RAILS BEST PRACTICES

SUPER DELUXE CHAMPIONSHIP EDITION

CONTROLLERS IN SPACE

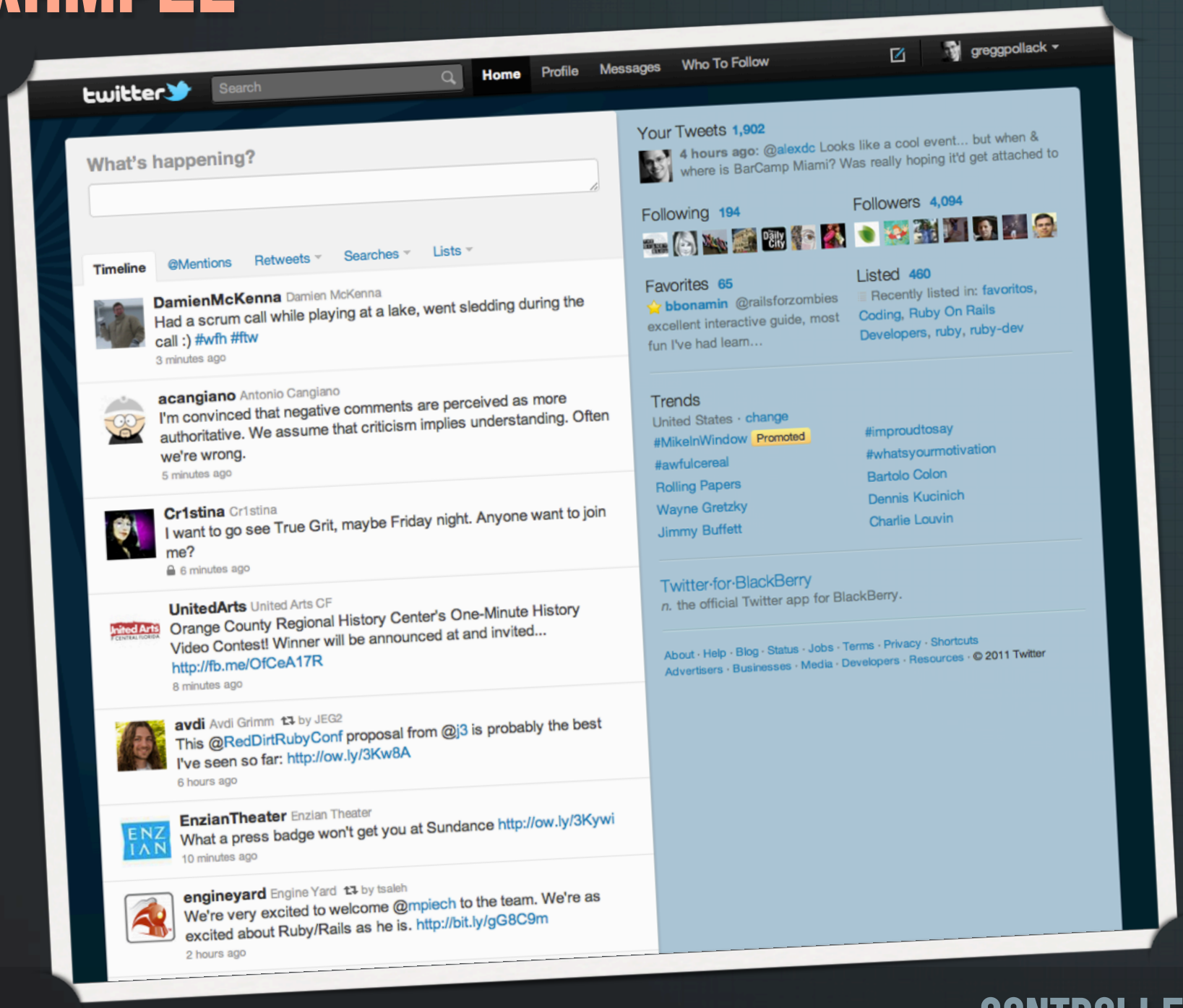
CONTROLLERS IN SPACE

LEVEL 1

SAD CODE

HAPPY CODE

EXAMPLE


twitter
Search
Home
Profile
Messages
Who To Follow
greggpollack
What's happening?
Timeline
@Mentions
Retweets
Searches
Lists
DamienMcKenna Damien McKenna
Had a scrum call while playing at a lake, went sledding during the call :) #wfh #ftw
3 minutes ago
acangiano Antonio Cangiano
I'm convinced that negative comments are perceived as more authoritative. We assume that criticism implies understanding. Often we're wrong.
5 minutes ago
Cr1stina Cr1stina
I want to go see True Grit, maybe Friday night. Anyone want to join me?
6 minutes ago
UnitedArts United Arts CF
Orange County Regional History Center's One-Minute History Video Contest! Winner will be announced at and invited... http://fb.me/OfCeA17R
8 minutes ago
avdi Avdi Grimm by JEG2
This @RedDirtRubyConf proposal from @j3 is probably the best I've seen so far: http://ow.ly/3Kw8A
6 hours ago
EnzianTheater Enzian Theater
What a press badge won't get you at Sundance http://ow.ly/3Kywi
10 minutes ago
engineyard Engine Yard by tsaleh
We're very excited to welcome @mpiech to the team. We're as excited about Ruby/Rails as he is. http://bit.ly/gG8C9m
2 hours ago
Your Tweets 1,902
4 hours ago: @alexdc Looks like a cool event... but when & where is BarCamp Miami? Was really hoping it'd get attached to
Following 194
Followers 4,094
Favorites 65
bbonamin @railsforzombies excellent interactive guide, most fun I've had learn...
Listed 460
Recently listed in: favoritos, Coding, Ruby On Rails Developers, ruby, ruby-dev
Trends
United States · change
#MikeInWindow Promoted
#awfulcereal
Rolling Papers
Wayne Gretzky
Jimmy Buffett
#improudtosay
#whatsyourmotivation
Bartolo Colon
Dennis Kucinich
Charlie Louvin
Twitter for BlackBerry
n. the official Twitter app for BlackBerry.
About · Help · Blog · Status · Jobs · Terms · Privacy · Shortcuts
Advertisers · Businesses · Media · Developers · Resources · © 2011 Twitter

FAT MODEL, SKINNY CONTROLLER

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  def retweet
    tweet = Tweet.find(params[:id])

    if tweet.user == current_user
      flash[:notice] = "Sorry, you can't retweet your own tweets"
    elsif tweet.retweets.where(:user_id => current_user.id).present?
      flash[:notice] = "You already retweeted!"
    else
      t = Tweet.new
      t.status = "RT #{tweet.user.name}: #{tweet.status}"
      t.original_tweet = tweet
      t.user = current_user
      t.save
      flash[:notice] = "Succesfully retweeted"
    end

    redirect_to tweet
  end
end
```

FAT MODEL, SKINNY CONTROLLER

*/app/controllers/**tweets_controller.rb***

```ruby
class TweetsController < ApplicationController
  def retweet
    tweet = Tweet.find(params[:id])
    flash[:notice] = tweet.retweet_by(current_user)
    redirect_to tweet
  end
end
```

*/app/models/**tweet.rb***

```ruby
class Tweet < ActiveRecord::Base
  def retweet_by(retweeter)
    if self.user == retweeter
      "Sorry, you can't retweet your own tweets"
    elsif self.retweets.where(:user_id => retweeter.id).present?
      "You already retweeted!"
    else
      ...
      "Succesfully retweeted"
    end
  end
end
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```
def index
  @tweets = Tweet.find(
              :all,
              :conditions => {:user_id => current_user.id},
              :order => 'created_at desc',
              :limit => 10
            )
  @trending = Topic.find(
                :all,
                :conditions => ["started_trending > ?", 1.day.ago],
                :order => 'mentions desc',
                :limit => 5
              )
  ...
end
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```
def index
  @tweets = Tweet.where(:user_id => current_user.id).
                  order('created_at desc').
                  limit(10)



  @trending = Topic.where('started_trending > ?', 1.day.ago).
                    order('mentions desc').
                    limit(5)



  ...
end
```

SCOPE IT OUT

*/app/controllers/**tweets_cont***

SCOPE TO THE USER

```
def index
  @tweets = current_user.tweets.
                order('created_at desc').
                limit(10)

  ...
end
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @tweets = current_user.tweets.recent.limit(10)
  ...
end
```

*/app/models/**tweet.rb***

```ruby
class Tweet < ActiveRecord::Base
  scope :recent, order('created_at desc')
  ...
end
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @tweets = current_user.tweets.limit(10)
  ...
end
```

*/app/models/**tweet.rb***

```ruby
class Tweet < ActiveRecord::Base
  default_scope order('created_at desc')
  ...
end
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @trending = Topic.trending.limit(5)
  ...
end
```

*/app/models/**topic.rb***

WILL ONLY WORK ONCE

```ruby
class Topic < ActiveRecord::Base
  scope :trending, where('started_trending > ?', 1.day.ago).
  ...              order('mentions desc')
end
```

SAME TIME!

1 `where('started_trending > ?', '12-01-2010 14:02')`

2 `where('started_trending > ?', '12-01-2010 14:02')`

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```
def index
  @trending = Topic.trending.limit(5)
  ...
end
```

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base
  scope :trending, lambda { where('started_trending > ?', 1.day.ago).
  ...                       order('mentions desc') }
end
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @trending = Topic.trending(5)
  ...
end
```

*/app/models/**topic.rb***

```ruby
class Topic < ActiveRecord::Base
  scope :trending, lambda { |num| where('started_trending > ?', 1.day.ago).
                                  order('mentions desc').
                                  limit(num) }

  ...
end
```

WRONG NUMBER OF ARGS, 0 FOR 1

```ruby
@trending = Topic.trending(5)
```

```ruby
@trending = Topic.trending
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @trending = Topic.trending(5)
  ...
end
```

*/app/models/**topic.rb***

RUBY 1.9 FTW!

```ruby
class Topic < ActiveRecord::Base
  scope :trending, lambda { |num = nil| where('started_trending > ?', 1.day.ago).
                                        order('mentions desc').
                                        limit(num) }

  ...
end
```

```ruby
@trending = Topic.trending(5)
```

```ruby
@trending = Topic.trending
```

SCOPE IT OUT

*/app/models/**tweet.rb***

```
class Tweet < ActiveRecord::Base
  default_scope order('created_at desc')
  ...
end
```

HOW DO WE OVERRIDE DEFAULT SCOPE?

```
@tweets = current_user.tweets.order(:status).limit(10)
```

```
@tweets = current_user.tweets.unscoped.order(:status).limit(10)
```

SCOPE IT OUT

*/app/controllers/**tweets_controller.rb***

```
t = Tweet.new
t.status = "RT #{@tweet.user.name}: #{@tweet.status}"
t.original_tweet = @tweet
t.user = current_user
t.save
```

CURRENT_USER HAS MANY TWEETS....

