基于之前已经成功的divise、form等的blog基础（见git的master目录）

本次大作业实现一个简单的电子商务网站（见git的project目录）。

在开始介绍前，说一下实现的小功能，在作业中实现过的就不说了。

因为之前接触Ionic的时候接触过Angular.js，所以这次使用angular.js还是相对容易些的，这次综合使用devise、bootstrap、angular、jquery都大量用到，以及一些raty.js星级评价库。同时还用到了rails的新特性：asset pipeline来调用js，以及ajax技术。

下面就写重点了，废话不多说。Part0为demo，Part1为功能讲解，Part2为主要的流程，最后附录为笔记，部分笔记丢失了，最后最后附我知道的最简单的ruby+rails的安装方法。  
***Part 0***

...

## Part 1

相关小功能：

1、实现图片平滑放大的效果使用jquery的css的image-zoo属性即可，在product的View模板的show中添加即可。

2、添加用户的星级评价。

首先修改comment部分的设置，添加rating的数值，如db的表格所示。

然后在product的model部分添加

def average\_rating

comments.average(:rating).to\_f

end

然后在相应的view部分show模板添加

<p>Average Rating: <div class="rated" data-score="<%= @product.average\_rating %>"></div> </p>

但是有一点得注意的是，这是得在用户登陆之后才能评价的。因此需要在new\_comment这个模板用户登陆验证。也就是：

<% if signed\_in? %>

1. 想到设计图片的首页有展示效果(相关文档在views/static\_pages)，于是：

(另外注意新建文件夹的名称必须要是复数的，否则无法识别！！！还有，因此效果图片的展示不是在每一个页面都会存在，因此不能放在layouts这个文件夹下，若是放在这个文件夹下则无法进行识别）

具体代码见static\_pages，这部分调用的是bootstrap的carousel的样式。为了也能手工调整，在原来自动滚动图片的基础上添加carousel-control的样式。

但是由于carousel本身的问题，图片只能靠左显示，同时，carousel会自适应图片的大小，这会使得滑动窗口的大小总是在变化，因此得自己设计样式：

style='max-height:600px;max-width:600px;margin:0 auto'

这里想要说的是在layouts文件夹的application.html文件中

<%= yield %>

这一行十分关键！！！

在图片中注明图片名字

<div class="carousel-caption">

<p><%= product.name %></p>

</div>

1. 关于自定义一个order，这里使用angular.js的addOrder和DeleteOrder方法来添加和删除相应的order。
2. 搜索

本控件中为本部分实现搜索的功能，其中:q为在layouts的nav文件中传过来的值。匹配#{search\_term}查找的项即可

def index

if params[:q]

search\_term = params[:q]

@products = Product.where("name LIKE ?", "%#{search\_term}%")

else

@products = Product.all

end

respond\_with @products

end

## Part 2

创建工程，修改gemfile文件

rails new ydd\_shop

gem ‘devise’

gem ‘cancan’

bundle install

rails g cancan:ability

同时先在app/models中创建ability.rb文件

class Ability

include CanCan::Ability

def initialize(user)

end

end

1、首先使用脚架

rails generate scaffold product name:string description:text image\_url:string

rails generate scaffold user first\_name:string last\_name:string

2、接着修改db部分

|  |  |  |  |
| --- | --- | --- | --- |
| **product** | **orders** | **users** | **comments** |
| name | user\_id | first\_name | body |
| description | product\_id | last\_name | rating |
| image\_url | total |  |  |

对应于指令：

rails g migration orders total:float

rails g migration add\_userId\_to\_orders

rails g migration add\_productId\_to\_orders

rails g migration comments body:text rating:integer

rails g migration AddUserToComments user:references

rails g migration AddUserToComments product:references

rails g migration AddAdminFlagToUsers

rails g devise users

最后

rake db:migrate

3、修改model中的文件，增加依附并且添加验证

user.rb添加

has\_many :orders

product.rb添加

has\_many :orders

has\_many :comments

validates :name, presence: true

order.rb添加

belongs\_to :product

belongs\_to :user

comment.rb添加

belongs\_to :user

belongs\_to :product

validates :body, presence: true

validates :user, presence: true

validates :product, presence: true

validates :rating, numericality: { only\_integer: true }

本系统设计为游客的话则为非管理员，若登陆之后则为管理员，因此cancan的ablity文件中的init方法为

user ||= User.new # guest user (not logged in)

can :manage, User, id: user.id

4、修改Controller文件夹

products\_controller添加

respond\_to :json, :html

重要的一点是修改private方法中的product\_params的定义

def product\_params

params.require(:product).permit(:name, :description, :image\_url)

end

有个一直存在的问题导致 form\_for 无法自动处理单数资源。为了解决这个问题，可以直接指定表单的 URL，例如：

form\_for @geocoder, url: geocoder\_path do |f|

## Part 3附录为遇到的问题做的笔记：

1. 打开工程的数据库sqlite3数据库的时候，如果是直接使用命令“sqlite3”，而不是使用"rails dbconsole",则进入的数据库会出错！！！
2. 将代码从自己原来上传到服务器的rails工程下载回来，必须先使用"rake db:migrate"否则会出错！！！

