，钻进了牛角尖陷阱，导致了浪费了大量的时间在自己无法解决的问题上，效率低下，还有遇到问题一定要记录，我发现我遇到小问题就不记录了，这不是一种好行为，只要超过 1 分钟未解决的问题，或者疑惑的地方就应该记录

- August 12, 2016 14:35

- [Permalink](#)

大约 2 个月 ago

如何在 heroku 上上传图片以及如何隐藏自己的密钥

如何在在 heroku 上可以上传图片以及如何隐藏密钥

1.要安装一个 gem 叫做 fog

2.新增 config/initializers/carrierwave.rb，填入教程里的代码

3.需要有一个空间可以上传图片，可以去 AWS 注册一个账号 然后到账号下拉选单里面点 security Credentials 然后选导览列的 Details 可以看到有个 [+Accesss Keys]准备按可以申请 key 跟 secret，然后去 s3 页面创建一个 bucket_name 和选择一个 region

4.不要把 aws 的 id 与 key push 到 github 上面，避免被盗用，可以使用 figaro 或者 heroku config

5.app/uploader/image_uploader.rb 隐藏 storage :file 加上 storage :fog 会导致本地服务器

页面无法预览

6.使用 figaro 来隐藏密钥

1. 安装 figaro gem 'figaro' , bundle ,figaro install
会产生一个 config/application.yml 的文件,并且会被.gitignore 文件自动隐藏。
2. cp config/application.yml config/application.yml.example 复制一个例子文件告诉别人你有这个文件
3. 在该文件中加入

production:

AWS_ACCESS_KEY_ID: "" # 你的 key

AWS_SECRET_ACCESS_KEY: ""# 你的 secret key

AWS_BUCKET_NAME: ""# 你的 S3 bucket 的 Region 位置

REGION: ""# 你設定的 bucket name

然后就可以在 carrierwave 的文件中加入 上面的名字类似下图。这样就可以保护你的 key!

```
if Rails.env.production?
  config.storage :fog
  config.fog_credentials = {
    provider:          'AWS',
    aws_access_key_id: ENV["AWS_ACCESS_KEY_ID"],

    aws_secret_access_key: ENV["AWS_SECRET_ACCESS_KEY"],

    region:            ENV['REGION'] # 你的 S3 bucket

  }
  config.fog_directory = ENV["AWS_BUCKET_NAME"] # 你設定的

else
  config.storage :file
end
end
```

1. 在 terminal 中输入 figaro heroku:set -e production 部署到 heroku 上
2. 然后在重新 commit & push to heroku 这些设定会生效。

感谢邵俊达大神提供的各种技术帮助

- August 10, 2016 23:18
- [Permalink](#)

大约 2 个月 ago

[8/10 日记](#)

1.今天学了什么

1. 如何修改 PUSH HEROKU 的名字
 1. cd .git
 2. atom config
2. Growth Hack
3. 如何申请 aws
4. [如何上传图片到 heroku 以及如何隐藏自己的秘钥](#)

2.Q&A

问题太多，还没整理

3.我有什么感想

今天听了 Growth Hack 的课程，我觉得营销手段真是到处都是，把营销手段玩好确实是可以随便赚钱，但是如果没有好的产品就会变成奸商，而且不会长久，当有一个强大的产品再配上 Growth Hack 的理念，我觉得真的会所向披靡。总之少废话，赶紧学习。

4.有哪些需要改进

今天学习状态还行，基本上遇到的坑都是在邵大神的帮助下解决的，实在是佩服的五体投地，但是我觉得遇到问题最好还是给自己定个时间尝试，然后记录自己的解题思路，确实解决不了再问，直接寻求答案会让人放弃思考，这并不是一件好事，学习都是从未知向更多的未知迈进，所以学的越多其实恐惧会越盛，只有克服恐惧，才会让你进步的更快，嘴上说不恐惧其实遇到问题还是会恐惧的，哎，还是回去翻翻把时间当作朋友吧。

- August 10, 2016 23:13

- [Permalink](#)

大约 2 个月 ago

[8/9 日记](#)

8/9 日记

1.今天学会了哪些东西

1. 在 controller 里写代码，尽量不要写缩写，因为要让语言更加人性化
2. 在循环里可以缩写 `XX do |f|`
3. include 是包含什么
4. `add_to_cat(@product,_useless)`加_是说明这东西不重要
5. rubocop 这个 gem 会帮助你自动批改代码
6. views 里面循环需要去 controller 里面调方法,循环里的方法可以自己起名字

```

<tbody>
  <tr>
    <% @orders.each do |order| %> <!--可以任意更改，相应的下面也需要改-->
    <td><%= link_to(order.token , order_path(order.token)) %></td>
    <td><%= order.total %></td>
    <% order.product_lists.each do |order| %>

    </tr>
    <tr>
      <td>
        <%= order.product_name %>
      </td>
    </tr>
  <% end %>
<% end %>
</tbody>

```

7.

循环里面可以嵌套循环，比如我想调用 product_lists 里的 product_name，但是变量 @orders 没有办法直接调用 product_name，那么可以嵌套一个循环但是这个 order 必须和 product_lists 有从属关系，那么就可以在 order.rb 加一个 has_many :product_lists 然后就可以调用 product_lists 里的参数。

2.Q&A

问题很多，还没有整理，但是我觉得大部分可以自己解决。

3.有什么感想

所谓成功太慢，幸福减半，现在算是彻底明白这句话的含义，最近梦里经常惊醒，梦见子欲养而亲不待的场景，这可能是我最惧怕的事情，到这个年龄一事无成说起来也挺丢人的，虽说有钱与否并不是我定义成功的标准，但是有能力没钱这个场景在现今这个社会是基本不可能出现的，所以说到底依然是我没有能力，因为没有能力错过了太多不该错过的，因为没有能力造成了很多无法挽回的遗憾，而现在我已经不想让遗憾继续下去。

4.有哪些需要改进

专注于学习，专注于学习，专注于学习。

- August 10, 2016 00:22

- [Permalink](#)

大约 2 个月 ago

8/8 日记

8/8 日记

1.今天老师讲了什么

1.如何把商品加入购物车

2.在 navbar 上显示加入几件商品

3.如何计算商品的总价

2.今天学会了哪些

1.库存为 0 的货品不能购买

在 views 里的 show 里的购物车上面加一个判断式 if @product.quantity > 0

2.清空购物车

可以在 cart_controller 里新加入一个方法,然后把 current_cart.destroy 掉, 在 routes 里加入 collection 方法, 执行 delete, 再添加路径到 views 里。

3.可以删除购物车内某一物品

新建一个 method 然后去 CartItem 去找数据, 然后拿来销毁, 在 routes 里加入 member 下的方法, 执行 delete

4.可以更改购物车内购买的数量

在 carts_controller 里建立两个新方法, 先去用 find 寻找 item 的数据, 然后让他自身+1 然后保存, 可以把部分数据放在 model 里, routes 里加入方法, views 里增添 routes 路径。

5.在购物车新增数量时候, 不能更新超过原有库存的数量

在 views 页面里加入判断方法 if cart_item.quantity < cart_item.product.quantity 这需要先先在 controller 加入一个新的方法来调用 Product 数据库里的数据。

6.views 里只能调用对应的 controller 里的方法, 如果想调用其他 controller 里的方法, 需要在对应的 controller 里的加入方法然后去其他数据库里取值。

7.views 里什么用@什么时候不用@, 当 views 里的页面上没有出现 数据遍历时需要加@, 出现数据遍历时不要加@, 加@是因为要从对应的 controller 里调用变量。

3.Q&A

今天没有遇到什么问题, 可能是因为不会的太多了吧...

4.我有什么感想

今天做任务一直没有停下来, 感觉自己还是有很多不会的, 多亏我们这组学神的帮忙才把作业全部写完, 今天学到了很多, 发现其他组做这些也挺快的, 发现别人做完, 什么都清楚, 我做完还一脑子浆糊, 已经被彻底碾压了。

今天老师加入了个人分数, 觉得这招太狠了, 这是根本停不下来的节奏啊!