```
current_user.tweets.create(
      :status => "RT #{@tweet.user.name}: #{@tweet.status}",
      :original_tweet => @tweet
      )
```

FANTASTIC FILTERS

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  def edit
    @tweet = Tweet.find(params[:id])
    ...
  end

  def update
    @tweet = Tweet.find(params[:id])
    ...
  end

  def destroy
    @tweet = Tweet.find(params[:id])
    ...
  end
end
```

FANTASTIC FILTERS

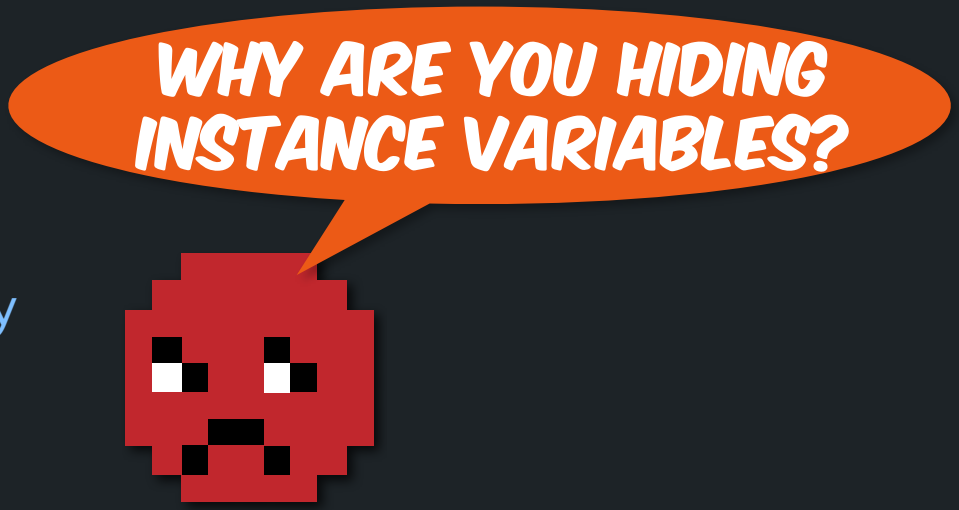
*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  before_filter :get_tweet, :only => [:edit, :update, :destroy]

  def get_tweet
    @tweet = Tweet.find(params[:id])
  end

  def edit
    ...
  end

  def update
    ...
  end

  def destroy
    ...
  end
end
```

FANTASTIC FILTERS

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  before_filter :get_tweet, :only => [:edit, :update, :destroy]

  def edit
    ...
  end

  def update
    ...
  end

  def destroy
    ...
  end

  private
  def get_tweet
    @tweet = Tweet.find(params[:id])
  end
end
```



FANTASTIC FILTERS

Keeping parameters in actions

```
class TweetsController < ApplicationController

  def edit
    @tweet = get_tweet(params[:id])
  end

  def update
    @tweet = get_tweet(params[:id])
  end

  def destroy
    @tweet = get_tweet(params[:id])
  end

  private
  def get_tweet(tweet_id)
    Tweet.find(tweet_id)
  end
end
```

FANTASTIC FILTERS

WHAT SHOULD THEY BE USED FOR?

- AUTHORIZATION
- LOGGING
- WIZARDS

FANTASTIC FILTERS

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  before_filter :auth, :only => [:edit, :update, :destroy]
```

```
                        :except => [:index, :create]
```

Global Filters

```
class ApplicationController < ActionController::Base
  before_filter :require_login
```

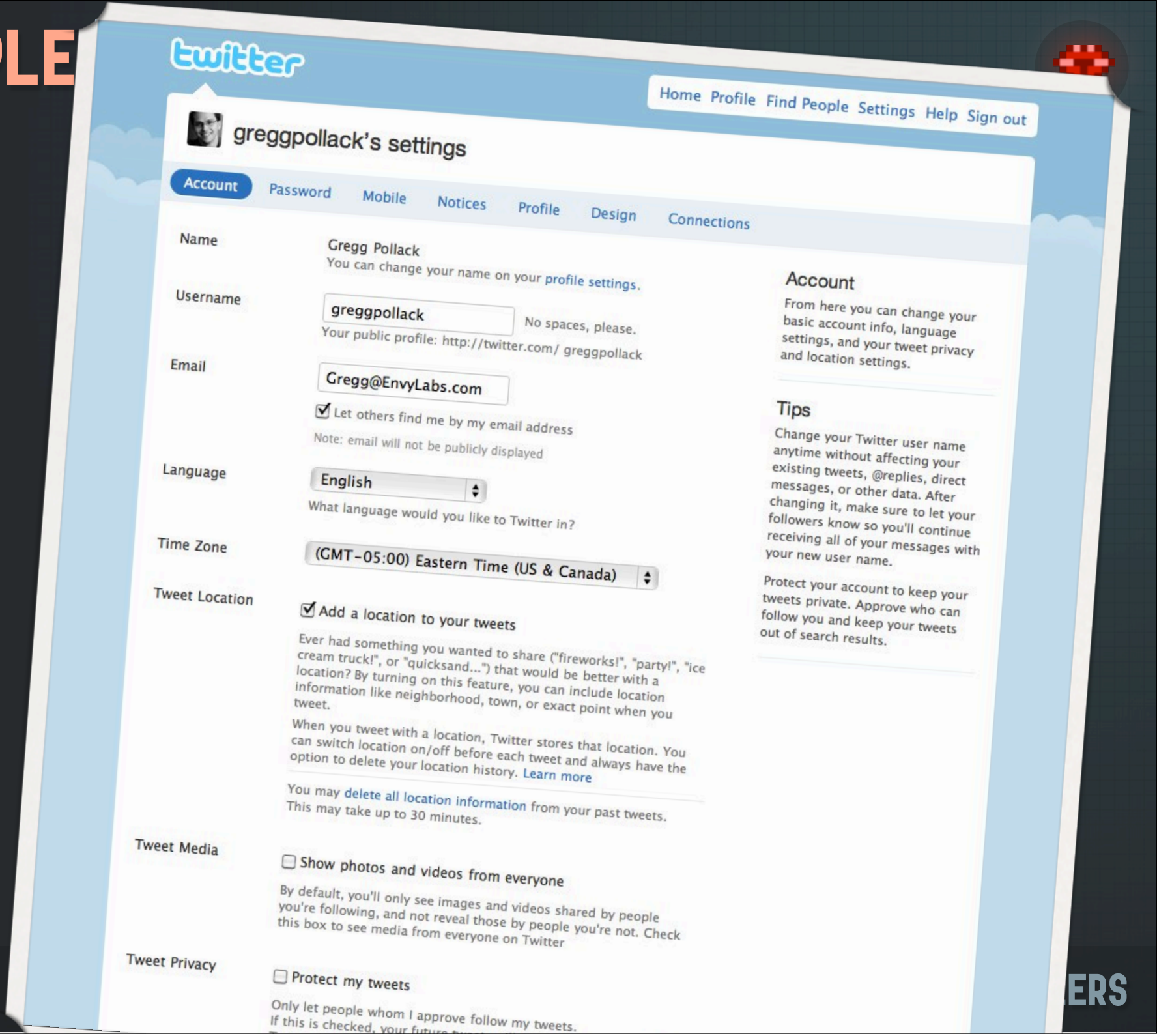
BUT WHAT ABOUT THE LOGIN PAGE ITSELF?

```
class SessionsController < ApplicationController
  skip_before_filter :require_login, :only => [:new, :create]
```

CONTROLLER COMMAND

LEVEL 2

EXAMPLE


twitter
Home Profile Find People Settings Help Sign out
greggpollack's settings
Account Password Mobile Notices Profile Design Connections
Name
Gregg Pollack
You can change your name on your profile settings.
Username
greggpollack
No spaces, please.
Your public profile: http://twitter.com/ greggpollack
Email
Gregg@EnvyLabs.com
Let others find me by my email address
Note: email will not be publicly displayed
Language
English
What language would you like to Twitter in?
Time Zone
(GMT-05:00) Eastern Time (US & Canada)
Tweet Location
Add a location to your tweets
Ever had something you wanted to share ("fireworks!", "party!", "ice cream truck!", or "quicksand...") that would be better with a location? By turning on this feature, you can include location information like neighborhood, town, or exact point when you tweet.
When you tweet with a location, Twitter stores that location. You can switch location on/off before each tweet and always have the option to delete your location history. Learn more
You may delete all location information from your past tweets. This may take up to 30 minutes.
Tweet Media
Show photos and videos from everyone
By default, you'll only see images and videos shared by people you're following, and not reveal those by people you're not. Check this box to see media from everyone on Twitter
Tweet Privacy
Protect my tweets
Only let people whom I approve follow my tweets.
If this is checked, your future
Account
From here you can change your basic account info, language settings, and your tweet privacy and location settings.
Tips
Change your Twitter user name anytime without affecting your existing tweets, @replies, direct messages, or other data. After changing it, make sure to let your followers know so you'll continue receiving all of your messages with your new user name.
Protect your account to keep your tweets private. Approve who can follow you and keep your tweets out of search results.

NESTED ATTRIBUTES

*/app/models/**user.rb***

```ruby
class User < ActiveRecord::Base
  has_one :account_setting, :dependent => :destroy
end
```

*/app/views/users/**edit.html.erb***

```erb
    <%= fields_for :account_setting do |a| %>
      <div class="field">
        <%= a.label :public_email %><br />
        <%= a.check_box :public_email %>
      </div>
      <div class="field">
        <%= a.label :show_media %><br />
        <%= a.check_box :show_media %>
      </div>
      <div class="field">
        <%= a.label :protect_tweets %><br />
        <%= a.check_box :protect_tweets %>
      </div>
    <% end %>
```

NESTED ATTRIBUTES

*/app/controllers/**users_controller.rb***

```
class UsersController < ApplicationController
  def create
    @user = User.new(params[:user])
    @account_setting = AccountSetting.new(params[:account_setting])

    if @user.save
      @account_setting.user = @user
      @account_setting.save
      redirect_to(@user, :notice => 'User was successfully created.')
    else
      render :action => "new"
    end
  end
end
```

NESTED ATTRIBUTES

*/app/controllers/**users_controller.rb***

using Nested Attributes

```
class UsersController < ApplicationController
  def create
    @user = User.new(params[:user])
    if @user.save
      redirect_to(@user, :notice => 'User was successfully created.')
    else
      render :action => "new"
    end
  end
end
```

NESTED ATTRIBUTES

*/app/models/**user.rb***

```ruby
class User < ActiveRecord::Base
  has_one :account_setting, :dependent => :destroy
  accepts_nested_attributes_for :account_setting
end
```

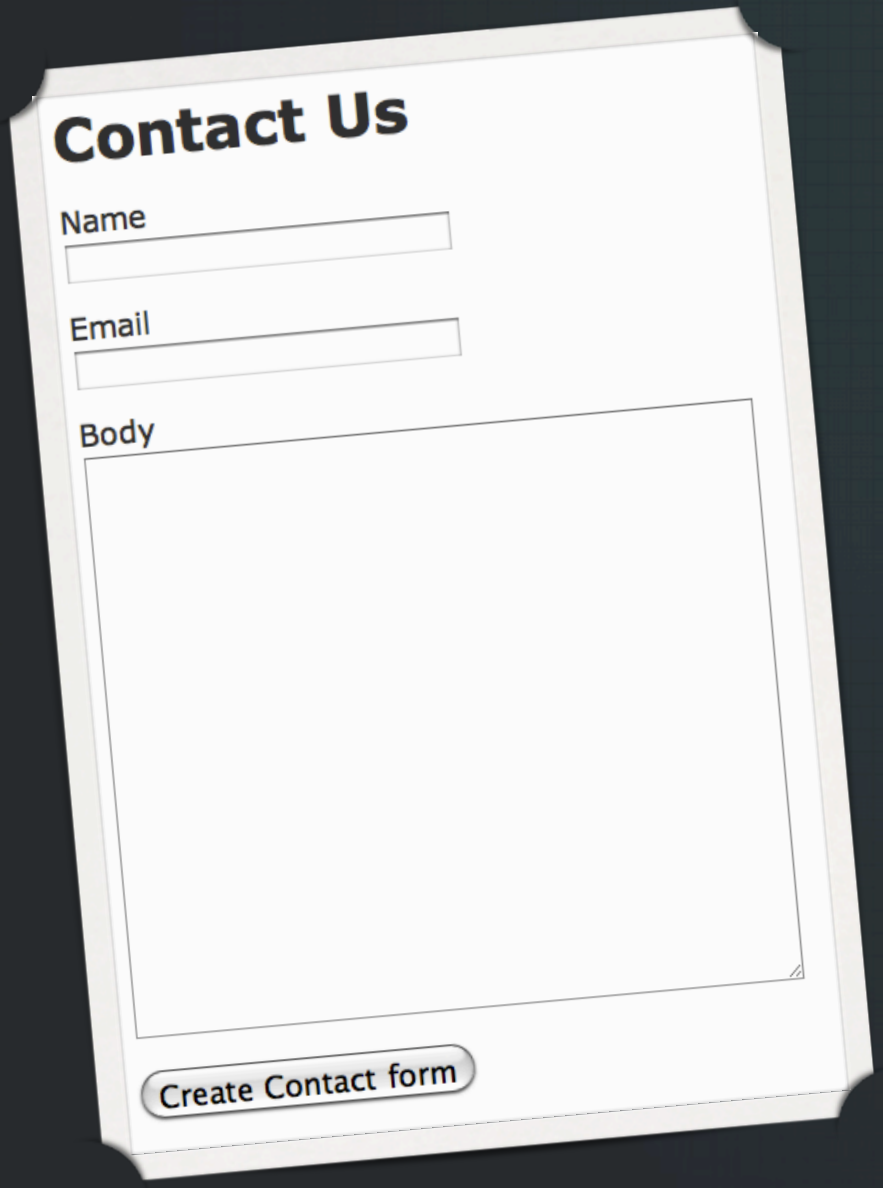
*/app/views/users/**edit.html.erb***