3、rails console 语句产生的irb控制数据库语句后没有";",在属性赋值或者创建属性的时候需要分号,而rails dbconsole 语句产生的sqlite3控制数据库语句后必须要有";"！！但有例外：.table .quit 这些后面都不需要分号

4、

RESTful风格的路由动词默认有7个(分别为：index, show, create, new, edit, update, destroy

respond\_to 对于理解RESTFUL很重要

render 和 redirect 只能用一个

\* redirect\_to(:action=>'xxx') 使用语法和url\_for一样(底层用url\_for)

\* redirect\_to("/localpath")

\* redirect\_to("http://url")

动作控制器通过render方法发送内容给用户,大多数都是提交一个“模板”。视图可以看懂erb的模板，显示出@变量（实例变量）----这种render提交和渲染（把模板呈现为用户看到的视图）过程是自动的，不用你命令render去执行。简单说就是controller中方法里定义的@foo变量，view中rhtml可以用<%=@foo%>显示出来

但动作”的结果是提交不同的“模板”时，手工使用render方法

def search

@results =Search.find(params[:query])

case @results

when 0 then render :action=> "no\_results"

when 1 then render :action=> "show"

when 2..10 then render :action=> "show\_many"

end

end

@blog=Blog.new(params[:blog])

render json:@blog

render action:"new" 对应于文件 “new.html.erb"

render 'form' 对应于文件 ”\_form.html.erb"

render :partial=>'menu'

把名字为\_menu.html.erb的模板渲染到当前模板中

<%= render :partial => "link\_area", :layout => "graybar" %>

partial的layout文件名必须以下划线开头：\_graybar.html.erb，而且必须把layout模板文件和partial放在同一个目录下。

:locals选项用于设置partial的局部变量：

<%= render :partial => "form", :locals => { :button\_label => "Create zone", :zone => @zone } %>

这样就可以在\_form.html.erb中访问button\_label和zone这两个变量。

<%= render :partial => "customer", :object => @new\_customer %>

这样就可以在\_customer.html.erb中访问customer这个变量，它指向@new\_customer。

:collection选项用于指定被传递给partial的集合对象

假设有books这么个集合，包含了5个Book对象，可以这样使用：

#main.html.erb

<%= render :partial => "book", :collection => books %>

#\_book.html.erb

<p><%= book.name%></p>

这 样，在main.html.erb中，\_book.html.erb的内容会被渲染5次。这时候，partial模板中，与partial同名的那个变量 指向了:collection选项传过来的集合中的每一项。如果你不想使用这个与partial同名的变量名，可以通过:as选项来设置你想要的变量名（:as的值只能用symbol，不能是string，否则在partial里会得到nil值）：

<%= render :partial => "product", :collection => @products, :as => :item %>

:spacer\_template

:spacer\_template选项用于指定填充于collection每个member之间的模板：

<%= render :partial => "product", :collection => @products, :spacer\_template => "product\_ruler" %>上面的代码中，\_product\_ruler.html.erb的内容将被填充到每一对\_product partial之间。