5.有哪些需要改进

还是浪费了不少注意力, 我觉得想要提升注意力必须要有方法, 我得尝试找到一套适合自己的方法论。

- August 08, 2016 19:41

- [Permalink](#)

大约 2 个月 ago

第二周周记

第二周感想

这一周是比较轻松的一周, 刚开始的前两天就把一周的东西做完了, 然后开始有点懈怠, 但是事实上大家的进度其实都是差不多的, 之后几天都忘记自己做了什么, 然后每一天都迅速的过去了, 直到第二周的最后一天 (今天), 才猛然惊醒, 突然发现自己这一周没有什么进步, 竟然莫名其妙的开始膨胀了, 细思之下发现是不是 xdite 老师故意安排比较轻松的任务, 然后想要看看大家的自学能力.....。

那天听助教说, 已经有同学能不看答案把东西做出来的, 觉得自己瞬间被甩开到千里之外。笑来老师说, 注意力是人最宝贵的资源, 我发现即使现在每天看起来都很认真的在做事, 事实上也是有大量的注意力在被我浪费掉, 我觉得如果能在该专注的时间能专注在自己该专注

的事情上，实际上根本不用谈努力。

这个世界上最可怕的事情是，比你牛逼的人比你更专注，更努力。

我竟然会为自己的一点小进步开始瞎得瑟，我是有多蠢。

认清自己，认清现实，着眼于未来的目标，而不是沉浸于自己那一点小进步。

第二周周记

第二周感想

这一周是比较轻松的一周，刚开始的前两天就把一周的东西做完了，然后开始有点懈怠，但是事实上大家的进度其实都是差不多的，之后几天都忘记自己做了什么，然后每一天都迅速的过去了，直到第二周的最后一天（今天），才猛然惊醒，突然发现自己这一周没有什么进步，竟然莫名其妙的开始膨胀了，细思之下发现是不是 xdite 老师故意安排比较轻松的任务，然后想要看看大家的自学能力.....。

那天听助教说，已经有同学能不看答案把东西做出来的，觉得自己瞬间被甩开到千里之外。笑来老师说，注意力是人最宝贵的资源，我发现即使现在每天看起来都很认真的在做事，事实上也是有大量的注意力在被我浪费掉，我觉得如果能在该专注的时间能专注在自己该专注的事情上，实际上根本不用谈努力。

这个世界上最可怕的事情是，比你牛逼的人比你更专注，更努力。

我竟然会为自己的一点小进步开始瞎得瑟，我是有多蠢。

认清自己，认清现实，着眼于未来的目标，而不是沉浸于自己那一点小进步。

- August 07, 2016 10:58

- [Permalink](#)

大约 2 个月 ago

第二周遇到最大的坑

没有遇到什么太大坑，基本上第一周已经走完因为这一周的任务和第一周很像，所以做的很顺畅，我觉得一定要说遇到最大的坑，就是没有遇到什么坑，现在突然发现遇到坑是一件让人开心的事情，因为又可以让自己的 debug 能力提升一些了，因为 xdite 老师都有教我们要重点纪录错误的事情，因为这样才不会再犯错。我没有上课之前，写代码的时候遇到问题，我只要看到不是代码打错导致的问题，我就直接 reset 回之前的做，因为我当时觉得自己并没有解决 bug 的能力，这样的好处是我避开了很多新手学一门技术的坑，但是由于我当时并没有做错误的纪录，所以错失了让自己成长的机会，这点让我很惋惜。

- August 07, 2016 10:56

- [Permalink](#)

大约 2 个月 ago

8/4 日记

1.老师今天讲了什么

今天老师讲了自己 hackathon 的经历，主要赢得奖项有几条

1. 幻灯片要做好
2. 要做减法而不是加法
3. 一定要先做 must have
4. 一定要花时间进行测试，不然上场爆掉就完蛋了。

今天完成了 图片上传、页面美化以及把商品加入收藏夹的功能

2.今天学会了哪些

1. 如何利用 carrierwave 上传图片和利用 mini_magic 显示图片
2. 定义页面的样式方法有很多，可以利用 bootstrap 里已经定义过的 css 样式，也可以自己定义 css 来改变页面布局。
3. 每个 div 都是一个块，网页都是由一块一块拼出来的
4. 终于明白<div class="col-md-x">是什么意思了，就是它在不同的屏幕下，会把页面切成几块，一共 12 块，每一块元素会占几个格子，可以利用这个让商品自动换行。
5. 执行 each do 代码的时候一定要记住代码的摆放位置，如果你把代码放在一个 div 里，那么它就会在这一个 div 里把每个遍历一遍。

3.Q&A

今天有点太顺畅没有遇到问题。

4.我有什么感想

突然发现 xdite 老师这种学习模式，

是先模仿然后尝试做、再做的过程中再慢慢学习，特别让人容易产生成就感。

因为看到自己有东西做出来这种成就感很容易让人产生更多的兴趣，然后这就成了一种正迭代的过程。

我已经爱上了这种学习模式，如果你让我拿着 ruby 这本书一直看，看到现在，我估计还是啥都做不出来，而这种学习模式让我现在有 10000% 的信心自己可以学好 rails 甚至更多的东西。

总结一句话，用“用”来学习知识可能是学习最好的方式，我会亲自来证明这套模式确实可行。

- August 04, 2016 23:42

- [Permalink](#)

大约 2 个月 ago

8/3 日记

Objective

今天老师讲了这一个星期要完成的事情，clone 一个 jd 最基本的页面。

完成了这个星期的任务。

Reflective

心情还不错，因为基本上没有遇到卡壳的地方。

Interpretive

我们今天学到了什麼？

1. 使用项目管理工具管理网站制作进度
2. 使用 user story 来拆分网站架构
3. 先找出 must have 来做

4.如何使用 carrierwave&minimagic 来上传图片并且在网页中显示。

今天一個重要的領悟是什麼？

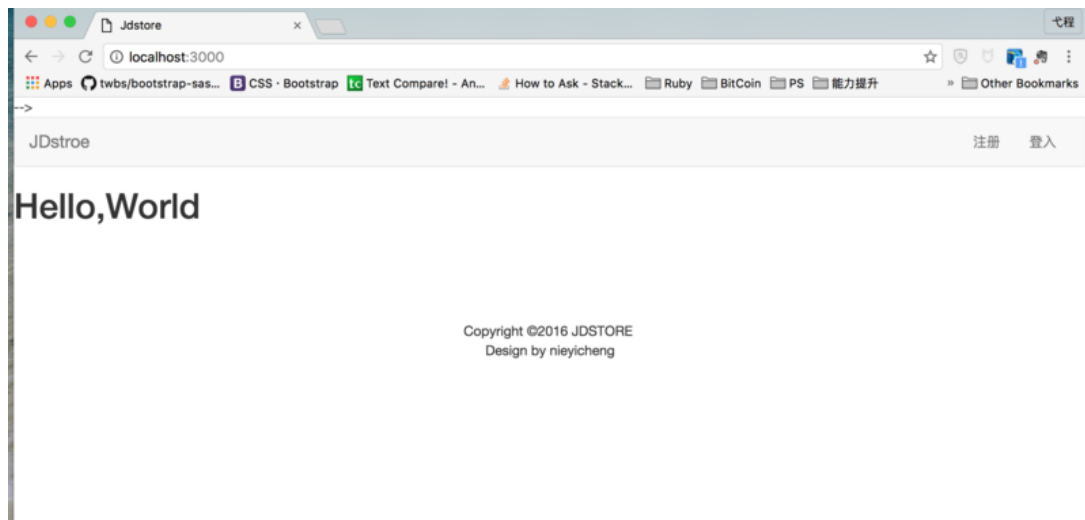
我觉得学习是拼图，并不是搭房子，知道框架以后，需要哪块学哪块，而不是一定要从所谓的“基础”学起。

Decisional

情理之中，意料之内。

Q&A

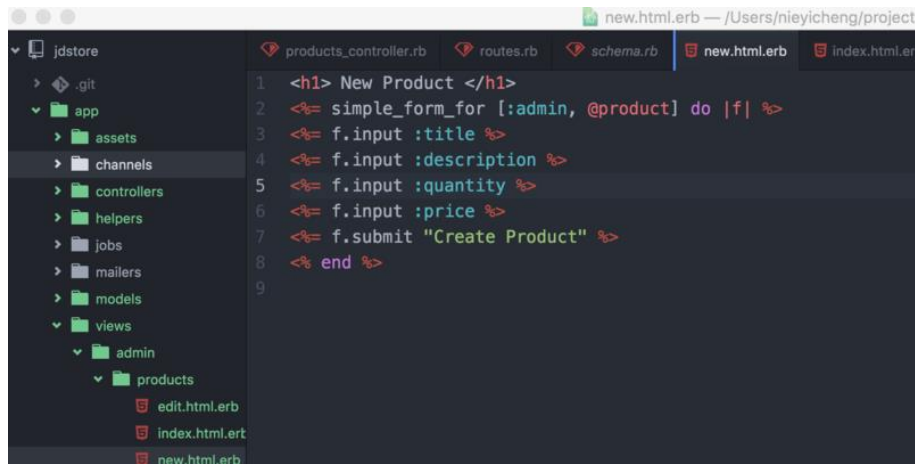
当作到第一步 bootstrap 套框架的时候出现这种问题，导航栏和文章内容全部贴边显示。