```ruby
<%= form_for(@user) do |f| %>
  ...
  <%= f.fields_for :account_setting do |a| %>
```

*/app/controllers/**users_controller.rb***

```ruby
class UsersController < ApplicationController
  def new
    @user = User.new(:account_setting => AccountSetting.new)
  end
end
```

MODELS WİTHOUT THE DATABASE



LEVEL 2 CONTROLLER COMMAND

MODELS WİTHOUT THE DATABASE

*/app/views/contact_us/**new.html.erb***

```
<h1>Contact Us</h1>

<%= form_for :contact, :url => send_email_path do |f| %>
  <div class="field">
    <%= f.label :name %><br />
    <%= f.text_field :name %>
  </div>
  <div class="field">
    <%= f.label :email %><br />
    <%= f.text_field :email %>
  </div>
  <div class="field">
    <%= f.label :body %><br />
    <%= f.text_area :body %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```

MODELS WİTHOUT THE DATABASE

*/app/controllers/**contact_us_controller.rb***

```ruby
class ContactUsController < ApplicationController

  def new
  end

  def send_email
    name = params[:contact][:name]
    email = params[:contact][:email]
    body = params[:contact][:body]

    if name.blank? || email.blank? || body.blank?
      flash.now[:notice] = "Please fill out all fields"
      render :action => 'new'
    else
      Notifications.contact_us(name, email, body).deliver
      flash[:notice] = "Email sent, we'll get back to you"
      redirect_to root_path
    end
  end
end
```

MODELS WİTHOUT THE DATABASE

*/app/controllers/**contact_us_controller.rb***

```
class ContactUsController < ApplicationController

  def new
    @contact_form = ContactForm.new
  end

  def send_email
    @contact_form = ContactForm.new(params[:contact_form])

    if !@contact_form.valid?
      render :action => 'new'
    else
      Notifications.contact_us(@contact_form).deliver
      flash[:notice] = "Email sent, we'll get back to you"
      redirect_to root_path
    end
  end
end
```

MODELS WİTHOUT THE DATABASE

*/app/controllers/**contact_us_controller.rb***

```
class ContactUsController < ApplicationController

  def new
    @contact_form = ContactForm.new
  end

  def send_email
    @contact_form = ContactForm.new(params[:contact_form])

    if @contact_form.valid?
      Notifications.contact_us(@contact_form).deliver
      flash[:notice] = "Email sent, we'll get back to you"
      redirect_to root_path
    else
      render :action => 'new'
    end
  end
end
```

USE THE POSITIVE INFLECTION

MODELS WİTHOUT THE DATABASE

*/app/controllers/**contact_us_controller.rb***

```
class ContactUsController < ApplicationController

  def new
    @contact_form = ContactForm.new
  end

  def send_email
    @contact_form = ContactForm.new(params[:contact_form])

    if @contact_form.valid?
      Notifications.contact_us(@contact_form).deliver
      redirect_to root_path, :notice => "Email sent, we'll get back to you"
    else
      render :action => 'new'
    end
  end
end
```

USE THE REDIRECT NOTICE SYNTAX

MODELS WİTHOUT THE DATABASE

*/app/controllers/**contact_us_controller.rb***

```ruby
class ContactUsController < ApplicationController

  def new
    @contact_form = ContactForm.new
  end

  def send_email
    @contact_form = ContactForm.new(params[:contact_form])

    if @contact_form.valid?
      Notifications.contact_us(@contact_form).deliver
      redirect_to root_path, :notice => "Email sent, we'll get back to you"
    else
      render :new
    end
  end
end
```

SHORTEN THE RENDER

MODELS WİTHOUT THE DATABASE

*/app/views/contact_us/**new.html.erb***

```
<h1>Contact Us</h1>

<%= form_for @contact_form, :url => send_email_path do |f| %>
```

MODELS WİTHOUT THE DATABASE

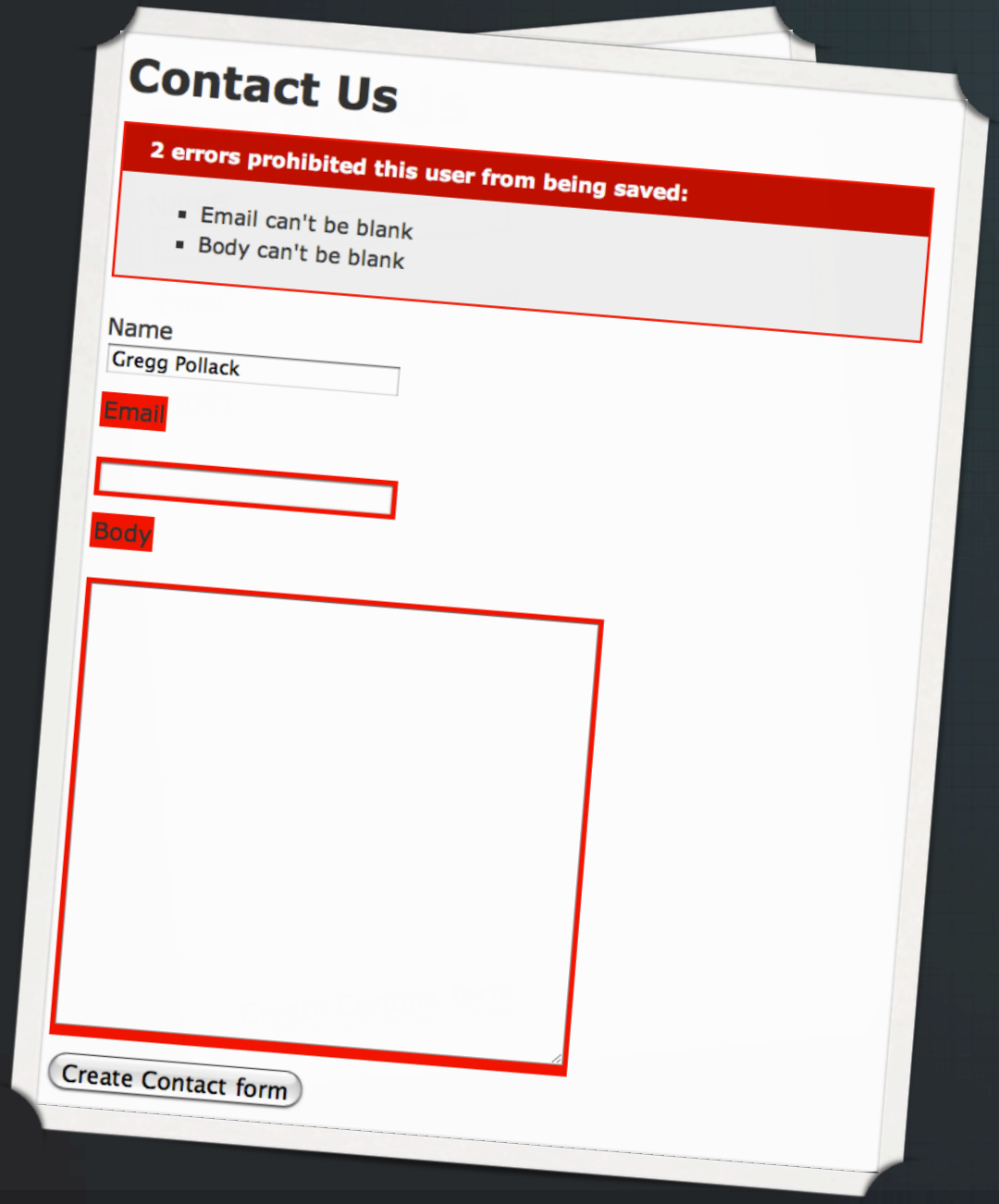
*/app/models/**contact_form.rb***

```ruby
class ContactForm
  include ActiveModel::Validations
  include ActiveModel::Conversion

  attr_accessor :name, :email, :body
  validates_presence_of :name, :email, :body

  def initialize(attributes = {})
    attributes.each do |name, value|
      send("#{name}=", value)
    end
  end

  def persisted?
    false
  end
end
```

```ruby
<%= form_for @contact_form
```

```ruby
ContactForm.new(params[:contact_form])
```

MODELS WİTHOUT THE DATABASE



REALLY REST

*/app/controllers/**users_controller.rb***

```ruby
class UsersController < ApplicationController

  def subscribe_mailing_list
    current_user.subscribe(params[:id])
    redirect_to current_user, :notice => "You've been subscribed"
  end

  def unsubscribe_mailing_list
    current_user.unsubscribe(params[:id])
    redirect_to current_user, :notice => "You have been unsubscribed"
  end

end
```

REALLY REST

*/app/controllers/**subscriptions_controller.rb***

```ruby
class SubscriptionsController < ApplicationController

  def create
    current_user.subscribe(params[:id])
    redirect_to current_user, :notice => "You've been subscribed"
  end

  def destroy
    current_user.unsubscribe(params[:id])
    redirect_to current_user, :notice => "You have been unsubscribed"
  end

end
```

REALLY REST

USE YOUR BEST JUDGEMENT

MORE THAN 2 LEVELS IS BAD

/users/1/posts/2/comments/3

NOT USING REST IS OKAY

/config/routes.rb

```
get "contact_us/new"
post "contact_us/send_email", :as => "send_email"
```

ENTER THE PRESENTERS

LEVEL 2 CONTROLLER COMMAND

ENTER THE PRESENTERS

*/app/controllers/**tweets_controller.rb***

```
def index
  @followers_tweets = current_user.followers_tweets.limit(20)
  @recent_tweet = current_user.tweets.first
  @following = current_user.following.limit(5)
  @followers = current_user.followers.limit(5)
  @recent_favorite = current_user.favorite_tweets.first
  @recent_listed = current_user.recently_listed.limit(5)

  if current_user.trend_option == "worldwide"
    @trends = Trend.worldwide.by_promoted.limit(10)
  else
    @trends = Trend.filter_by(current_user.trend_option).limit(10)
  end
  ....
end
```

ENTER THE PRESENTERS

*/app/controllers/**tweets_controller.rb***

```
def index
  @presenter = Tweets::IndexPresenter.new(current_user)
end
```

*/config/**application.rb***

```
config.autoload_paths += [config.root.join("app/presenters")]
```

*/app/presenters/tweets/**index_presenter.rb***

```
class Tweets::IndexPresenter
  def initialize(user)
    @user = user
  end
```