和:object一样，:collection也有简写形式： <%= render :partial => @products %>

5、增加两个对象的关系可以有三种方式：

一种是在创建第二个对象的时候建立与第一个对象的关系：

rails g scaffold 第二个对象 第一个对象:references content:text blog\_id:integer

另外一种是

rails g migration AddUserToBlogs user:references

rails g migration AddUserToComments user:references

注意：AddxxxToyyy是convention

在db/migrate中add\_user\_to\_comments.rb为：

class AddUserToComments < ActiveRecord::Migration

def change

add\_reference :comments, :user, index: true, foreign\_key: true

end

end

然后rake db:migrate

最后一种

rails g migration AddRecipeCategoryRelation

class AddRecipeCategoryRelation < ActiveRecord::Migration

def change

add\_reference :recipes, :category

end

end

这之后要使用rake db:migrate来更新数据库

（（（（（rake db:rollback 可以回滚

但两种方法之后 在model 文件夹相应的文件下 都需要类似这样的操作，同时在这个地方完成检验的操作

class Blog < ActiveRecord::Base

validates\_presence\_of :title, :content

has\_many :comments

end

class Comment < ActiveRecord::Base

belongs\_to :blog

end

（（（（（（has\_and\_belongs\_to\_many

最后还别忘了在路由文件中，对资源进行调整，比如：

Blog3::Application.routes.draw do

# resources :comments

resources :blogs do

resources :comments

end

最后的最后rake routes

6、###这是检验操作#####

Validation Helpers（常见验证需求）

acceptance  接受条款

validates\_associated  验证关联数据模型(避免死循环)

confirmation  验证password确认

exclusion  不包含在某个集合中（保留内容）

format  符合特定格式（正则表达）

inclusion  包含于某个集合中（限制输入范围）

length  字符串长度（等于、最大、最小、范围）

numericality  数值数据范围

presence  不为空

absence  当前值为空

uniqueness  值唯一(Username)

比如检查用户名的：：：：

class BlogsController < ApplicationController

before\_action :authenticate, except: [:index, :show]

…

private

def authenticate

authenticate\_or\_request\_with\_http\_basic "My custom message"   
do |user\_name, password|

user\_name == "wty" && password == "wty"

end

end

7、（显示）距离距离创建的时间

<%= time\_ago\_in\_words blog.created\_at %>

8、form\_for form\_tag

<%= form\_for @article, url: {action: "create"}, html: {class: "nifty\_form"} do |f| %>

<%= f.text\_field :title %>

<%= f.text\_area :body, size: "60x12" %>

<%= f.submit "Create" %>

<% end %>

生成如下的HTML：

<form accept-charset="UTF-8" action="/articles/create" method="post" class="nifty\_form">

<input id="article\_title" name="article[title]" type="text" />

<textarea id="article\_body" name="article[body]" cols="60" rows="12"></textarea>

<input name="commit" type="submit" value="Create" />

</form>

<%= form\_tag("/search", method: "get") do %>

<%= label\_tag(:q, "Search for:") %>

<%= text\_field\_tag(:q) %>

<%= submit\_tag("Search") %>

<% end %>

生成如下的HTML：

<form accept-charset="UTF-8" action="/search" method="get"><div style="margin:0;padding:0;display:inline"><input name="utf8" type="hidden" value="&#x2713;" /></div>

<label for="q">Search for:</label>

<input id="q" name="q" type="text" />

<input name="commit" type="submit" value="Search" />

</form>

它们最大的区别和最关键的区别就是input中的name值，分别是：Form\_for是name="article[body]"，而Form\_tag是name="q"，而这个name值决定了数据发往服务器时的格式是怎么的：Form\_for传出去的格式是{article: {body: value}}，而Form\_tag则是{q: value}

然后清楚了它们的两个的区别后，又会衍生另一个问题，就是，那我用Form\_for做搜索不就也可以了？答案是：当然可以。但这里会有一个问题，如果你直接这样使用form\_for @search的话，整个页面就会出错了。原因在于Rails并不知道@search是不是一个hash类型，而且并没有定义里面的参数值，所以你这样定义就可以顺利通过了：@search = {q: ""}。同理可得，其实@article = Article.new就相当是一个给予article默认key值和默认value值的过程。所以其实无论是Form\_for或Form\_tag都好，Rails无形中都帮我们把整个概念整理的更简洁了。

所以在创建和更新方面常见的操作为@article = Article.new(params[:article])，而在search中则多为@article = Article.search(name: params[:name])。

9、helper\_method 可以定义一个在整个工程中都能用的变量名

Rails默认为每个controller指定一个helper，所有的helper都放在app/helpers目录下

有些Helper我们希望是全局共享的，一般我们将这些Helper方法都扔在ApplicationHelper模块里

10、js相关

在view中（.html.erb文件）中用:remote => true触发AJAX调用

Unobtrusive Javascript

用controller中的action充当事件处理器

Rails采用 jquery 库对AJAX调用以及参数的序列化进行封装

可以在控制器方法中识别请求类型（AJAX/普通请求）

respond\_to do |format|

format.js { ... }

在.js.erb视图模板文件中生成javascript代码

步骤：

首先：在Gemfile中 gem 'jquery-ui-rails'

然后：在app/assets/javascripts/application.js中添加

//= require jquery

//= require jquery\_ujs

//= require jquery-ui/effect.all

//= require\_tree .

