RoR 개념 및 method 정리

해쉬태그(dictionary) variable[:key]=value	b={id:3, status:"I'm fine", zombie:"Jim"} b[:id] → 3 이 나온다.
Table 에서 꺼내 쓰기 t=Tweet.find(3) t.id	Table 은 클래스로 정의되고, instance(한 행)들의 집합이라고 보면 된다. t=Tweet.find(3) : find 안의 숫자는 id 이고, Tweet 는 단수, 대문자 시작으로 class(table)를 의미한다. t 는 id 가 3 인 Tweet table 의 한 instance 이다. t[:id]는 t.id 랑 같은 뜻이고 그 값은 여기서 3 이다. t[:status] 역시 t.status 이고 "I'm fine 이 나옴"
데이터베이스 관련된 CRUD 명령어	
Create 하기! variable=TableName.create(Hash)	t=Tweet.create(status:"I'm fine",zombie:"Zim") 으로하면 새거 하나 만들어짐. Id 는 자동으로 부여되므로 따로 지정 ㄴㄴ
Read 하기! Tweet.find(2,3)	Tweet table 에서 Id 가 2,3 인 instance 를 array 로 배출한다.
Tweet.first Tweet.last	Tweet table 에서 젤 처음거 나옴 Tweet table 에서 젤 마지막꺼 나옴
Tweet.count	Tweet table 에서 갯수를 센다
Tweet.all	Tweet table 전체를 array 로 뽑아논다
Tweet.order(:zombie)	Tweet table 을 zombie attribute 순서로 배열한다
Tweet.limit(10)	Tweet 의 갯수를 앞에서부터 10 개만 뽑는다
Tweet.where(zombie:"ash")	Zombie attribute 의 값이 "ash"인 놈들을 뽑아낸다.
Tweet.limit(10).where(zombie:"ash").count	위의 read 명령어들을 체인처럼 계속 엮어서 명령어낼수있음.
Update 하기! t=Tweet.find(2) t.update={status:"hi", zombie:"hello"}	업데이트하려는 instance 를 잡아서, t.update={key:새로운 값}으로 지정하면 된다
지우기 Tweet.find(3).destroy Tweet.destroy_all	 Tweet 에서 id=3 인 놈을 지운다 Tweet 에서 다 지운다.

MODEL 개념 및 명령어

홈페이지의 DB 를 담당하는 곳.
App/models/tweet(table 이름).rb 에 저장한다.

class Tweet < ActiveRecord::Base #여기에 작성 end	Table 의 이름이 대문자,단수로 class 이름으로 들어간다.(Tweet)
validates: status(원하는 attribute), presence:true, length:{minimum:3, maximum:20}, uniqueness:true (원하는 조건들 작성)	원하는 attribute 가 원하는 조건을 만족해야지만, DB 가 생성된다. presence: 존재 numericality: 숫지안지 uniqueness: 유일한지 confirmation: PW 같은게 서로 같은지 acceptance : 동의서 체크했는지 length: {maximum:20} 길이 조건 만족하는지 format: with:/regex/i 원하는 형식인지 inclusion : in:21..99 원하는 범위에 들어가는지
Table 관계짓기 zombie.rb 에선 has_many: tweets tweet.rb 에선 belong_to: zombie	두 table 간에 관계가 있다면 즉 Zombie 가 여러개 tweet 을 쓴다고 할때 해야할일. Zombies table 의 id(zombie_id)가 tweets table 의 한 attribute 로 들어가있는 것으로 표현한다. 이경우 zombie 1 마리가 여러개 트윗을 남기므로, zombie.rb 에서 has_many:tweets (복수로 쓴다) 를 추가, tweet.rb 에서 belong_to:zombie (단수로 쓴다)를 추가해줘야함. 물론 디비에서도 zombie_id 가 tweet 의 attribute 에추가되어 있어야 함.
관계 찾기 ash=Zombie.find(1) Tweet.create(status:"EYE", zombie: ash) ash.tweets Tweet.find(4).zombie	 생성할때는 zombie 의 instance 를 바로 tweet 의 zombie attribute 로 넣어준다. zombie ash 가 쓴 tweet 을 모두 가져온다 id=4 인 tweet 을 쓴 zombie 를 찾아준다.