ENTER THE PRESENTERS

*/app/presenters/tweets/**index_presenter.rb***

```
class Tweets::IndexPresenter
  def initialize(user)
    @user = user
  end
```

Old Controller

```
def index
  @followers_tweets = current_user.followers_tweets.limit(20)
  @recent_tweet = current_user.tweets.first
  ...
  if current_user.trend_option == "worldwide"
    @trends = Trend.worldwide.by_promoted.limit(10)
  else
    @trends = Trend.filter_by(current_user.trend_option).limit(10)
  end
end
```

ENTER THE PRESENTERS

*/app/presenters/tweets/**index_presenter.rb***

```ruby
class Tweets::IndexPresenter
  def initialize(user)
    @user = user
  end

  def followers_tweets
    @user.followers_tweets.limit(20)
  end

  def recent_tweet
    @user.tweets.first
  end

  def trends
    if @user.trend_option == "worldwide"
      Trend.worldwide.by_promoted.limit(10)
    else
      Trend.filter_by(@user.trend_option).limit(10)
    end
  end
```

ENTER THE PRESENTERS

*/app/presenters/tweets/**index_presenter.rb***

```ruby
class Tweets::IndexPresenter
  def initialize(user)
    @user = user
  end

  def recent_tweet
    @user.tweets.first
  end
```

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @presenter = Tweets::IndexPresenter.new(current_user)
end
```

*/app/views/tweets/**index.html.erb***

```erb
<%= @presenter.recent_tweet.body %>
<%= @presenter.recent_tweet.created_at %>
```

TWO OBJECTS!

ENTER THE PRESENTERS

*/app/presenters/tweets/**index_presenter.rb***

```ruby
class Tweets::IndexPresenter
  def initialize(user)
    @user = user
  end

  def recent_tweet
    @recent_tweet ||= @user.tweets.first
  end
```

MEMOIZED

*/app/controllers/**tweets_controller.rb***

```ruby
def index
  @presenter = Tweets::IndexPresenter.new(current_user)
end
```

*/app/views/tweets/**index.html.erb***

```
<%= @presenter.recent_tweet.body %>
<%= @presenter.recent_tweet.created_at %>
```

ONE OBJECT!

ENTER THE PRESENTERS

*/app/presenters/tweets/**index_presenter.rb***

```
class Tweets::IndexPresenter
  extend ActiveSupport::Memoizable

  def initialize(user)
    @user = user
  end

  def recent_tweet
    @user.tweets.first
  end

  memoize :recent_tweet, :followers_tweet, ...
```

*/app/controllers/**tweets_controller.rb***

```
def index
  @presenter = Tweets::IndexPresenter.new(current_user)
end
```

*/app/views/tweets/**index.html.erb***

```
<%= @presenter.recent_tweet.body %>
<%= @presenter.recent_tweet.created_at %>
```

MEMOİZATİON

ONE IS BETTER THAN THE OTHER

`||=` VALUE IS NOT STORED IF FALSE OR NIL IS RETURNED

```ruby
extend ActiveSupport::Memoizable
memoize :recent_tweet, :followers_tweet, ...
```

```ruby
def expensive(num)
  # lots of processing
end

memoize :expensive
```

```ruby
expensive(2)
expensive(4)

expensive(2)
expensive(4)
```

LOADED FROM CACHE

REJECT SQL INJECTION

```ruby
User.where("name = #{params[:name]}")
```

```ruby
User.where("name = ?", params[:name])
```

```ruby
User.where(:name => params[:name])
```

```ruby
Tweet.where("created_at >= :start_date AND created_at <= :end_date",
  {:start_date => params[:start_date], :end_date => params[:end_date]})
```

```ruby
Tweet.where(:created_at =>
        (params[:start_date].to_date)..(params[:end_date].to_date))
```

RAİLS 3 RESPONDER SYNTAX

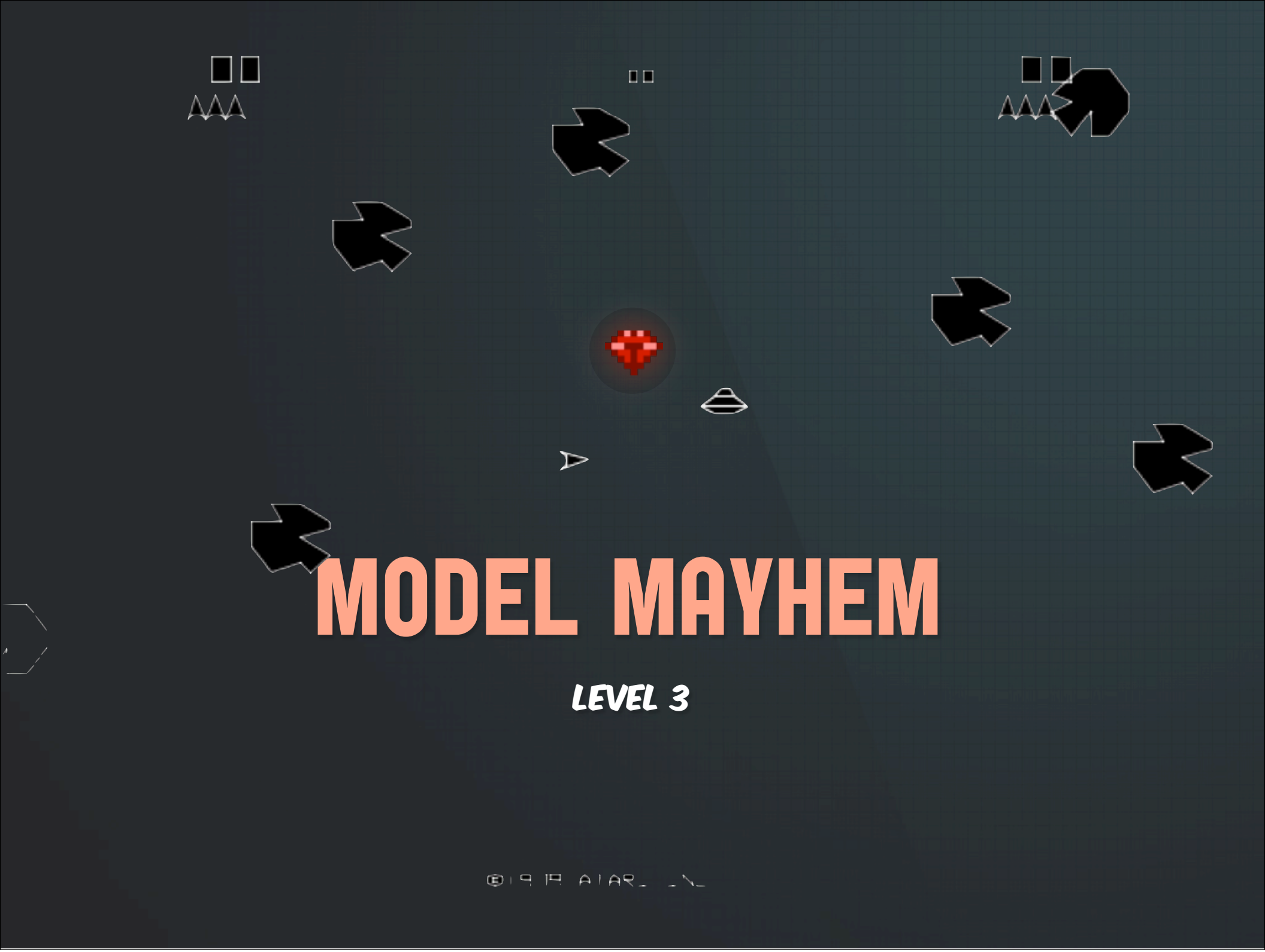
*/app/controllers/**users_controller.rb***

```
class UsersController < ApplicationController
  def index
    @users = User.all
    respond_to do |format|
      format.html
      format.xml { render :xml => @users.to_xml }
    end
  end

  def show
    @user = User.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @user }
    end
  end
...
```

RAİLS 3 RESPONDER SYNTAX

*/app/controllers/**users_controller.rb***

```
class UsersController < ApplicationController
  respond_to :html, :xml, :json

  def index
    @users = User.all
    respond_with(@users)
  end

  def show
    @user = User.find(params[:id])
    respond_with(@user)
  end
...
```


MODEL MAYHEM
LEVEL 3

LOVING YOUR INDICES

```
current_user.tweets
```

```
class AddIndexesToTables < ActiveRecord::Migration
  def self.up
    add_index :tweets, :user_id
  end


  def self.down
    remove_index :tweets, :user_id
  end
end
```

LOVING YOUR INDICES

```
current_user.tweets.order('created_at desc').limit(10)
Topic.where("started_trending > ?", 1.day.ago).order('mentions desc').limit(5)
```

IF THESE QUERIES ARE RUN A GREAT DEAL

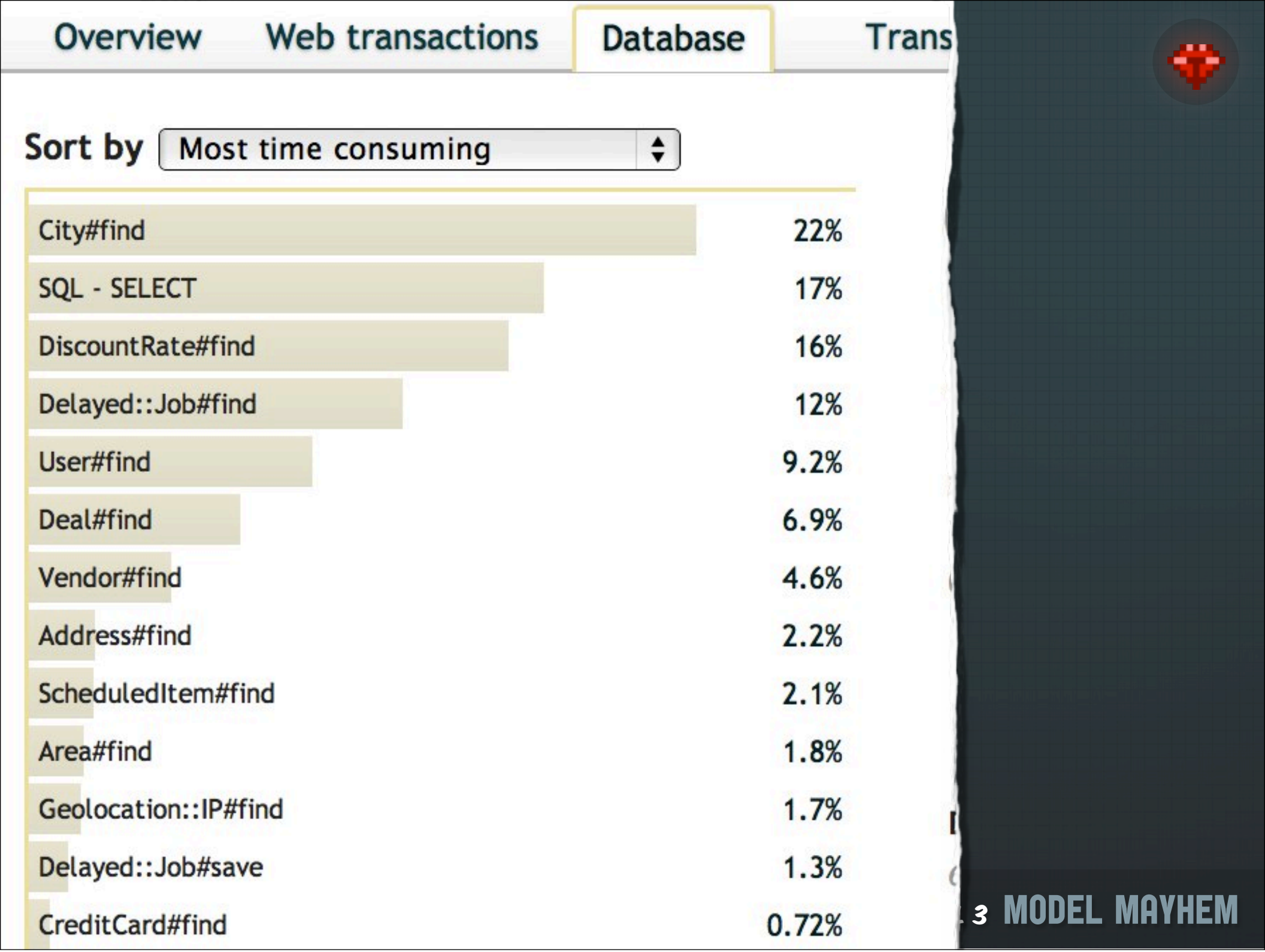
```
class AddIndexesToTables < ActiveRecord::Migration
  def self.up
    add_index :tweets, [:user_id, :created_at]
    add_index :topics, [:started_trending, :mentions]
  end