后来发现是少了 `div class="container-fluid"` 这个容器 `container-fluid` 的作用就是把东西像包裹住 不让内容贴着显示器显示。

```
<body>
  <div class="container-fluid">
    <%= render "common/navbar" %>

    <%= yield %>
  </div>
  <%= render "common/footer" %>
</body>
</html>
```

simple_form_for [:admin, @product] 这个 admin 是 product 的上级目录

- August 04, 2016 08:31
- [Permalink](#)

大约 2 个月 ago

[8/2 日记](#)

Objective

今天老师讲了 !号与?号的意义与用法，如何传资料给数据库以及特别热血的心灵鸡汤！

今天完成了 step8

Reflective

讲技术知识的时候听不太懂，有点沮丧，讲心灵鸡汤的时候很激动。

Interpretive

我們今天學到了什麼？

1. !current_user (如果 current_user 不存在)

2. current_user != user (不等于)

3 ; !放在后面 我执行这个 method 会改变自己本身 (ruby)

有一个例外 在 rails 里面 user.create => false

加一个!的话 user.create! => ActiveRecord::RecordInvalid

如果表单送不出去也没有错误提示，可以在你的 controller 的 action 里加一个惊叹号 它就会中断告诉你，你的栏位

可能哪边没有填好。

4. ?问号的意思是，我执行下去预期它会回传一个 boolean。

5. 一定要做自己所喜欢的工作，这样效率才会翻倍，不要以为随便找一份清闲的工作然后利用闲暇时间再做会更好。

7. 帮助别人 会让你能以不同角度审视问题，自己的收获会更多

6. 分享，分享会使人快速进步，不要以为分享给别人就是把自己的秘诀告诉别人了，恰恰相反，分享的过程中会梳理你的思路，让你对知识理解的更佳透彻。

7. 有担当，遇到不属于自己的问题的时候，不要完全认为和自己无关，适当的开展自己的技能书，让自己除了专一长之外也变成一个多面手。

今天一個重要的領悟是什麼？

学习不思考等于白学。

Decisional

情理之中，意料之内。

Q&A



做到 step8 的时候犯了一个及其愚蠢的错误，没有认真的看 bug 以为是自己的逻辑和老师不一样，然后缺少一个栏位，但是尝试增加栏位失败，于是求助于助教，结果在错误页面发现了是 created_at 写成了 created_ !!!!!!! what a fuck !!!!!, 犯了两次这种错误了，事不过三，下次遇到错误，不要自以为是，一定要认真把错误页面的每个细节都看清除。

```
erb 8 <td>#</td>
rb 9 <td>Title</td>
erb 10 <td>Description</td>
11 </tr>
html 12 </thead>
erb 13 <tbody>
erb 14 <%= @groups.each do |group| %>|
15 <tr>
erb 16 <td>#</td>
17 <td><%= link_to(group.title, group_path(group)) %></td>
18 <td><%= group.description %></td>
19 <td>
20 <%= link_to("Edit", edit_group_path(group), class: "btn btn-sm btn-
21 <%= link_to("Delete", group_path(group), class: "btn btn-sm btn-
22 method: :delete, data: { confirm: "Are you sure?" } ) %>
23 </td>
24 </tr>
25 <% end %>
26 </tbody>
27 </table>
28 </div>
29
```

代码打错 发现首页出了莫名其妙的数据 助教找了半天才发现，原来 杯来该执行的<% @groups.each do |group| %>打成了 <%= @groups.each do |group| %> 把数据印出来了，我太坑了。

- August 02, 2016 21:31
- [Permalink](#)

大约 2 个月 ago

[8/1 日记](#)

Objective

今天的课程学习了 db migration 的基础知识、遇到问题的解决方法
以及 redirect_to 和 render 的用法 (render 在 contro 和 render 在 views 里的用法)
今天完成了 step6 step7 和资料的查询

Reflective

普普通通直到笑来老师送猪蹄过来。

Interpretive

我們今天學到了什麼？

- 1.db migration 是 Rails 用来操作资料库的 (新增 table,新增栏位...etc)
- 2.资料库栏位无法进行版本控制, 写作容易出事, production 与 local 栏位容易对不齐 (解法:将操作资料库的方法用 Ruby 写进 Git。
- 3.rake db:migrate 可以前进 rake db:roolback 可以后退
- 4.跑完 migrate 刚发现打错字:rake db:rollback (回退一次) 修改完, 再跑 rake db:migrate
- 5.已经过很久才发现打错字:rails g migration fix_xxx_xxx, rename_column :xxxx, :xxxx , rake db:migrate
- 6.忘记加预设值 rails g migration fix_xxxx_xxxx , change_column_default
- 7.全烂了怎么办 db:drop(资料库移除) db:create(资料库建立) db:schema:load(资料库栏位建立) db:seed(建立种子资料)
- 8.render contro 内的用法和 render views 里的用法是不一样的
- 9.scope 的用法
- 10.待续

今天一個重要的領悟是什麼？

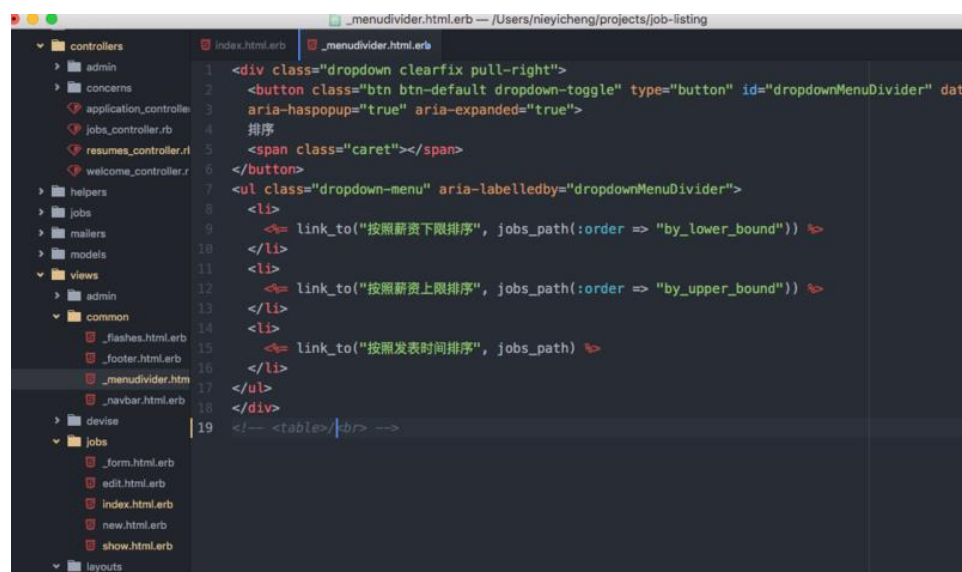
猪蹄好吃！

Decisional

今天遇到很多奇怪 bug,没有自己独立解决很惭愧, 明天尽量自己能手动排除 bug.

Q&A

在做到下拉菜单那一步的时候, 发现菜单无法下拉, 发现在文档底部加入<table>
 发现莫名其妙的就可以下拉



```
1 <div class="dropdown clearfix pull-right">
2   <button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenuDivider" data-
3     aria-haspopup="true" aria-expanded="true">
4     排序
5   <span class="caret"></span>
6 </button>
7 <ul class="dropdown-menu" aria-labelledby="dropdownMenuDivider">
8   <li>
9     <%= link_to("按照薪资下限排序", jobs_path(:order => "by_lower_bound")) %>
10  </li>
11  <li>
12    <%= link_to("按照薪资上限排序", jobs_path(:order => "by_upper_bound")) %>
13  </li>
14  <li>
15    <%= link_to("按照发表时间排序", jobs_path) %>
16  </li>
17 </ul>
18 </div>
19 <!-- <table> <br> -->
```

然后去 jobs 里的 index 里查看表单 发现 div 删除

```
<div class="col-md-12">
  <table class="table table-hover">
    <thead>
      <tr>
        <td class="bg-success">Jobs</td>
        <td class="bg-success">Wage Lower Bound</td>
        <td class="bg-success">Wage Upper Bound</td>
        <td class="bg-success">Email</td>
        <td class="bg-success">Created At</td>
      </tr>
    </thead>
    <tbody>
      <%= @jobs.each do |job| %>
        <tr>
          <td>
            <%= link_to(job.title, job_path(job)) %>
          </td>
          <td><%= job.wage_lower_bound %></td>
          <td><%= job.wage_upper_bound %></td>
          <td><%= job.contact_email %></td>
          <td><%= job.created_at %> </td>
        </tr>
      </tbody>
    </table>
  </div>
```