接着：修改创建comment提交表单的方式（app/views/blogs/show.html.erb）：

<% form\_for [@blog, Comment.new], remote: true do |f| %>

<p>

<%= f.label :content, "New Comment" %> <br />

<%= f.text\_area :content %>

</p>

<p><%= f.submit "Add Comment" %></p>

<% end %>

再接着：修改创建comment的action，处理（app/controllers/comments\_controller.rb）：

def create

@blog = Blog.find(params[:blog\_id])

@comment = @blog.comments.create!(params[:comment])

respond\_to do |format|

format.html { redirect\_to(@blog) }

format.js

end

end

最后：编写基于AJAX的comment创建后页面处理过程（app/views/comments/create.js.erb）：

$('#comments').append('<%= escape\_javascript(render @comment) %>');

$('#comment\_<%= @comment.id %>').effect('highlight', 'slow');

$('#new\_comment')[0].reset();

11、

关于Devise的创建后的东西！

务必注意rails generate devise User后

则Helper\_method 的current\_user\member\_singed\_in?\user\_session不用再定义了

以及session部分不用自己再重新写controller

12、

cancan 希望做到的是，把權限判定的處理部分從 Helper / Controller / View 裡面，全部移到app/models/ability.rb 進行判定

View 只需要判斷是否可以執行動作，而不必問是否有權限

<% if can? :update, @article %>

<%= link\_to "Edit", edit\_article\_path(@article) %>

<% end %>

Controller 不需要手動判斷是否具有權限

class ArticlesController < ApplicationController

authorize\_resource

def show

# @article is already authorized

end

end

但驚人的是 \*\* view 的權限會是與 controller 的權限判定規則 \*\* 卻是一致的。（以往「自刻」權限判定，往往加了 view 卻會忘記 controller, 加了 controller 卻會忘記 view ）

**Cancan 希望做到的：權限中心化管理**

而是否有權限存取，則全交給 app/models/ability.rb 去判斷處理。

class Ability

include CanCan::Ability

def initialize(user)

if user.blank?

# not logged in

cannot :manage, :all

basic\_read\_only

elsif user.has\_role?(:admin)

# admin

can :manage, :all

elsif user.has\_role?(:member)

can :create, Topic

can :update, Topic do |topic|

(topic.user\_id == user.id)

end

can :destroy, Topic do |topic|

(topic.user\_id == user.id)

end

basic\_read\_only

else

# banned or unknown situation

cannot :manage, :all

basic\_read\_only

end

end

protected

def basic\_read\_only

can :read, Topic

can :list, Topic

can :search, Topic

end

end

####查阅无数资料，知道的最简单的安装ruby和rails的方法###############

# ! /usr/bin/sh

# Ubuntu系统下安装ruby/rails必要的库和编译环境

sudo apt-get update

sudo apt-get install -y build-essential openssl curl libcurl3-dev libreadline6 libreadline6-dev git-core zlib1g zlib1g-dev libssl-dev libyaml-dev libxml2-dev libxslt-dev autoconf automake libtool imagemagick libmagickwand-dev libpcre3-dev libsqlite3-dev

# rbenv环境安装

git clone https://github.com/sstephenson/rbenv.git ~/.rbenv

echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc

echo 'eval "$(rbenv init -)"' >> ~/.bashrc

source ~/.bashrc

type rbenv

git clone https://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build

# ruby环境安装，首先列出可安装的版本，然后选择后进行下载编译

rbenv install -l

rbenv install 2.3.0

# 设置当前使用的ruby版本并将gem的源改为淘宝镜像

rbenv global 2.3.0

rbenv rehash

gem sources --remove https://rubygems.org/

gem sources -a https://ruby.taobao.org/

# 安装rails

gem install bundler rails

# 检查安装后的软件版本

ruby -v

gem -v

rake -V

rails -v

# 安装JAVA

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-java6-installer

# 升级

#sudo update-java-alternatives -s java-8-oracle

# 设置JAVA环境变量

sudo apt-get install oracle-java8-set-default

# 设置JAVA字符

sudo mkdir /usr/lib/jvm/java-6-oracle/jre/lib/fonts/fallback

cd /usr/lib/jvm/java-6-oracle/jre/lib/fonts/fallback

sudo ln -s /usr/share/fonts/truetype/wqy/wqy-microhei.ttc wqy-microhei.ttf

sudo ln -s /usr/share/fonts/truetype/wqy/wqy-zenhei.ttc wqy-zenhei.ttf

# 安装node.js（已经包含npm）

sudo add-apt-repository ppa:chris-lea/node.js

sudo apt-get update

sudo apt-get install nodejs