  def self.down
    remove_index :tweets, [:user_id, :created_at]
    remove_index :topics, [:started_trending, :mentions]
  end
end
```

LOVING YOUR INDICES

USE YOUR BEST JUDGEMENT

MORE INDICES, MORE TIME IT TAKES TO REINDEX

IF A 2 SECOND QUERY RUNS 5 TIMES A WEEK,
WHO CARES?


Overview
Web transactions
Database
Trans
Sort by
Most time consuming
City#find 22%
SQL - SELECT 17%
DiscountRate#find 16%
Delayed::Job#find 12%
User#find 9.2%
Deal#find 6.9%
Vendor#find 4.6%
Address#find 2.2%
ScheduledItem#find 2.1%
Area#find 1.8%
Geolocation::IP#find 1.7%
Delayed::Job#save 1.3%
CreditCard#find 0.72%
3 MODEL MAYHEM

Database report

COMPARE LAST 24 HOURS WITH:

☑ Previous 24 Hour Period ☐ Last Sunday ☐ Av

Filter Model Operation...

	Throughput	Total Time	Average Time	Model Operation Plotting: Total time
	40 cpm	0.258 s	6.4 ms	
Previous 24 Hour Period	44 cpm	0.291 s	6.6 ms	
				Delayed::Job#find
Last 24 Hours	25 cpm	0.190 s	7.6 ms	
Previous 24 Hour Period	25 cpm	0.196 s	7.7 ms	
				User#find
Last 24 Hours	11 cpm	0.119 s	10 ms	
Previous 24 Hour Period	13 cpm	0.139 s	11 ms	
				DiscountRate#find
Last 24 Hours	18 cpm	0.112 s	6.2 ms	
Previous 24 Hour Period	20 cpm	0.124 s	6.1 ms	

Use attr_protected or we will hack you

written by **Steven Bristol** on March 11th, 2008


```
$ curl -d "user[login]=hacked&user[is_admin]=true&user[password]
=password&user[password_confirmation]=password&user[email]=hacked@by.me"
http://url_not_shown/users
```

```
user[is_admin]=true
```

1. Use attr_protected.

At Less we do all four.

Use attr_protected

The attr_protected method in Rails will prevent the fields from being assigned via mass assignment. Here is an example:

Bad:

```
class User < ActiveRecord::Base
#no attr_protected here
#this will allow the creation of your hacked admin user
end

class Users < ApplicationController
  def create
      @user = User.create params[:user]
  end
```

PROTECTİNG YOUR ATTRİBUTES

*/app/models/**user.rb***

```
class User < ActiveRecord::Base
  attr_protected :is_admin
end
```

*/app/models/**user.rb***

```
class User < ActiveRecord::Base
  attr_accessible :email, :password, :password_confirmation
end
```

WHITELISTS ARE BETTER FOR SECURITY

DEFAULT VALUES

*/app/models/**account_setting.rb***

```ruby
class AccountSetting < ActiveRecord::Base
  belongs_to :user

  before_create :set_default_timezone

  def set_default_timezone
    self.time_zone = "EST"
  end

end
```

DEFAULT VALUES

*/app/models/**account_setting.rb***

```ruby
class AccountSetting < ActiveRecord::Base
  belongs_to :user
end
```

*/db/migrate/**20110119150620_add_default_time_zone_account_settings.rb***

```ruby
class AddDefaultTimeZoneToAccountSettings < ActiveRecord::Migration
  def self.up
    change_column_default :account_settings, :time_zone, 'EST'
  end

  def self.down
    change_column :account_settings, :time_zone, :string, nil
  end
end
```

PROPER USE OF CALLBACKS

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base

  before_create :set_trend_ending

  private

  def set_trend_ending
    self.finish_trending = Time.now + (60 * 60 * 24 * 7)
  end

end
```

PROPER USE OF CALLBACKS

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base

  before_create :set_trend_ending

  private

  def set_trend_ending
    self.finish_trending = 1.week.from_now
  end

end
```

PROPER USE OF CALLBACKS

*/app/models/**topic.rb***

```ruby
class Topic < ActiveRecord::Base
  TRENDING_PERIOD = 1.week

  before_create :set_trend_ending

  private

  def set_trend_ending
    self.finish_trending = TRENDING_PERIOD.from_now
  end

end
```

PROPER USE OF CALLBACKS

Creating an object

```
before_validation
after_validation
before_save
after_save
before_create
around_create
after_create
```

Updating an object

```
before_validation
after_validation
before_save
after_save
before_update
around_update
after_update
```

Deleting an object

```
before_destroy
after_destroy
around_destroy
```

RAİLS DATE HELPERS

Date Helpers

```
1.minute
2.hour
3.days
4.week
5.months
6.year
```

* singular or plural

Modifiers

```
beginning_of_day
beginning_of_week
beginning_of_month
beginning_of_quarter
beginning_of_year
```

* end can be used

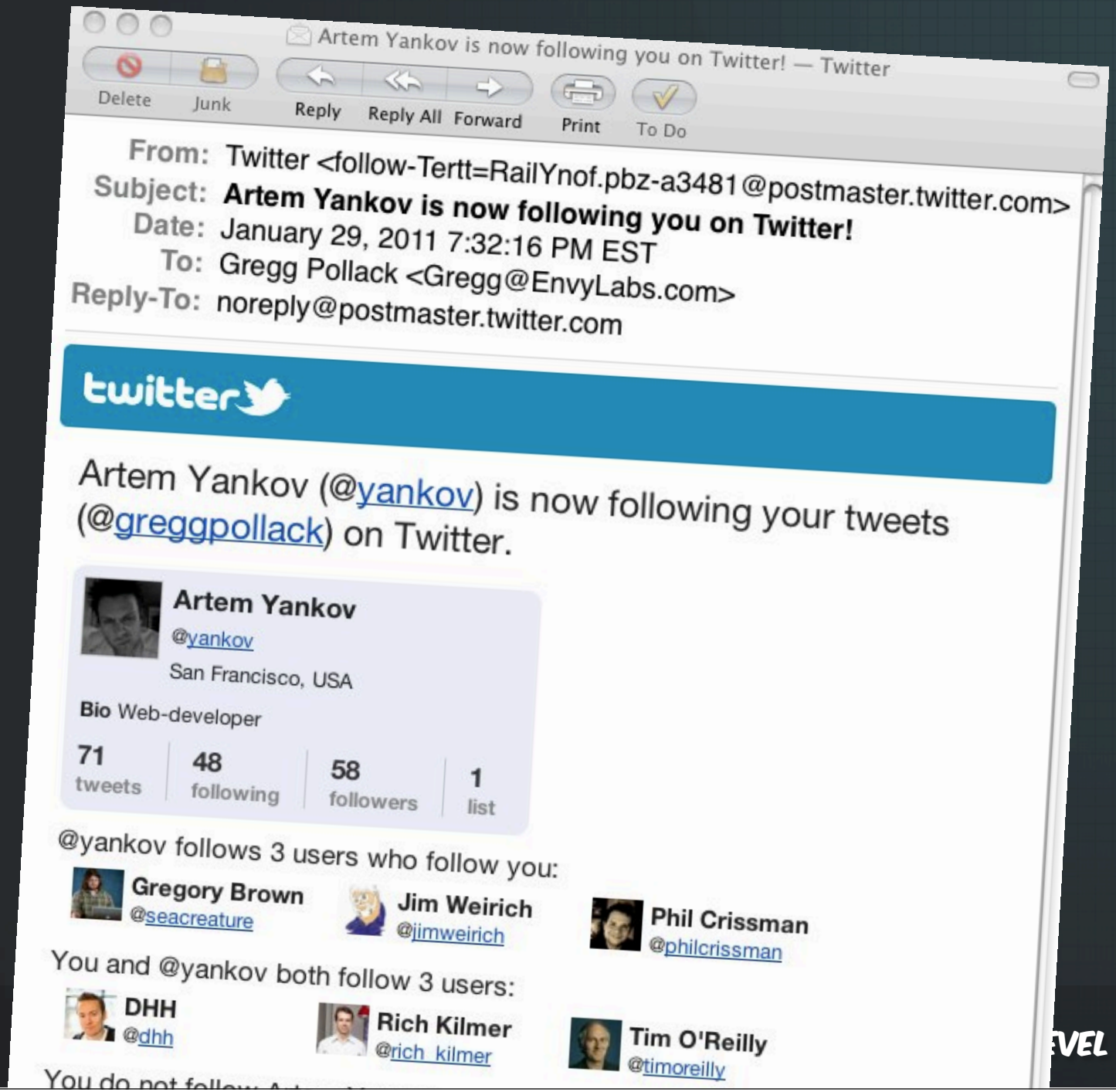
```
next_week
next_month
next_year
```

* prev can be used

More Modifiers

```
2.weeks.ago
3.weeks.from_now
```

PROPER USE OF CALLBACKS



PROPER USE OF CALLBACKS

*/app/models/**following.rb***

```
class Following < ActiveRecord::Base

  after_create :send_follower_notification

  def send_follower_notification
    if self.followed_user.receive_emails?
      queue_new_follower_email
    end
  end

end
```

PROPER USE OF CALLBACKS

*/app/models/**following.rb***

```
class Following < ActiveRecord::Base

  after_create :queue_new_follower_email,
               :if => :followed_can_receive_emails?

  def followed_can_receive_emails?
    self.followed_user.receive_emails?
  end
end
```

PROPER USE OF CALLBACKS

*/app/models/**following.rb***

```
class Following < ActiveRecord::Base

  after_create :queue_new_follower_email,
    :if => Proc.new {|f| f.followed_user.receive_emails? }

end
```

IMPROVED VALIDATION

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base

  def validate
    unless ContentModerator.is_suitable?(self.name)
      self.errors.add(:name, 'is inappropriate')
    end
  end

end
```

IMPROVED VALIDATION

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base
  validate :appropriate_content

  private

  def appropriate_content
    unless ContentModerator.is_suitable?(self.name)
      self.errors.add(:name, 'is inappropriate')
    end
  end

end
```

IMPROVED VALIDATION

*/app/models/**topic.rb***

```ruby
class Topic < ActiveRecord::Base
  validates :name, :appropriate => true
end
```

*/lib/**appropriate_validator.rb***

```ruby
class AppropriateValidator < ActiveRecord::EachValidator
  def validate_each(record, attribute, value)
    unless ContentModerator.is_suitable?(value)
      record.errors.add(attribute, 'is inappropriate')
    end
  end
end
```

DON'T FORGET TO REQUIRE THIS

/lib isn't auto-loaded by default

SOWING THE SEEDS

*/db/migrate/**20110114221048_create_topics.rb***