也可以下啦，后来尝试把 col-md-12 改成了 13 发现可以用，尝试后发现，12 包括 12 以下都没办法把菜单拉下，12 以上就都可以了。后来又尝试在 12 的时候全屏网页，发现菜单也可以下拉，这是一个很神奇的问题。

今天遇到两个大坑，第一个是按照自己的思路做 step7 从属关系少了一个，发现无法 publish,hide 和 create 后来 1 组的学霸帮忙解决了，是少了一组从属关系

第二个大坑是 sign up 和 sign in 无法点击登录，后来又是一组的王同学帮忙解决的，是数据库的问题，关键是我在他处理之前 已经两次 drop、create 以及 migrate 依然没把问题解决，他来也是按照流程走一遍竟然可以了，我擦，这是为啥？

- August 02, 2016 08:14

- [Permalink](#)

大约 2 个月 ago

[在 view 裡面的 render 用法](#)

<%= render "item" %>

展开变成

<%= render :partial => "item" %> 它会吃同一个 controller，贴到别的地方会爆炸

<%= render :partial => "welcome/item" %> 好习惯是写下来 partial 是指哪一个 controller 他会吃

app/views/welcome/_item.html.erb

有底线通常构成大块的某一个部件

下面三个等价

<%= render :collection => @jobs %> 它会自动去寻找下面单数的 job

<%= render :collection => @jobs,

```
:partial => "jobs/job", :as => :job %>
<% @jobs.each do |job| %>
<%= render :partial => "job",
:locals => { :job => job } %>
```

- August 01, 2016 23:38

- [Permalink](#)

大约 2 个月 ago

scope 是什么？怎么用

– 什麼是 rails 中的 scope
scope 的作用就是将经常使用的复杂语法包装一下，下一次只要输入这个包装就能把语法用出来

```
class Topic < ActiveRecord::Base
  scope :recent, -> { order("created_at DESC") }
end
```

上面這段代码定义了 recent 這個 scope，以后我们只要下 recent 这个指令就会呼出 order("created_at DESC").

– 為什麼我們要用 scope

可以使我们的程序代码更简洁。

– 如何使用 scope

沒帶参数的方式

```
class Post < ActiveRecord::Base
  scope :published, -> { where(published: true) }
end
```

帶有的方式

```
class Post < ActiveRecord::Base
  scope :created_before, ->(time) { where("created_at < ?", time) }
end
```

可以串接在一起，顺序没有影响

```
class Event < ActiveRecord::Base
  scope :published, -> { where(published: true) }
  scope :created_before, ->(time) { where("created_at < ?", time) }
end
```

– 参考资料

http://guides.rubyonrails.org/active_record_querying.html

- August 01, 2016 23:11

- [Permalink](#)

大约 2 个月 ago

controller 内 render 的用法

controller 内 render 的用法

render 是将制定页面的样板拿出来，并没有执行 controller

通常 render 的使用时机是让使用者回到同一个页面，例如表单填写不完全时再重回表单填写页，这样做的原因是 render 会传模版给使用者，二这个模版在使用者第一次送出表单时已经被存起来了，所以 render 同一个模版的时候就会保留刚刚使用者打的表单资料，不用全部重打。

- render :text
render 文字。若是把复杂的 code 包在 js 里面的时候，可能会出现单引号双引号打架之类的问题，所用 render 的方式吐出 string 也许是一种方法。
- render :new
就是让使用者回到 new 这个相同的页面，表单填错无法提交也不会清空数据比如说我送到 create 的内容没有成功，但是我希望资料留在表单不动，就可以用 render :new
:new
- render :layout
rails 预设的 layout 是 app/views/layouts/application.html.ebr 这个档案。但有时候我们会希望预设的版型不一样，比方说我们的 admin 页面 head 内不希望加上 GA 和一些有的没的追踪 script。
这时候我们就可以建立一个新的 layout 版形 app/view/layouts/admin.html.erb
只要在 controller 中指定使用 admin layout 即可：
class AdminsController < ApplicationController
 layout "admin"
end
稍微进阶一点的做法：
我们也可以指定某个 action 要使用 admin layout：
class AdminsController < ApplicationController
 layout "admin", :only => :new
 另外也可以在 render 的时候就指定要使用哪一个 layout
 def show
 render :layout => "admin"
 end
 甚至可以指定模板再指定 layout
 def index
 render :template => "others/weired_topics", layout: "admin"
 end
end
- August 01, 2016 22:57
- [Permalink](#)

大约 2 个月 ago

第一周周记

一周感想

上课前老师给了我们课前环境配置、初级、中级教材。

刚看的时候，讲真，完全看不懂啊，打的时候就在想，我他妈在打什么东西啊？虽然我也明白 Professor X 所说的学习理念，但是讲道理，活了这么多年也从没根据这套理念学习过任何东西，一样还好好地活着，虽然活的糙了点。

后来想与其浪费时间纠结不如直接照着做算了，又不会怀孕，然后就一直码代码，一开始做的时候真的什么都不理解，也不知道该怎么查，只能按照老师的要求做，第二遍第三遍的时候发现，我擦，居然能看懂一点了，最起码能看懂我每一步操作在展示什么功能，而且码代码的速度是越来越快，这就给我增加了信心。

头两天上课的时候，老师把一周的任务基本上都安排完了，真是压力山大，突然觉得自己又啥都不会了，然后又慢慢复习，学习新的，看不懂的地方查，查不会纪录下来，继续前进，发现每天都在很神奇的进步，今天能看懂一点这个地方，明天能看懂一点那个地方，所以这种我从来没有用过的学习理论就在我身上逐渐得到证实，在这过程中，无论是我打代码的速度和对代码的理解甚至 debug 的能力都在快速成长，最重要的是我觉得 xdite 老师在刻意的锻炼我们的自学能力，我觉得可能这才是这次笑来老师以及 xdite 老师办全栈工程师班的最终目的。

总之就是：少废话，快学习，多挣钱。

一周所学

- 如何给空页面套上 bootstrap - bootstrap flash - bootstrap dropdown
- 使用的 devise 实现用户注册、登录、登出
- 如何套上 SimpleForm
- HTML 基础知识
- CSS 基础知识
- 使用 annotate gem 查看表结构
- 如何使用 routes 来设置网页访问路径
- mvc 结构
- 如何使用 migration 增加新栏位
- before_action 的使用
- validate 限制条件
- helper 的使用
- 如何让文章按照发表时间排序
- sidebar 的加入
- more and more.
- July 31, 2016 21:43
- [Permalink](#)

这周遇到最大的坑—— migrate

当 rake db:migrate 之后 reset 回之前的 commit 再按照步骤做到 rake db:migrate 会报错，一开始不知道怎么回事，后来反复做了大概三遍，最后求助于助教，才弄明白 要 rake db:drop，但是这个方法会把数据库清空，后来发现有更好的方法就是 rails destroy xxx 可以把特定某一步的数据抵消，然后再一遍 rake db:migrate。

- July 31, 2016 12:53

- [Permalink](#)

2 个月 ago

这周学过最棒的概念——专注力

第一周学到最棒的概念或者工具

专注力

如果说最棒的概念的话那就是听笑来老师说的：

时间不是最宝贵的，最宝贵的是你的专注力。

以前总觉得要节约时间，每天减少睡觉时间，上厕所时间，吃饭时间，但是发现效率并没有办法提高，反而会起到反效果。

做时间日志的时候，发现其实自己正在专注在工作或学习一天最多只有 4-5 个小时，一直尝试着改变，可是并没有什么效果。

当来全栈班上课的第一天，在李老师的演讲中听到这句话，简直如雷灌顶，脑子里从来没有这个概念，但一直知道自己需要这个概念，所以一句话就点透了我。

以前一直觉得要管理时间，学各种各样的时间管理工具，之后看了李老师的书，觉得要与时间作朋友，再之后听了李老师的演讲才明白，我真正需要的是如何有效的提升自己的专注力，而不是去和时间去较劲。

当李老师说很多年轻人都已经废了，因为他们根本不知道珍惜自己的专注力，我也深有体会。

之前有个朋友带我玩竞技游戏，他大概玩了又 7-8 年了，我刚开始接触觉得很有意思，就开始玩了，玩了三个月就把他给秒了，我就在思考，不是说一万小时成就一个天才吗，他这时间算下来也差不多了，为什么会被一个新手干掉呢，难道真的有所谓的智商差距吗？当时并没有结论。

后来听了李老师的演讲，突然就想起了我这个朋友，他打游戏的空闲时间基本上一会看看手机，一会切个网页，然后还控制不了情绪，一输了就乱蹦，而我在空闲时间大脑里一直在思考队伍配置，应对策略以及战术。