VIEW 개념 및 명령어

페이지를 표시해주는 놈
app/views/tweets/show.html.erb 에 기본을 저장하고
app/views/layouts/application.html.erb 에 반복해서 쓰는 기본 형식을 저장한다

application.html.erb <!DOCTYPE html> <html> <head> … </head> <body> <%= yield %> </body> </html>	이렇게 head 등 반복되는 기본 포멧을 application.html.erb 에 저장해둔다. <%= yield %>에는 다른 html.erb 의 body 내용들이 들어간다.
tweets/show.html.erb	<% tweet=Tweet.find(1)%> <h1><%= tweet.status %></h1> <p> Posted by <%=tweet.zombie.name%> <%= link_to tweet.zombie.name, tweet.zombie %> 예시. 이렇게 body 의 내용만이 들어감.
<% %>	Ruby 보고 해석하는 신호(VIEW 파일에서 쓰임)
<%= %>	Ruby 보고 해석하고 html 로 남기라는 신호 (VIEW 파일에서)
<% tweet=Tweet.find(1) %>	Tweet table 의 id 1 인 놈을 tweet 변수에 저장하는 뜻 근데 나중에 보면 알겠지만, 데이터 로딩은 VIEW 에서 안하고 controller 에서 한다.
<%= @tweet.zombie.name % >	tweet 을 쓴 좀비의 이름을 html 로 남기라는 뜻. @를 변수명앞에 써주는 건, 변수가 VIEW 가 아닌 컨트롤러에서 정의된 변수라는 것. 즉 컨트롤러에서 가져온 변수를 VIEW 에서 쓸때 이렇게 쓴다.
<%= link_to text_to_show, url_code %>	Html 의 a 랑 비슷한 태그. Text_to_show 란 곳에 링크가 걸릴 글자를 쓰고, url_code 에 가고싶은 url 이나 urlcode 를 쓴다. Ex) <% link_to @tweet.zombie.name, @tweet.zombie %>
<%= link_to text_to_show, url_code , confirm:"R U sure?"%>	위랑 똑같고 다만 클릭시 바로 이동하기 전에 "R U sure?" 한번더 물어본다.
<%= link_to "EDIT",edit_tweet_path(tweet) %>	EDIT 클릭시 tweet 수정링크로 연결한다.
<%= link_to "Destroy",tweet, method: delete %>	DESTROY 클릭시 tweet 을 지운다.
<% Tweet.all.each do |tweet| %>	Tweet.all 에 있는 것들을 하나씩 꺼내서 tweet 변수에 넣고, …을 시행하라.

<table>
<tr><td>…
<% end %></td><td></td></tr>
<tr><td><% if 조건 %>
…
<% end %></td><td>If 문.
<% if Tweet.all.size==0%>
..
<% end %></td></tr>
<tr><td>URL generator method</td><td>routes 에 resources:tweets 문장을 추가하면
미리 만들어진 5 가지 url 코드들을 추가 설정없이 쓸 수 있다.

<table>
<tr><th>Code
(link_to 의 url_code)</th><th>실제 url</th></tr>
<tr><td>tweets_path</td><td>/tweets</td></tr>
<tr><td>new_tweet_path</td><td>/tweets/new</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>tweet</td><td>/tweet/1(1 에는 tweet 의 id 가 들어간다)</td></tr>
<tr><td>edit_tweet_path(tweet)</td><td>/tweet/1/edit</td></tr>
<tr><td>tweet, method: :delete</td><td>/tweet/1</td></tr>
<tr><td colspan="2">(아래 3 개의 tweet=Tweet.find(1)인 경우.)
위 표의 code 가 <%= link_to “적절단어”, url_code %>에서 url_code 에 들어가지 음음!</td></tr>
</table>
</td></tr>
</table>

Controller 개념 및 명령어

Model, view 를 컨트롤하는 브레인!
View 에서는 data 를 로딩하지 않는다고 보면됨. 컨트롤에서 하지.
app/controllers/tweets_controller.rb 에서 수정!

class TweetsController < ApplicationController
  def show
  @tweet= Tweet.find(params[:id])
  render action:'status'
  end
end
	일단 파일명인 tweets_controller 를 보자. tweets 과 똑같은 app/view/tweets 폴더에 들어간다.
	