```
class CreateTopics < ActiveRecord::Migration
  def self.up
    create_table :topics do |t|
      t.string :name
      t.datetime :started_trending
      t.integer :mentions

      t.timestamps
    end

    Topic.create(:name => "Rails for Zombies", :mentions => 1023)
    Topic.create(:name => "Top Ruby Jobs", :mentions => 231)
    Topic.create(:name => "Ruby5", :mentions => 2312)
  end

  def self.down
    drop_table :topics
  end
end
```

SOWING THE SEEDS

*/db/**seeds.rb***

```ruby
Topic.create(:name => "Rails for Zombies", :mentions => 1023)
Topic.create(:name => "Top Ruby Jobs", :mentions => 231)
Topic.create(:name => "Ruby5", :mentions => 2312)
```

Run from command line

```
$ rake db:seed
```

MENTIONS WON'T BE SET!

*/app/models/**topic.rb***

```ruby
class Topic < ActiveRecord::Base
  attr_protected :mentions
end
```

SOWING THE SEEDS

/db/seeds.rb

```
topics = [
  {:name => "Rails for Zombies", :mentions => 1023},
  {:name => "Top Ruby Jobs", :mentions => 231},
  {:name => "Ruby5", :mentions => 2312}
]

topics.each do |attributes|
  Topic.create do |t|
    t.name = attributes[:name]
    t.mentions = attributes[:mentions]
  end
end
```

WHAT IF WE WANT TO BE ABLE TO UPDATE THE SEED?

SOWING THE SEEDS

/db/seeds.rb

```
topics = [
  {:name => "Rails for Zombies", :mentions => 1023},
  {:name => "Top Ruby Jobs", :mentions => 231},
  {:name => "Ruby5", :mentions => 2312}
]

Topic.destroy_all

topics.each do |attributes|
  Topic.create do |t|
    t.name = attributes[:name]
    t.mentions = attributes[:mentions]
  end
end
```

DANGEROUS IF THERE ARE LOTS OF RELATIONSHIPS

SOWING THE SEEDS

/db/seeds.rb

```ruby
topics = [
  {:name => "Rails for Zombies", :mentions => 1023},
  {:name => "Top Ruby Jobs", :mentions => 231},
  {:name => "Ruby5", :mentions => 2312}
]

topics.each do |attributes|
  Topic.find_or_initialize_by_name(attributes[:name]).tap do |t|
    t.mentions = attributes[:mentions]
    t.save!
  end
end
```


MODEL BERT
LEVEL 4
PLAYER 1
11435
CHANGE TO:
LEVEL 2
ROUND: 1
@!#?@!

N+1 IS NOT FOR FUN

*/app/models/**user.rb***

```
class User
  def recent_followers
    self.followers.recent.collect{ |f| f.user.name }.to_sentence
  end
end
```

```
=> "Gregg, Eric, Dray, and Nate"
```

```
Select followers where user_id=1
    Select user where id=2
    Select user where id=3
    Select user where id=4
    Select user where id=5
```

N+1 IS NOT FOR FUN

*/app/models/**user.rb***

```
class User
  def recent_followers
    self.followers.recent.includes(:user).collect{ |f| f.user.name }.to_sentence
  end
end
```

```
Select followers where user_id=1

   Select users where user_id in (2,3,4,5)
```

2 QUERIES INSTEAD OF 5!

BULLET GEM

https://github.com/flyerhzm/bullet

TO FIND ALL YOUR N+1 QUERIES

COUNTER_CACHE MONEY

*/app/views/tweets/**index.html.erb***

```
<% @tweets.each do |tweet| %>

    <div class="tweet">
      <%= tweet.status %>
      <span class="retweets">
        <%= tweet.retweets.length %> ReTweets
      </span>
    </div>

<% end %>
```

2 ReTweets
1 ReTweets
0 ReTweets

BAD ENGLISH

COUNTER_CACHE MONEY

*/app/views/tweets/**index.html.erb***

```
<% @tweets.each do |tweet| %>

    <div class="tweet">
      <%= tweet.status %>
      <span class="retweets">
        <%= pluralize(tweet.retweets.length, "ReTweet") %>
      </span>
    </div>

<% end %>
```

2 ReTweets
1 ReTweet
0 ReTweets

COUNTER_CACHE MONEY

*/app/views/tweets/**index.html.erb***

```
<% @tweets.each do |tweet| %>

    <div class="tweet">
      <%= tweet.status %>
      <span class="retweets">
        <%= pluralize(tweet.retweets.length, "ReTweet") %>
      </span>
    </div>

<% end %>
```

LOTS OF UNNEEDED OBJECTS

FOR EACH TWEET

1. SELECT ALL RETWEETS WHERE USER_ID=X
2. POPULATE AN ARRAY OF TWEET OBJECTS
3. CALL LENGTH ON THAT ARRAY

COUNTER_CACHE MONEY

*/app/views/tweets/**index.html.erb***

```
<% @tweets.each do |tweet| %>

    <div class="tweet">
      <%= tweet.status %>
      <span class="retweets">
        <%= pluralize(tweet.retweets.count, "ReTweet") %>
      </span>
    </div>

<% end %>
```

FOR EACH TWEET

1. SELECT ALL RETWEETS WHERE USER_ID=X
2. DO A COUNT QUERY FOR RETWEETS

POSSIBLY 10+ COUNT QUERIES

COUNTER_CACHE MONEY

*/app/views/tweets/**index.html.erb***

with counter_cache

```
<% @tweets.each do |tweet| %>

    <div class="tweet">
      <%= tweet.status %>
      <span class="retweets">
        <%= pluralize(tweet.retweets.size, "ReTweet") %>
      </span>
    </div>

<% end %>
```

FOR EACH TWEET

1. SELECT ALL RETWEETS WHERE USER_ID=X

THERE IS NO STEP 2

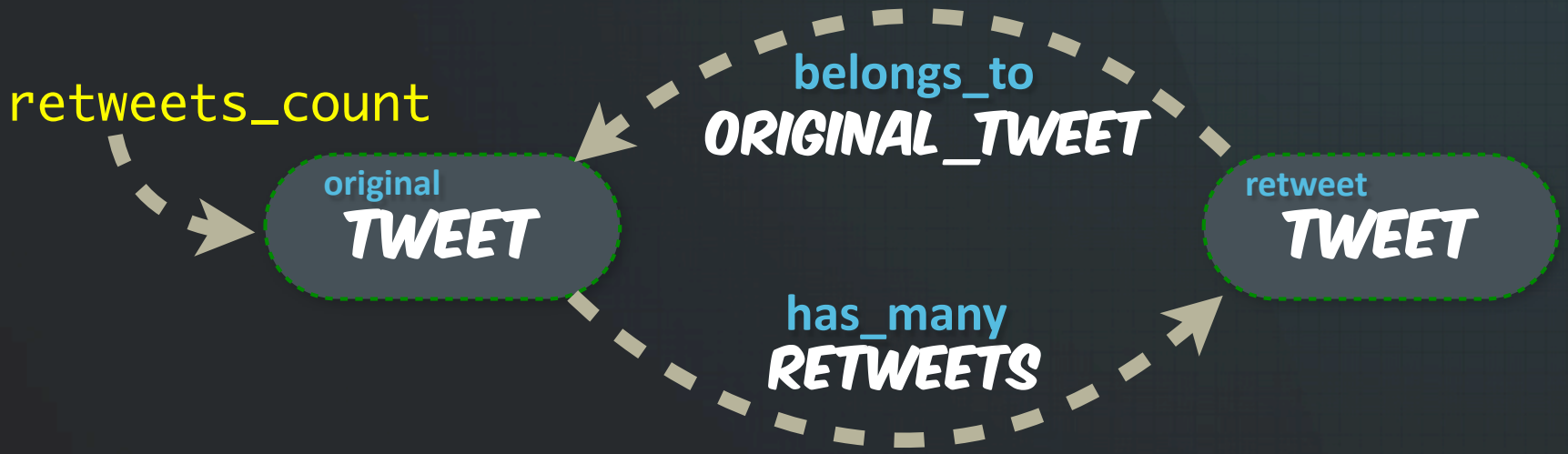
COUNTER_CACHE MONEY

*/app/models/**tweet.rb***

```
class Tweet < ActiveRecord::Base
  belongs_to :original_tweet,
             :class_name => 'Tweet',
             :foreign_key => :tweet_id

  has_many :retweets,
           :class_name => 'Tweet',
           :foreign_key => :tweet_id
end
```

Our migration

```
class AddCountRetweets
  def self.up
    add_column :tweets,
               :retweets_count,
               :integer,
               :default => 0
  end
  ...
```



COUNTER_CACHE MONEY

*/app/models/**tweet.rb***

```
class Tweet < ActiveRecord::Base
  belongs_to :original_tweet,
             :class_name => 'Tweet',
             :foreign_key => :tweet_id,
             :counter_cache => true

  has_many :retweets,
           :class_name => 'Tweet',
           :foreign_key => :tweet_id
end
```

UNABLE TO FIND TWEETS_COUNT

```
retweets_count
```

COUNTER_CACHE MONEY

*/app/models/**tweet.rb***

```
class Tweet < ActiveRecord::Base
  belongs_to :original_tweet,
             :class_name => 'Tweet',
             :foreign_key => :tweet_id,
             :counter_cache => :retweets_count


  has_many :retweets,
           :class_name => 'Tweet',
           :foreign_key => :tweet_id
end
```

```
current_user.tweets.create(
    :status => "RT #{self.user.name}: #{self.status}",
    :original_tweet => self
    )
```

WILL CAUSE AN INSERT AND

```
UPDATE "tweets" SET "retweets_count" = "retweets_count" + 1 WHERE ("tweets"."id" = 42)
```

COUNTER_CACHE MONEY

	WITHOUT CACHE COUNTER	WITH CACHE COUNTER
`t.retweets.length`	pull all records then calls .length	pull all records then calls .length
`t.retweets.count`	count query	count query
`t.retweets.size`	count query	no query look at cache

BATCHES OF FİND_EACH

*/lib/tasks/**long_running_task.rake***

```
desc 'Task involving all tweets'
task :tweet_task => :environment do

  Tweet.all.each do |tweet|
    p "task for #{tweet}"
  end

end
```

NOT SO GOOD IF YOU HAVE MILLIONS OF TWEETS

BATCHES OF FİND_EACH

*/lib/tasks/**long_running_task.rake***

```
desc 'Task involving all tweets'
task :tweet_task => :environment do

  Tweet.find_each do |tweet|
    p "task for #{tweet}"
  end

end
```

PULLS BATCHES OF 1,000 AT A TIME

BATCHES OF FİND_EACH

*/lib/tasks/**long_running_task.rake***

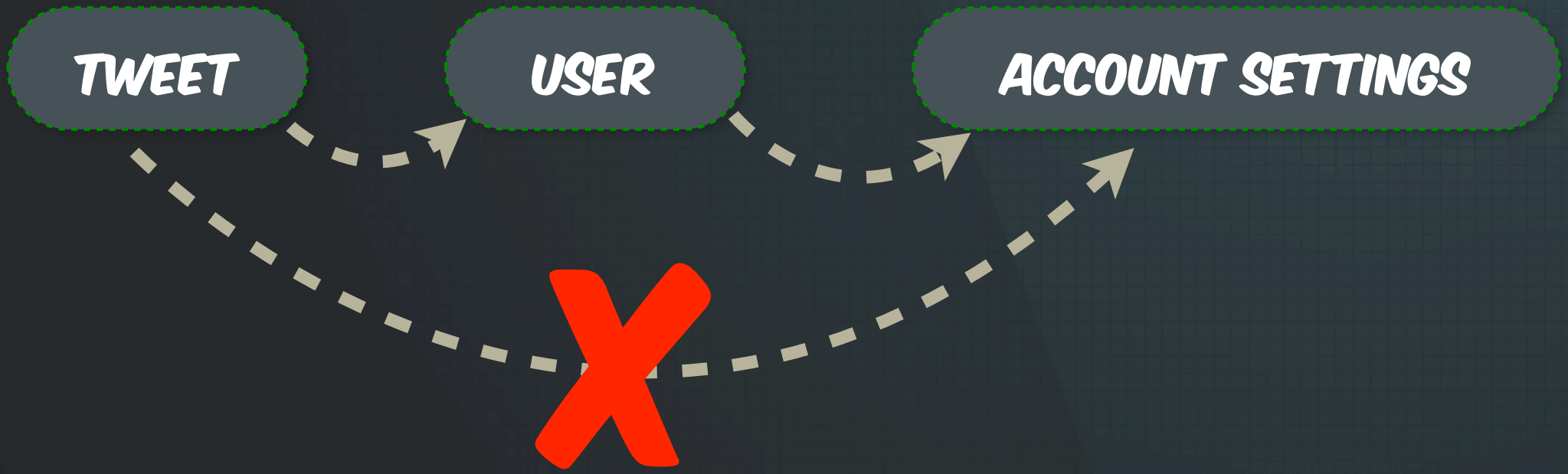
```
desc 'Task involving all tweets'
task :tweet_task => :environment do