缺乏专注力可能就是导致他并没有成为高手的原因，可是对游戏的热爱并没有办法迁移到其它事情上面来。

所以当我遇到喜欢的或者想做的事情，会特别专注，很容易进心流的状态，但是喜欢做的事情很少，要做的事情却很多，所以专注力一直都没有提升。

当读过把时间当作朋友，我开启了自己心智，正在从猴子慢慢进化成人。

当来了全栈班，发现自己很喜欢这种拼搏的生活。

当听了笑来老师的演讲，我才知道，我他妈的以前是浪费了多少专注力。

- July 29, 2016 10:22

- [Permalink](#)

2 个月 ago

7/28 日记

Objective

今天课程预告了下周要做的哪些事情，并解答了一些问题，今天完成了 job-listing 的代码优化和 views 的美化，然后完成了 css。

Reflective

今天状态不错，因为能看懂一点代码了。

Interpretive

- 1.利用 layout 可以帮网站穿不同的衣服
 - 2.html 的注解是<!-- -->
 - 3.当 git commit -m"content" 当注释的信息打错的时候可以用 git commit --amend 来解决
 - 4.当想修改 github 上的 commit 可以使用指令 git push origin -f name 可以强制改写 github 上 commit 的注释
 - 5.每次 bundle install 之后必须得重开服务器
 - 6.toggle 就是切换、开关
 - 7.generate scaffold name title:string description:text
自动生成 CRUD，做网站、API 会吐出 json，不要用。
 - 8.redirect_to :back
rails 会自动记录你上一页访问的地址，所以:back 会执行后退功能。
- 今天最重要的一个领悟是，打代码的时候一定要知道每行代码的基本格式，不然有时候打错都不知道。

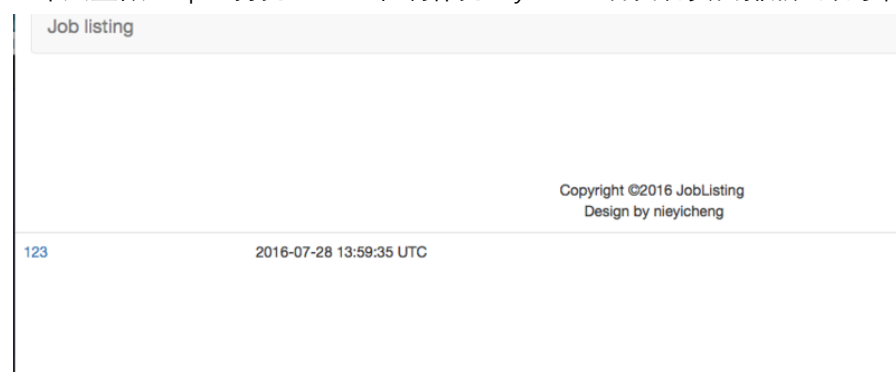
Decisional

轻松、开心。

明天要进一步提升专注力，减少时间浪费。

Q&A

- 1.昨天重做 step1 打完 CURD 和制作完 layout 之后发现页面排版出现了问题



然后弄了好久都没发现出了什么问题，最后同学帮助解决了。。。

原因是 views\Jobs\index.html.erb 里的 table 打成了 tbale。以后再出现这种问题，就要思考一下导致错误有哪些原因，看似有问题的地方并不一定是问题的关键。

```
1 <table class="table table-bordered">
2   <% @jobs.each do |job| %>
3     <tr>
4       <td>
5         <%= link_to(job.title, job_path(job)) %>
6       </td>
7       <td>
8         <%= job.created_at %>
9       </td>
10    </tr>
11  <% end %>
12
13 </table>
14
```

1.<% if flash.any? %> 问号是干嘛的？

2.为什么 model 产生要用单数 controller 要用复数

- July 29, 2016 08:32

- [Permalink](#)

2 个月 ago

CSS 自学作业

1.margin 与 padding 的差异

margin 是指（外边距）

padding 是指（内边距）

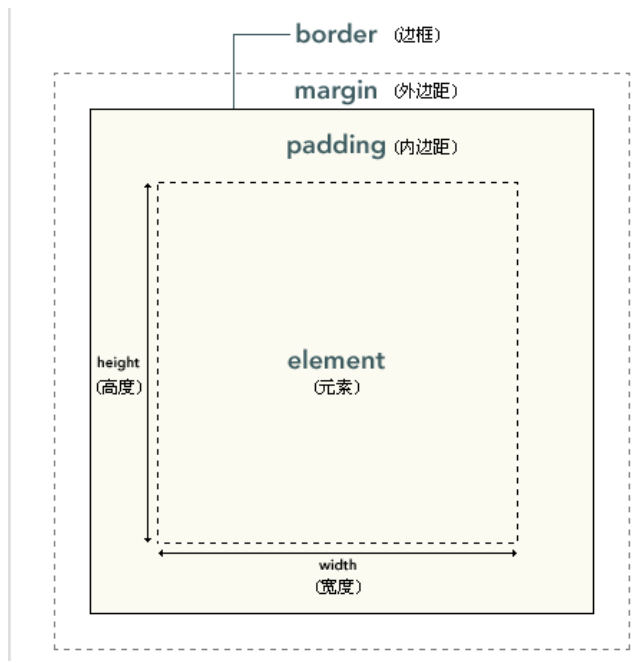
元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

2.什么是 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。
(img)

元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。



- element：元素。
- padding：内边距，也有资料将其翻译为填充。
- border：边框。
- margin：外边距，也有资料将其翻译为空白或空白边。

```
h1 {margin : 10px 0px 15px 5px;}
```

```
margin-top: 10px
```

```
margin-right: 0px
```

```
margin-left : 5px
```

```
margin-bottom: 15px
```

上面的例子为 h1 元素的四个边分别定义了不同的外边距，所使用的长度单位是像素 (px)，这些值的顺序是从上外边距(top)开始围着元素顺时针逆转的：

```
margin: top right bottom left
```

3.为何要使用 em 而非 px 来定义字的大小

如果要避免在 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。

W3C 推荐使用 em 尺寸单位。

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： $\text{pixels}/16=\text{em}$

(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： $\text{pixels}/20=\text{em}$)

```
h1 {font-size:3.75em;} /* 60px/16=3.75em */
```

```
h2 {font-size:2.5em;} /* 40px/16=2.5em */
```

```
p {font-size:0.875em;} /* 14px/16=0.875em */
```

- July 28, 2016 14:26

- [Permalink](#)

2 个月 ago

7/27 日记

Objective

今天的课程主要是郑老师答疑大家的疑问。

今天完成了 step4-5 以及剩下的 html 相关自学。

Reflective

今天休息不太好有点累，学习状态还不错，不过依然有很多不懂的，但是不管三七二十反正学就对了，迟早有一天全部都能弄懂。

interpretive

1.link_to("a",root_path) 和 link_to a, root_path

一样，不过建议使用第一种，link_to 与()之间不能有空格

2.rails 里的 :=> 就是按这个按钮选这个选项的印

3.job = Job.find(params[:id]) 和 @job = Job.find(params[:id]) 的区别是 第一个只在当前文件夹读取

第二个加@是在全局读取。

4.@topics = Topic.all 加 s 的意思是叙述我们捞出来的东西是复数 在 ruby 里第一个字母大写必然指的是这个类，

.all 就是把资料库所有资料全部取出来的意思

5.def index

@topic = Topic.where(:is_hidden => false).order("id DESC")是正方向显示时间 改成 ASC 是逆方向显示

where 的作用就是你不从资料库里捞起所有资料可以用 where 过滤

6.job_params 就是在 views 里面收集资料 打包压缩成 params

7.gem "annotate" (bundle install, annotate) 可以查看文件名类型

8.activerecord 和 applicationrecord 是同样的东西

9.class JobsController < ApplicationController 就是前者继承了后者的特性

10.redirect 是导向某路径，render 是回退，重新显示。

11.rails g devise:views 可以让封装的 devise 显示出来

12.rake db:roll back 回退到上一个数据

13.add_column 就是加栏位 新增任何栏位都必须先加 migrate 里 remove_column rename_column

14.自己新增的 css 可以在 app\stylesheets\name\application.scss 里挂起来

15.col-md-4 col-md-offset-4 屏幕排版和偏移

16.rails model 的作用就是把复杂的电脑语句包装成人能看懂的语句，rails 利用 model 让 ruby 包装一层自动翻译成底层资料。

17.before_filter 通常放在你要限制的地方

18.current_user.admin? = true

if !current_user.admin? => false

惊叹号就是相反的意思，还可以用 unless

也可以写 if current_user.admin? != true(显得像新手)

19.admin_job_path(job)和 admin_job_path(job.id) 是等价的

前者会自动默认是 id.