	그다음에 show 함수이므로 추가 명령이 없으면 함수명인 (app/view/tweets 폴더 안에 있는) show.html.erb 에 들어간다.
	  즉 app/view/tweets/show.html.erb 파일을 키는 거지.

render action:'status'	만약 옆 코드처럼 render action: 'status'가 들어가면, 함수명인 show 대신 status.html.erb 가 켜진다. Render action:'status' 명령어가 없으면 함수명인 show.html.erb 가 켜지고
@tweet= Tweet.find(params[:id])	@tweet=…은 변수를 설정하는건데 변수명앞에 @가 있는건 이 변수를 view 에서도 쓰겠다는 뜻! params[:id]는 params 에 주소(url) 안에 변수가 해쉬태그로 들어오는데, key 가 id 인 값을 넣으라는 뜻.
STRONG PARAMETERS @tweet=Tweet.create(params.require(:tweet).permit(:status)) ~~~.permit([:status, :location])	주로 create, update 할때 해킹같은걸 막기 위해 strong parameter 를 사용. require(필요한 key 값).permit(쓰고싶은 attribute 입력) url 이 /tweet?tweet[status]="I'mdead"라고 할때 params={tweet:{status:"I'mdead"}}이 된다. require 에서 tweet key 의 value 를 받아오고 (여기선 {status:"I'mdead}) 거기서 permit 에서 정의한 status 를 가져온다. 여러개 attribute 를 가져오려면 옆처럼!
respond_to do |format| format.html format.json {render json:@tweet}	json 에 정보를 주고싶다!(API 등) 이렇게 하면 json 인 경우 json 으로 html 인 경우 view 로 간다.
Controller 에서 권한 설정하기! 각 함수에 권한을 설정해보자!	

def edit @tweet=Tweet.find(params[:id]) if session[:zombie_id]!=@tweet.zombie_id flash[:notice]=”Sorry. You can’t do this” redirect_to(tweets_path) end end	글쓴이 id 랑 tweet 쓴사람 id 랑 일치 안할 경우 (session 은 유저마다 가지고 있는 값을 저장하는 곳)
flash[:notice]=”Sorry. You can’t do this”	변수에 저장하고
redirect_to(tweets_path)	다시 tweets_path 로보낸다. 경고까지 주고 싶으면 redirect_to(tweets_path, notice:”sorry”)
Application.html.erb 로 돌아가서 <% if flash[:notice] %> <%= flash[:notice] %> <% end %> <%= yield%>	flash 변수에 저장되어있으면 변수를 써준다.
중복을 최소화하기 Class TweetsController… before_action: get_tweet, only: [:edit, :update, :destroy] def get_tweet @ tweet=Tweet.find(params[:id] before_action: 함수명, only: [:적용하고싶은 함수들]	자꾸 반복되는 코드는 함수를 만들어서 앞으로 빼주자. edit, update, destroy 함수전에 get_tweet 를 한다.

route 개념 및 명령어

URL 등을 보고 어느 controller 에 어떤 action 을 쓸지 정해주는 놈
/config/routes.rb

ZombieTwitter:: Application.routes.draw do resources : tweets end	->요걸 쓰면 tweets 에 대한 URL generator method 를 추가 설정없이 쓸 수 있다.
get '/new_tweet' => 'tweets#new', as :'real_new_tweets' get "주소"=>"컨트롤러#액션", as:"url_code 이름"	주소 끝에 /new_tweet'이 오면 tweets 컨트롤러에 new 라는 action 을 시행해라. 그리고 이 주소를'real_new_tweets'라는 url_code 로 저장해라. 이 url_code 는 link_to 함수등에서 변수로 쓸수있지 ㅇㅇ.
get '/all'=>redirect('/tweets')	/all 이 오면 /tweets 로 다시 이동시켜라. 주소가 여러개면 복잡하니깐 하나로 깔끔하게 통일하고 싶을때 쓰자!
root to: "tweets#index" root to:"컨트롤러#액션"	기본주소. 뒤에 아무것도 안쳤을때 나오는 주소. url_code 는 root_path 이다.
Route parameter	url 안에 변수가 들어간다면?
route 수정 get 'local_tweets/:zipcode' => 'tweets#index', as 'local_tweets'	local_tweets/12321 을 치면 params[:zipcode]에 12321 이 저장됨. 그리고 tweets#index 를 키는거지. url_code 는 local_tweets 로 저장했는데 나중에 local_tweets(12321)등으로 부를 수 있음. 그담엔 tweets controller 를 적절히 수정해야겠지?

tweets_controller 수정

```
def index
 if params[:zipcode]
  @tweets=Tweet.where(zipcode:params[:zipcode])
 else
   @tweets=Tweet.all
 end
 ….
End
```

params[:zipcode]가 있으면 그걸 value 로 가지고 있는 tweet 만 추출, 없으면 걍 다 추출.