  Tweet.find_each(:batch_size => 200) do |tweet|
    p "task for #{tweet}"
  end

end
```

PULLS BATCHES OF 200 AT A TIME

LAW OF DEMETER

EACH UNIT SHOULD HAVE LIMITED KNOWLEDGE ABOUT OTHER UNITS

"DON'T TALK TO STRANGERS"



LAW OF DEMETER

*/app/models/**tweet.rb***

```
class Tweet < ActiveRecord::Base
  def location_data
    if self.user.account_setting.location_on_tweets
      self.location
    else
      "unavailable"
    end
  end
end
```

THE TWEET SHOULDN'T KNOW ABOUT ACCOUNT_SETTING!

LAW OF DEMETER

*/app/models/**tweet.rb***

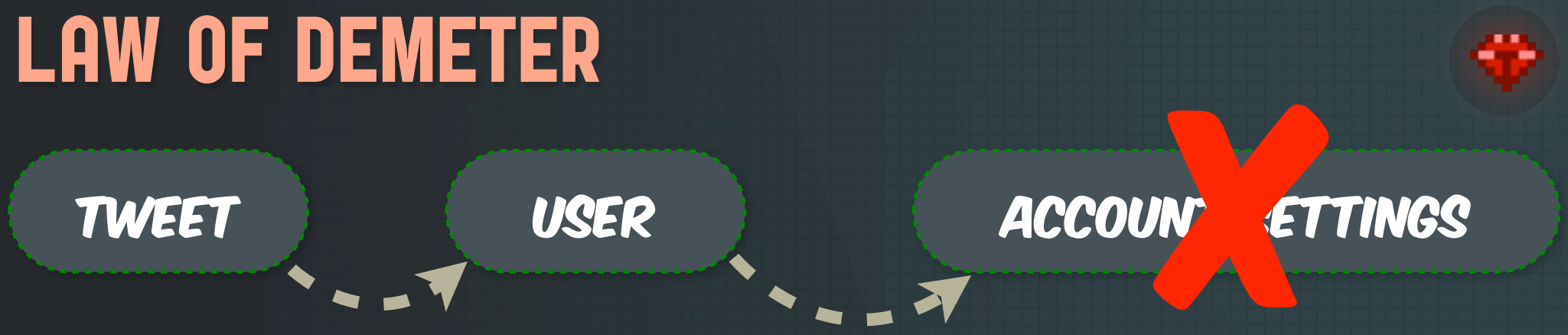
```ruby
class Tweet < ActiveRecord::Base
  def location_data
    if self.user.location_on_tweets
      self.location
    else
      "unavailable"
    end
  end
end
```

*/app/models/**user.rb***

```ruby
class User < ActiveRecord::Base
  has_one :account_setting, :dependent => :destroy

  delegate :location_on_tweets, :public_email,
           :to => :account_setting

end
```

ADDITIONAL METHODS

LAW OF DEMETER



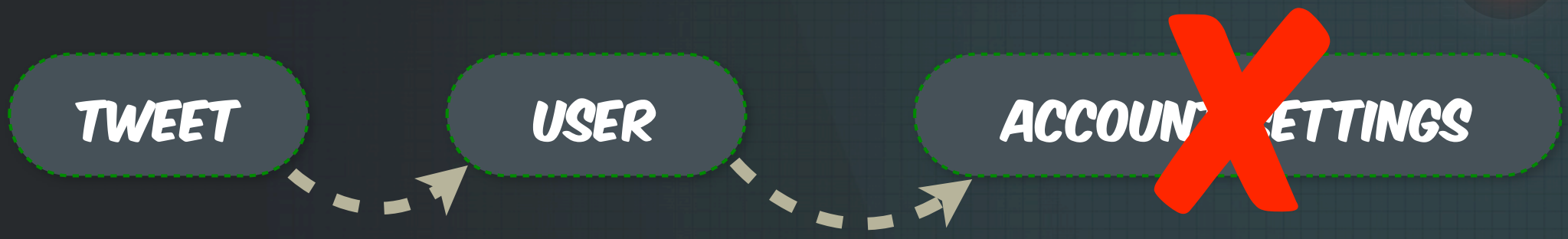
*/app/models/**user.rb***

```
class User < ActiveRecord::Base
  has_one :account_setting, :dependent => :destroy

  delegate :location_on_tweets, :public_email,
           :to => :account_setting

end
```

```
self.user.location_on_tweets
```

ERROR!! ACCOUNT_SETTING IS NIL!

LAW OF DEMETER



*/app/models/**user.rb***

```
class User < ActiveRecord::Base
  has_one :account_setting, :dependent => :destroy

  delegate :location_on_tweets, :public_email,
           :to => :account_setting,
           :allow_nil => true

end
```

```
self.user.location_on_tweets
```

RETURNS NIL WHEN ACCOUNT_SETTINGS IS MISSING

HEAD TO TO_S

*/app/models/**user.rb***

```ruby
class User < ActiveRecord::Base
  def display_name
    "#{first_name} #{last_name}"
  end
end
```

```
<%= @user.display_name %>
```

HEAD TO TO_S

*/app/models/**user.rb***

```ruby
class User < ActiveRecord::Base
  def to_s
    "#{first_name} #{last_name}"
  end
end
```

```
<%= @user %>
```

TO_PARAM-ALAMA DİNG DONG

SEO FRIENDLY URLS

/post/2133

/post/rails-best-practices

/post/2133-rails-best-practices

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base
  def to_param
    "#{id}-#{name.parameterize}"
  end
end
```

TO_PARAM-ALAMA DİNG DONG

```
<%= link_to topic.name, topic %>
```

Will generate

```
/post/2133-rails-best-practices
```

```
{:id => "2133-rails-best-practices"}
```

```
Topic.find(params[:id])
```

Will call to_i

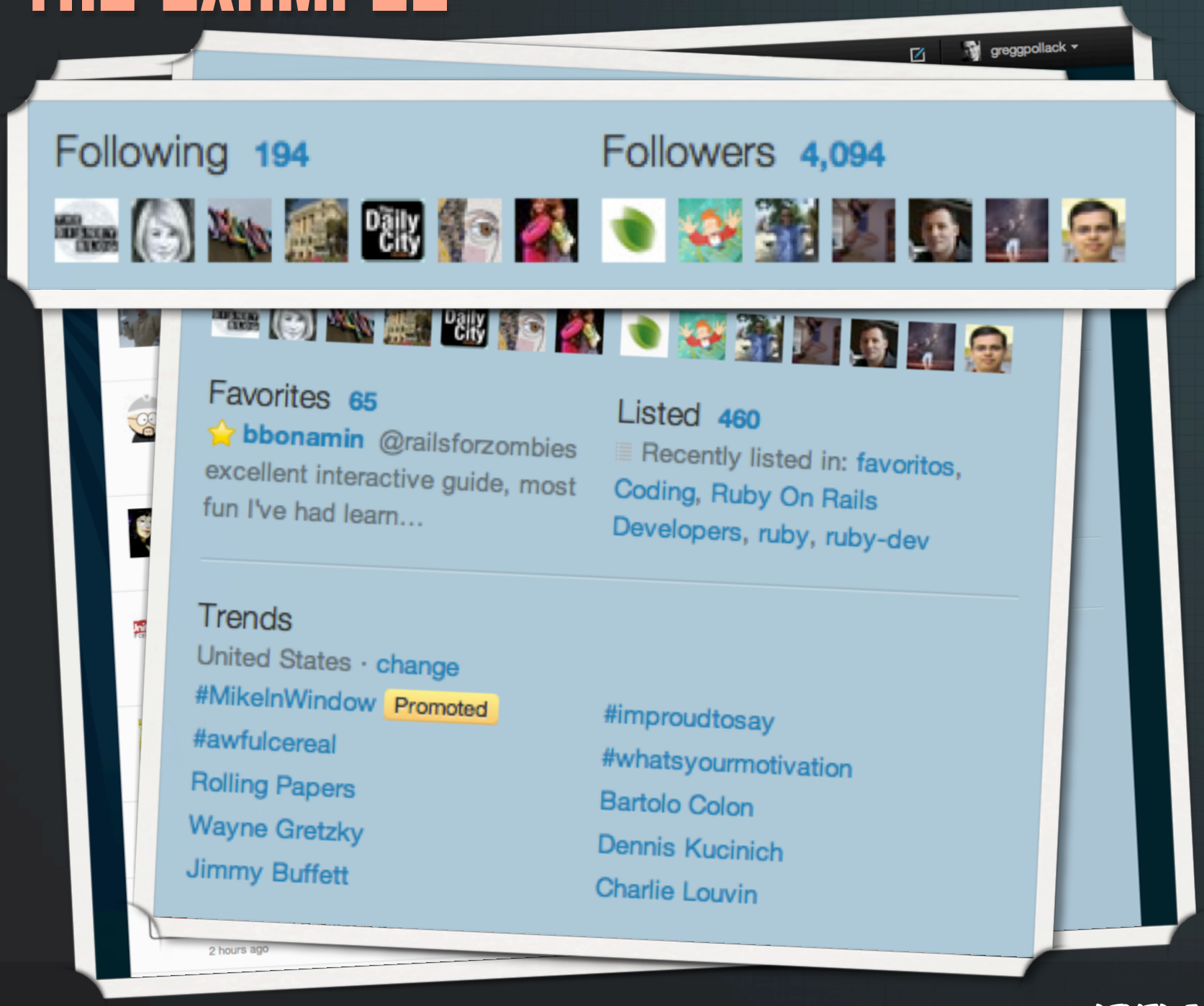
```
Topic.find(2133)
```

*/app/models/**topic.rb***

```
class Topic < ActiveRecord::Base
  def to_param
    "#{id}-#{name.parameterize}"
  end
end
```


FROGGY VIEWS
LEVEL 5

THE EXAMPLE


greggpollack
Following 194
Followers 4,094
Favorites 65
bbonamin @railsforzombies excellent interactive guide, most fun I've had learn...
Listed 460
Recently listed in: favoritos, Coding, Ruby On Rails Developers, ruby, ruby-dev
Trends
United States · change
#MikeInWindow Promoted
#awfulcereal
Rolling Papers
Wayne Gretzky
Jimmy Buffett
#improudtosay
#whatsyourmotivation
Bartolo Colon
Dennis Kucinich
Charlie Louvin
2 hours ago

NO QUERIES IN YOUR VIEW!