20.new_admin_job (new a member) 和 admin_jobs_path(a collection)

想象一下你要展示的数据 是一个的还是一群的，一个的就是单数，一群的就是复数。

21.想要增加栏位必须要通过 add_column 去做一个全新的然后再跑 migrate, 在 migrate 里改是没有用的，rails 里面不会回去检查。

22.如何修改北京时间 rails timzone beijing

23.下划线的用法，有的要背，有的是自己定义的。

24.cd ..返回上个目录

今天一个重要的领悟是，在慢慢码代码的时候很多东西慢慢就懂了

Decisional

累，累，累。

努力，努力，努力。

Q&A

1.tag 老是打成 tage 导致程序无数次报错

2.做完所有项目后，测试项目 发现以陌生人身份 查看 show 档案时候会报错，

NoMethodError in Admin::JobsController#show

undefined method `admin?' for nil:NilClass

Extracted source (around line #6):

```
4
5 def require_is_admin
6   if !current_user.admin?
7     flash[:alert] = 'You are not admin'
8     redirect_to root_path
9   end
end
```

Rails.root: /Users/nieyicheng/projects/job-listing

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

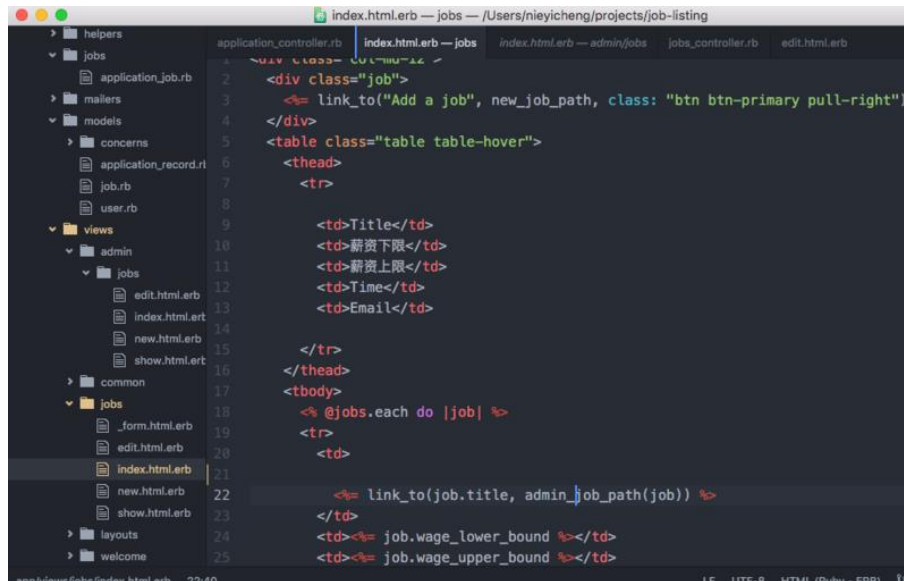
app/controllers/application_controller.rb:6:in `require_is_admin'

Request

Parameters:

{"id"=>"8"}

求助于助教，发现以陌生人身份登录进去的时候网址的链接是 <http://localhost:3000/admin/jobs/8> 本来应该是 <http://localhost:3000/jobs/8>，所以错误的原因可能是 job index 里的路径写错了，查看之后发现果然如此，把 admin_删了之后，变正常了，所以以后做错之后要通过报错页面信息和路径来思考错的在什么地方。



```
1 <div class="job">
2   <%= link_to("Add a job", new_job_path, class: "btn btn-primary pull-right") %>
3 </div>
4
5 <table class="table table-hover">
6   <thead>
7     <tr>
8
9       <td>Title</td>
10      <td>薪资下限</td>
11      <td>薪资上限</td>
12      <td>Time</td>
13      <td>Email</td>
14
15    </tr>
16  </thead>
17  <tbody>
18    <%= @jobs.each do |job| %>
19      <tr>
20        <td>
21          <%= link_to(job.title, admin_job_path(job)) %>
22        </td>
23        <td><%= job.wage_lower_bound %></td>
24        <td><%= job.wage_upper_bound %></td>
25      </tr>
26    </tbody>
27  </table>
```

- July 27, 2016 22:37

- [Permalink](#)

2 个月 ago

Html 自学作业

<div>与的不同

html <div>是块级元素，它是可用于组合其他 HTML 元素的容器。

<div>元素没有什么特定含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。

如果与 CSS 一同使用，<div> 元素可用于对大的内容块设置样式属性。

<div> 元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式方法。使用元素进行文档布局不是表格的正确用法。<table> 元素的作用是显示表格化的数据。

HTML 元素 是内联元素，可用做文本的容器

 元素也没有特定的含义。

<p>for example.another example</p>

如果不对 span 应用样式，那么 span 元素中的文本与其他文本不会有任何视觉上的差异。

当与 css 一同使用时候， 元素可用于为部分文本设置样式属性。

可以对同一个 元素应用 class 或 id 属性。

class 与 id 的不同

class 属于规定元素的类名

class 用于元素组（类似的元素，或者可以理解为某一类元素），而 id 用于标识单独的唯一元素。

class 在 html 中有 ID 类和 CLASS 类。其实就是定义引用的型号。而其中 ID 类是用#来识别，如 #a，当你在制作网页时，可用 ID=a 来应用你之前定义的#a。而 CLASS 类则是用.a，当你在制作网页时，可用 class=a 来应用你之前定义的.a。而 ID 类和 CLASS 类的区别就是，ID 在一个网页中只能被引用一次，而 CLASS 类可以被多次引用。

而这个引用是什么意思呢？其实就是建立一个仓库，里面存放着很多的样式，比如文字颜色，大小等。你在外面要建设网站了，你想把文字设置成红色。但是仓库里有一堆的样式如#a {color:red;(字体颜色为红色)font-size:14px;(字体大小为 14PX)}，如何知道你引用哪一个呢，所以就给每个样式定义了一个编号也就是#a 或.a 中的 a 就是编号。然后在引用时就可以用 ID=a 或 CLASS=a 来引用。

p 与 br 的不同

<p>...</p>是用来定义段落，上一个与下一个会自动换行。(块级元素特征)

是用来在段落内空行的，

如何使用 table 排版

表格由<table>标签来定义。每个表格均有若干行（由<tr>标签来定义），每一行被分割为若干个单元格（由<td>标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。

数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

表格的表头使用<th></th>标签进行定义。大多数浏览器会把表头显示为粗体居中的文本

- <caption>Name</caption>定义表格的标题
- <thead> 标签来定义表格的表头。标签用于组合 HTML 表格的表头内容。
- <tbody>元素用于对 HTML 表格中的主体内容进行分组
- <tfoot>元素用于对 HTML 表格中的表注(页脚)内容进行分组。

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

浏览器会显示如下

Heading Another Heading

row 1, cell 1 row 1, cell 2

row 2, cell 1 row 2, cell 2

- July 27, 2016 14:09
- [Permalink](#)

2 个月 ago

[7/26 日记](#)

Objective

学习了 terminal 基本指令、git 指令、Gemfile 的使用以及好 gem 到哪找、.gitignore 如何隐藏一些关键数据文档、以及 User Story 的思考方法。
完成了：step1-step3 以及 html 的部分自学。

Reflective

今天的情绪，在没有任何参考的情况下做出 step1 的时候很兴奋，做到 step2 的时候发现思路错了，卡了好久。

做 step3 的时候有点懵逼了。

Interpretive

- 1.装了新的 gem 要除了 bundle instal 还要重开服务器
- 2.irb 是即时互动的 ruby
- 3.master 就是主分支
- 4.heroku 是利用 git 部署上传的
- 5.开发习惯 一个窗口开 rails s 第二个窗口 git 第三个 atom
- 6.db/migrate/改了之后要跑 rake db : migrate
- 7.rake routes 可以查看所有路径
- 8.Gemfile 套件都放在这里
- 9..gitignore 可以设定哪一些档案不被 git 存储
- 10.html.erb 即是 HTML 又可以理解 ruby 在说什么，可以写 html 又可以写 ruby
- 11.<% 就是执行
- 12.<%= 就是印出来
- 13.User Story 就是切换到不同的角色让做出的东西更贴近真实
- 14.before_action / before_filter 没有什么区别后者好像快被淘汰了
- 15.当运行 rake db:migrate 出错时 可以尝试
 - rake db:drop
 - rake db:create
 - rake db:migrate 16.rails console 打命令出现错误的时候记得看看数据库里有没有数据 17.出现 if 一定要加 end 错了无数次了 笨。

git

git init (把该目录变成一个被 git 掌握的目录)
git add README (把某个档案加入 git 的控制)
git add . (把当前目录加入 git 的控制)
git commit -m "讯息" 存储到 git。
git remote add origin git://github.com/xxx.git
把远端的“原始来源”设为 github 的端点
git push origin master
本地的主分支叫 master
远端的端点叫 origin
git push -u origin master
以后预设都是推到 origin master，所以以后可以直接 git push
git status
commit 之前 先做一下。

会显示你刚刚变更了哪些档案，绿色就是加到缓存区了，红色是还没有被加进去。

git pull

拉下远端变更

git checkout -b "ch01"

新增一个 branch 叫"ch01"

git checkout "ch01"

切换到 ch01

git checkout --hard 版本号

删除此版本号之前的东西

git branch -m [name] 把当前分支改成 name

git branch -m [name] [name1] 把 name 改成 name1

Decisional

很累很懵逼，不过依然会很努力。

- July 26, 2016 23:44

- [Permalink](#)

2 个月 ago

[7/25 日记](#)

一、关于今天的课程，还记得

1.Computational Thinking

就是电脑解决问题的思维

如何分析问题，如何解决问题，把大问题拆分成小问题去解决。

在做项目的过程中寻找资源

2.编程的世界很孤独

绝大多数时间只有你能救自己

试着不要被“挫折感”淹没

3.如何走问题中走出来

我遇到了什么问题？

想要怎么解决

我希望别人怎么帮助我

如果别人现在无法帮助我的话，我的解决方案

这个问题卡住的话，会让我接下来无法继续下去吗？

4.不做伸手党

太急着发问，没回去重看教程

太急着发问，而不试着自己看“非常明显”的答案

太急着解决，而没意识到现在这个任务，不重要

5.开发时遇到问题怎么办？

深呼吸，有 BUG 是正常，不要慌张

读错误信息，错误信息只是英文，不是外星文。说服自己看得懂。

错误信息怎么阅读

- *Warning*: 只是提醒, 不太会是关键问题
- *Error*: 错误信息 (真正关键的地方) 把 *Error* 的第一行第二行贴到 Google, 看看有没有人已经在 Stackoverflow 问过 通常答案就在上面。

6. 正确在频道上发问

我的 代码 的放置网址: [https://github.com/nieyicheng/...](https://github.com/nieyicheng/)

正在执行什么样的动作: 我目前在“第几章”“第几步”“干嘛时遇到错误”

用的作业系统与 Ruby/Rails 版本

目前浏览器的错误信息截图: (使用 droplr 上传)

目前 rails s 最新 log: (至少贴 100 行, 且贴在 <http://gist.github.com> 上)

到 <https://github.com/xingrowth/fullstack-course/issues> 開一張 issues

贴到频道

二、完成了哪些

完成了第一周第一天的作业

三、如何形容今天的情绪

以前很难理解为什么会有人追星, 见到偶像的那种激动, 今天终于明白了, 哈哈, 激动的不行了。

四、今天学到了什么?

1. 如何利用 Bootstrap 构建网站的全局

2. 如何用 devise 构建会员系统

3. 如何用 simple_form 来替换表单

4. 全栈工程师不是十八班武艺样样精通, 而是能自己完整的做出一套东西, 即使刚做出来的时候很烂

5. 想学好一门技术一定要去研究他的历史, 这绝对是学习最大秘诀。

6. 英文水平决定你的编程发展水平。

7. 人生三大坑 随大溜, 凑热闹, 操别人的心。

8. 不是想尽办法去节省时间, 而是如何在有限的时间内增加自己的专注力。

五、今天最重要的一个领悟

我一定会成为真正的全栈工程师。

六、如何用一句话形容今天的工作

这他妈的才是我想做的。

七、有哪些工作需要明天继续努力?

1. 如何提升更多的专注力

2. 如何让自己的编码的错误更少

3. 如何更好的理解程序里的思维

- July 25, 2016 16:00

[Permalink](#)

2 个月 ago

[Coding 5](#)

九、CH02 step4

手打代码, 打完之后报错, 然后没有认真的看错误页面, 一个一个比对, 花了一个小时 发现 多打了一个括弧, 就在错误代码显示的地方, 太蠢了, 所以下次错误代码一定要认真看,

彻底排查过之后再找其他的问题。

十、在做 CH07 留下的额外作业

给 My Posts 增加 edit 和 delete 功能

先做 delete

先是照葫芦画瓢的在 post controller 加入了

```
def destroy
```

```
@post = Post.find(params[:id])
```

```
@post.destroy
```

```
redirect_to posts_path, alert: "Group deleted"
```

发现报错,然后想起了昨天有人在群里问了,然后我就认真看了一下 发现可以用 rake routes 寻找路由

然后把 posts_path 改成了 account_posts_path 然后就成功了。

第二个增加 edit 功能

翻看了第 3 章和第 5 章 然后摸索出了点门路

首先在 post controller 里加入

```
def edit
```

```
@post = Post.find(params[:id])
```

```
end
```

然后 touch app/views/post/edit.html.erb

接着把 new 的内容改个标题 复制了进去,觉得没啥毛病,就尝试点击 My Posts 里的 Edit,发现报错。

仔细思考了一下,觉得 edit 去找文章不光要找到文章还要找到文章所在的 group

然后在

```
def edit 下 加入了
```

```
@group = Group.find(params[:id])
```

然后测试,发现成功了,哈哈。

第三个,要使 Submit 成功就得加入 update

于是在 post controller 里加入了

```
def update
```

```
@group = Group.find(params[:group_id])
```

```
@post = Post.find(params[:id])
```

```
@post.update(post_params)
```

- July 24, 2016 18:16

[Permalink](#)

2 个月 ago

[Coding 3](#)

六、做到 CH07 可以看到自己发表的文章的然后报错

第一次贴代码的时候就报错,第二次手打代码的时候是好的,然后项目全部做完之后测试的时候报错,

尝试了 reset 回 CH07 重做,依然报错,

NoMethodError in Account::Posts#index

Showing /Users/nieyicheng/rails101-1/app/views/account/posts/index.html.erb where line #19 raised:

undefined method `title' for nil:NilClass

Extracted source (around line #19):

```
17     <tr>
18       <td> <%= post.content %> </td>
19       <td> <%= post.group.title %> </td>
20       <td> <%= post.updated_at %> </td>
21       <td> <%= link_to('Edit', edit_group_post_path(post.group, post), class: "btn btn-default
22         btn-xs") %></td>
```

Rails.root: /Users/nieyicheng/rails101-1

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

app/views/account/posts/index.html.erb:19:in `block in
_app_views_account_posts_index_html_erb_451061469729970785_70191048015440'
app/views/account/posts/index.html.erb:16:in `<%= link_to %>' app/views/account/posts/index.html.erb:451061469729970785_70191048015440'

这个是错误页面，没办法，只能求助于助教。

错误原因肯那个是某笔 post 不是透过 group 建立出来的，所以那笔 post 资料没有 group_id，所以你从 post 去找 group 就会出错 你可以用 Post.all 看一下你所有的 post 资料裡的 group_id 有没有值，

解决方法是:利用 rails console 删除所有数据就可以了。

1.Group.destroy_all Post.destroy_all

ps: Post.first 是查看第一笔 post 资料

Post.first.group 应该是查看 first 和 group 的关联显示 nil 就说明有问题

- July 24, 2016 18:15

[Permalink](#)

Coding 4

七、在使用 HEROKU 上传的时候 无数次上传失败，错误信息是 timeout 就是网络问题 没法上传，开不开 VPN 都没法上传，然后我换了个命令 直接上传最初版本的 master 然后速度非常快，但是因为没有修改 Gemfile 所以没有上传成功，我看上传信息的 source 是 rubygems.org 我就怀疑 是不是 source 的问题 于是我 ch08 换成了 source 然后上传成功了。

八、做到 CH05 的时候出错，然后删除 CH05 重做 当运行 rake db 的时候 报错 显示

StandardError: An error has occurred, this and all later migrations canceled:

SQLite3::SQLException: table "posts" already exists: CREATE TABLE "posts" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "content" text, "group_id" integer, "user_id" integer, "created_at" datetime NOT NULL, "updated_at" datetime NOT NULL)

然后上网查询，运行 rake db:drop:all 然后再运行 rake db:migrate 就可以了。

- July 24, 2016 18:15
- [Permalink](#)

2 个月 ago

Coding 2

三、CH05 实际发表文章

1.step1 posts_controller 忘记打 private 的内容，导致 Write a Post 无效

2.step 4

<%= post.created_at %>

打成了 create 导致 show 页面出错。

四、CH06 建立群成员资料表

step3:设定 Group 与 User 之间的关系

教材出错

class Group < ApplicationRecord

写成了 class User < ApplicationRecord

class 应该与文件名所对应。

五、CH6 实际操作加入群组或突出群组

1.step1 两次忘记 写上 redirect_to group_path(@group)和 end 导致一次程序报错一次退出小组没有刷新提示。这组代码是提前后刷新当前页面的意思。

- July 24, 2016 18:14

[Permalink](#)

2 个月 ago

[Coding 1](#)

一、CH03 限制标题为空的章节

1.忘记在 group.rb 中加入代码

2.groups_controller 中的 def create if else 忘记加入 end

3.models/group.rb ApplicationRecord 忘记改成 Active

4.views/new.html 粗心少加一行

<%= msg %>

导致点击 submit 时候没有显示 Title can't be blank

二、CH04 让这个网站有实际的登入、登出功能

1.打代码的时候走神忘记输入

<%= link_to("登入",new_user_session_path) %>导致页面没有登入按钮

2.登陆页面发现没有下拉登出功能，对比了半天没发现问题，然后逐行对比，发现是把 data 打成了 date 浪费了接近半个小时的时间找问题。

- July 24, 2016 18:12

[Permalink](#)

2 个月 ago

[Tips](#)

1.每次修改完 Gemfile 都要执行一次 bundle install,而且要重开 server

- 因為 gem 不是動態載入 Rails 環境的，只有「你在 app/ 下開發的程式碼」是動態載入，所以當有 app 外的檔案被修改，你都應該「重開」治百病。

2.css 和 scss 有什么不同，sass 是一套程序语言，可以用来产生 css，意思是你可以写程序产生 css，提供一些 css 原声没有办法的进阶作法。(import 是 scss 内引入“模组”的意思)

3.config/routes.rb root 'welcome/index' 可以用来确定主页

(你可以想像成它是全站的「交通警察」，它負責決定使用者的請求「導」到哪邊。所有的網

址生成都歸它指揮。)

4.validates :title, presence: true 這是 Rails 的 Model 內建的「驗證」函式。意思是「不得為空」。

5.<%= render "form" %> 是什麼意思？其實這是 <%= render :partial => "form" %> 的「縮寫」，意思是我要引用 Partial (局部) 頁面的方式。因為我們發現有很大量的程式碼重複，所以可以用這樣的方式收納。

它的規則是同目錄底下的 _xxx.html.erb

6.

- = 是指派。@groups = Group.all, 把 Group.all 數值 指派給 @groups 的意思。
- == 「等於」
- != 「不等於」
- July 24, 2016 18:11
- [Permalink](#)

2 个月 ago

[new page](#)

如何新增页面

1. rails g controller welcome

2. config/routes.rb 新增一行設定：

```
Rails::Application.routes.draw do
  get "welcome/say_hello" => "welcome#say"
  # ...
end
```

get 這一行的意思是將 `http://localhost:3000/welcome/say_hello` 這樣的網址對應到 `welcome Controller` 的 `say Action`。編輯 `app/controllers/welcome_controller.rb`，加入一個 `say` 方法：

```
class WelcomeController < ApplicationController
  def say
  end
end
```

在 `Controller` 中，一個公開函式 (public method) 就代表一個 `Action`，一個 `Action` 對應一個 `HTTP` 的請求和回應。請新增 `app/views/welcome/say.html.erb` 這個檔案，依照慣例目錄名就是 `Controller` 名稱、檔案名是 `Action` 名稱，第一個附檔名說明了這是 `HTML` 格式的檔案，第二個附檔名說明這是 `ERB` 樣板 (我們會在 `View` 一章仔細介紹樣板)。編輯該檔案內容如下：

```
<h1>Hello, World!</h1>
```

讓我們再新增一個頁面並加入超連結。再次編輯路由檔案 `config/routes.rb` 加入一個路由，變成這樣：

```
Rails::Application.routes.draw do
  get "welcome/say_hello" => "welcome#say"
  get "welcome" => "welcome#index"
  # ...
end
```

這一行的意思是將 `http://localhost:3000/welcome` 這樣的網址對應到 `welcome Controller` 的 `index Action`。編輯 `app/controllers/welcome_controller.rb` 加入

```
class WelcomeController < ApplicationController
  #...

  def index
  end
end
```

新增 `app/views/welcome/index.html.erb` 內容是

```
<p>Hola! It's <%= Time.now %></p>
<p><%= link_to 'Hello!', welcome_say_hello_path %></p>
```

`Time` 是 `Ruby` 內建的時間類別，`Time.now` 會輸出目前時間。`link_to` 是 `Rails` 內建的方法可以輸出超連結，而 `welcome_say_hello_path` 會輸出 `/welcome/say_hello` 這個網址。這種出現在 `View` 中的輔助方法統稱作 `Helper`。瀏覽 `http://localhost:3000/welcome`，將看到 `Hola!` 及 `Hello!` 超連結。

- July 24, 2016 18:02

- [Permalink](#)

2 个月 ago

Git

git 指令

git init (把该目录变成一个被 git 掌握的目录)

git init (把该目录变成一个被 git 掌握的目录)

git add .把修改数据存入缓存

git commit -m "Initial Commit" 存储进度并加上注释

git checkout -b [name] 新建分支

git checkout [name] 切换分支

git remote add origin git://github.com/xxx.git

把远端的“原始来源” 设为 github 的端点

git push origin master

本地的主分支叫 master

远端的端点叫 origin

git push -u origin master

以后预设都是推到 origin master, 所以以后可以直接 git push

git status

commit 之前 先做一下。

会显示你刚刚变更了哪些档案, 绿色就是加到缓存区了, 红色是还没有被加进去。

git pull

拉下远端变更

git branch -d [name] -d 选项只能删除已经参与了合并的分支, 对于未有合并的分支是无法删除的。如果想强制删除一个分支, 可以使用-D 选项

合并分支: \$ git merge [name] 将名称为[name]的分支与当前分支合并。

git branch -m [name] 把当前分支改成 name

git branch -m [name] [name1] 把 name 改成 name1

git push origin :heads/[name] 删除远程分支

- July 24, 2016 18:00

- [Permalink](#)

2 个月 ago

Rails

rails 指令

1.rails new 【name】 新建程序

2.rails console 控制台 (Group.create(title: "Board 1", description: "Board 1 body"))

3.rails g controller 【name】 新增一个[name] controller

ps:建立一个新的网页 包括了控制特性等等 还差一个 views 里的 index 可以用 touch 来创建。

4.rails destroy controller [name]

生成控制器时，除了控制器文件本身之外，Rails

还会生成很多其他的文件。撤销生成的文件不仅仅要删除主要的文件，还要删除一些辅助的文件。（事实上，我们还要撤销对 routes.rb 文件自动做的一些改动。）在 Rails 中，我们可以通过 rails destroy 命令完成这些操作。

1. rails g model [name] title:string description:text

- 为了做到在讨论群要有“标题”与“叙述”
- 我们在这里要建立一个 model Group,并建立资料表 group 的而个栏位建立一个 model Group, 并建立资料 group 的二个栏位：1.title(string 字符串属性)
2.description(text 文字属性)

6.rails destroy model [name]

7.rake db:migrate

每次产生 migration 都要跑一遍。

8.rake db:rollback

可以撤销一个迁移操作

9.\$rake db:migrateVERSION=0

将数字 0 换成其他的数字就会回到相应的版本状态，

- July 24, 2016 17:56

- [Permalink](#)

2 个月 ago

Terminal

iterm 指令

ls 罗列当前目录下的内容

cd 更换工作目录

cd ~切换到 home

pwd 显示当前完整工作目录

touch 创建一个文件

mv 移动/更名文件或目录

rm 删除文件活目录

nano 使用 nano 编辑纯文本文件

open 打开一个文件，就好像你在 finder 里双击那个文件图标一样

clear 清空屏幕

sudo 用管理员的身份去执行一个命令

- July 21, 2016 20:48

- [Permalink](#)

3 个月 ago

[Hello World](#)

Hi, This a demo post of [Logdown](#).
Logdown use Markdown as main syntax, you can find more example by reading this [document on Wikipedia](#)
Logdown also support drag & drop image uploading. The picture syntax is like this:



Blogging with code snippet:
inline code

Plain Code

puts "Hello World!"

Code with Language

puts "Hello World!"

Code with Title

hello_world.rb
puts "Hello World!"
MathJax Example

Mathjax

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Inline Mathjax

The answer is $a^2 + b^2 = c^2$

Table Example

Tables	Are	Cool
col 1	Hello	\$1600
col 2	Hello	\$12
col 3	Hello	\$1