*/app/views/tweets/**index.html.erb***

```
<% current_user.who_to_follow.limit(5).each do |f| %>
  <li><%= f.name %> <%= link_to "Follow", follow_user_path(f) %></li>
<% end %>
```

QUERY SHOULDN'T BE IN OUR VIEW!

NO QUERIES IN YOUR VIEW!

*/app/views/tweets/**index.html.erb***

```
<% @who_to_follow.each do |f| %>
  <li><%= f.name %> - <%= link_to "Follow", follow_user_path(f) %></li>
<% end %>
```

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController

  def index
    @who_to_follow = current_user.who_to_follow.limit(5)
  end

end
```

HELPER SKELTER

*/app/views/tweets/**index.html.erb***

```
<div class="followers">
  Followers
  <span><%= @followers_count %></span>
  <% @recent_followers.each do |f| %>
     <a href="<%= user_path(f) %>">
        <img src="<%= f.avatar.url(:thumb) %>" />
     </a>
  <% end %>
</div>

<div class="following">
  Following
  <span><%= @following_count %></span>
  <% @recent_following.each do |f| %>
    <a href="<%= user_path(f) %>">
      <img src="<%= f.avatar.url(:thumb) %>" />
    </a>
  <% end %>
</div>
```

HELPER SKELTER

*/app/views/tweets/**index.html.erb***

```
<%= follow_box("Followers", @followers_count, @recent_followers) %>
<%= follow_box("Following", @following_count, @recent_following) %>
```

*/app/helpers/**tweets_helper.rb***

```
def follow_box(title, count, recent)
  str = "<div class=\"#{title.downcase}\">" +
        "#{title}<span>#{count}</span>"
  recent.each do |user|
    str += "<a href=\"#{user_path(user)}\">"
    str += "<img src=\"#{user.avatar.url(:thumb)}\">"
    str += "</a>"
  end
  raw(str += "</div>")
end
```

USE PROPER LINK_TO AND IMAGE_TAG

HELPER SKELTER

*/app/views/tweets/**index.html.erb***

```
<%= follow_box("Followers", @followers_count, @recent_followers) %>
<%= follow_box("Following", @following_count, @recent_following) %>
```

*/app/helpers/**tweets_helper.rb***

```
def follow_box(title, count, recent)
  str = "<div class=\"#{title.downcase \">" +
        "#{title}<span>#{count}</span>"
  recent.each do |user|
    str += link_to user do
      image_tag(user.avatar.url(:thumb))
    end
  end
  raw(str += "</div>")
end
```

USE HTML HELPERS?

HELPER SKELTER

*/app/views/tweets/**index.html.erb***

```
<%= follow_box("Followers", @followers_count, @recent_followers) %>
<%= follow_box("Following", @following_count, @recent_following) %>
```

*/app/helpers/**tweets_helper.rb***

```
def follow_box(title, count, recent)
  content_tag :div, :class => title.downcase do
    str = title + content_tag(:span, count)
    recent.each do |user|
      str += link_to user do
        image_tag(user.avatar.url(:thumb))
      end
    end
    raw(str)
  end
end
```

ANNOYING STR VARIABLE

HELPER SKELTER

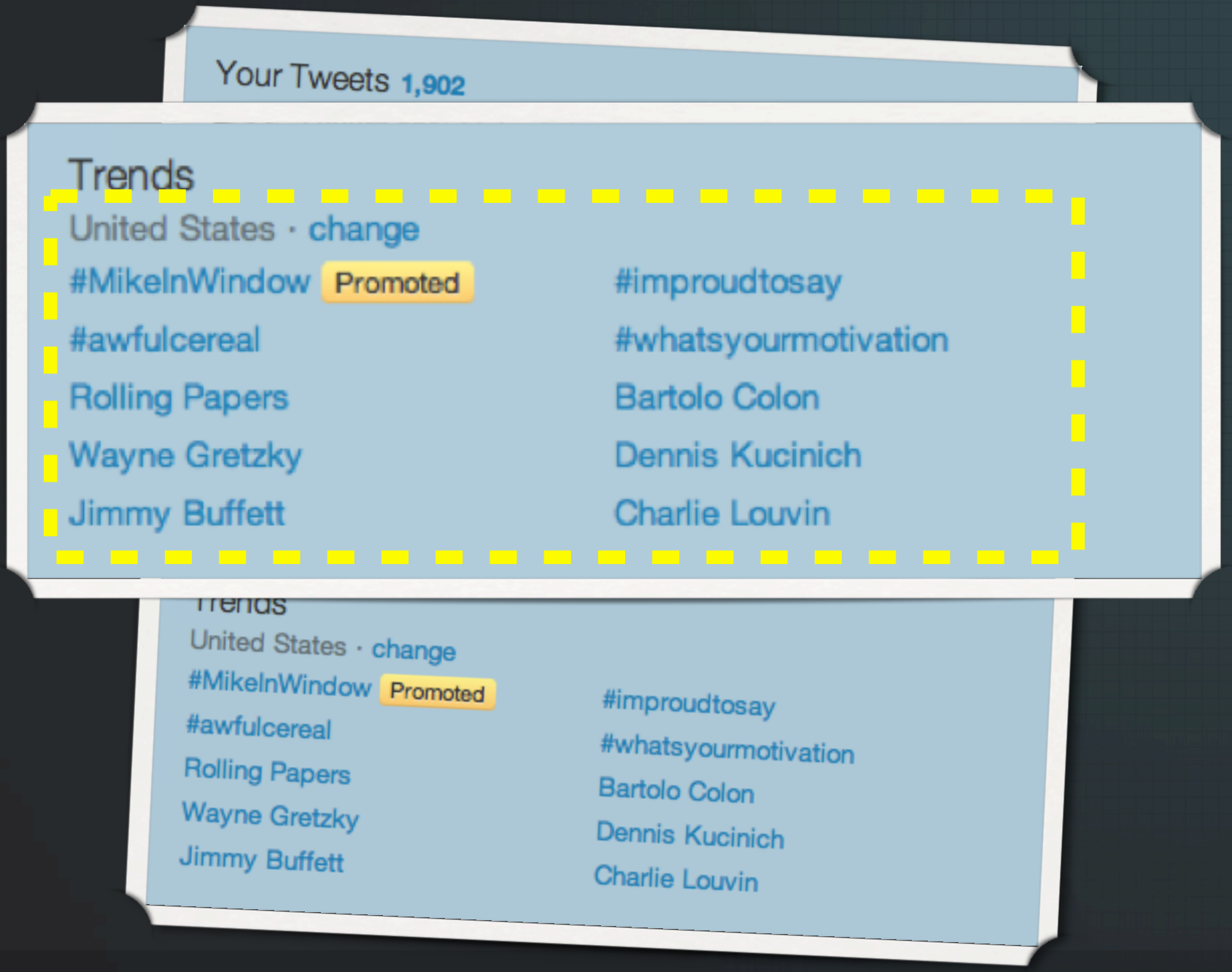
*/app/views/tweets/**index.html.erb***

```
<%= follow_box("Followers", @followers_count, @recent_followers) %>
<%= follow_box("Following", @following_count, @recent_following) %>
```

*/app/helpers/**tweets_helper.rb***

```
def follow_box(title, count, recent)
  content_tag :div, :class => title.downcase do
    raw(
      title +
      content_tag(:span, count) +
      recent.collect do |user|
        link_to user do
          image_tag(user.avatar.url(:thumb))
        end
      end.join
      )
  end
end
```

THE EXAMPLE


Your Tweets 1,902
Trends
United States · change
#MikeInWindow Promoted
#awfulcereal
Rolling Papers
Wayne Gretzky
Jimmy Buffett
#improudtosay
#whatsyourmotivation
Bartolo Colon
Dennis Kucinich
Charlie Louvin

PARTİAL SANİTY

*/app/views/tweets/**index.html.erb***

```
<h2>Trends</h2>
<%= render :partial => 'trending' %>
```

*/app/views/tweets/**_trending.html.erb***

```
<h3><%= @user.trending_area %></h3>
<ul>
  <% @trending.each do |topic| %>
    <li>
      <%= link_to topic.name, topic %>
      <% if topic.promoted? %>
        <%= link_to image_tag('promoted.jpg'), topic %>
      <% end %>
    </li>
  <% end %>
</ul>
```

PARTIAL SANITY

*/app/views/tweets/**index.html.erb***

```
<h2>Trends</h2>
<%= render 'trending' %>
```

*/app/views/tweets/**_trending.html.erb***

```
<h3><%= @user.trending_area %></h3>
<ul>
  <% @trending.each do |topic| %>
    <li>
      <%= link_to topic.name, topic %>
      <% if topic.promoted? %>
        <%= link_to image_tag('promoted.jpg'), topic %>
      <% end %>
    </li>
  <% end %>
</ul>
```

THERE ARE INSTANCE VARIABLES IN OUR PARTIAL!

PARTİAL SANİTY

*/app/views/tweets/**index.html.erb***

```
<h2>Trends</h2>
<%= render 'trending', :area => @user.trending_area,
                       :topics => @trending %>
```

*/app/views/tweets/**_trending.html.erb***

```
<h3><%= area %></h3>
<ul>
  <% topics.each do |topic| %>
    <li>
      <%= link_to topic.name, topic %>
      <% if topic.promoted? %>
        <%= link_to image_tag('promoted.jpg'), topic %>
      <% end %>
    </li>
  <% end %>
</ul>
```

PARTİAL SANİTY

*/app/views/tweets/_**trending.html.erb***

```
<h3><%= area %></h3>
<ul>
  <% topics.each do |topic| %>
    <%= render 'topics/topic', :topic => topic %>
  <% end %>
</ul>
```

*/app/views/topics/_**topic.html.erb***

```
<li>
  <%= link_to topic.name, topic %>
  <% if topic.promoted? %>
    <%= link_to image_tag('promoted.jpg'), topic %>
  <% end %>
</li>
```

PARTİAL SANİTY

*/app/views/tweets/_**trending.html.erb***

```
<h3><%= area %></h3>
<ul>
  <% topics.each do |topic| %>
    <%= render topic %>
  <% end %>
</ul>
```

USING CLASS NAME TO FIND PARTIAL

*/app/views/topics/_**topic.html.erb***

```
<li>
  <%= link_to topic.name, topic %>
  <% if topic.promoted? %>
    <%= link_to image_tag('promoted.jpg'), topic %>
  <% end %>
</li>
```

PARTİAL SANİTY

*/app/views/tweets/_**trending.html.erb***

```
<h3><%= area %></h3>
<ul>
  <%= render :partial => 'topics/topic', :collection => topics %>
</ul>
```

*/app/views/topics/_**topic.html.erb***

```
<li>
  <%= link_to topic.name, topic %>
  <% if topic.promoted? %>
    <%= link_to image_tag('promoted.jpg'), topic %>
  <% end %>
</li>
```

PARTİAL SANİTY

*/app/views/tweets/_**trending.html.erb***

```
<h3><%= area %></h3>
<ul>
  <%= render topics %>
</ul>
```

*/app/views/topics/_**topic.html.erb***

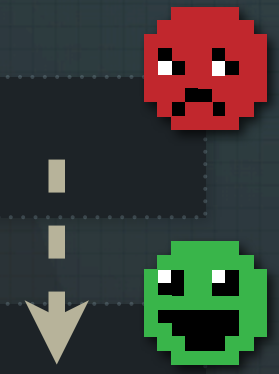
```
<li>
  <%= link_to topic.name, topic %>
  <% if topic.promoted? %>
    <%= link_to image_tag('promoted.jpg'), topic %>
  <% end %>
</li>
```

EMPTY STRİNG THİNGS

```
<% if @user.email.blank? %>
```

```
<% unless @user.email? %>
```

```
<% if @user.email.present? %>
```



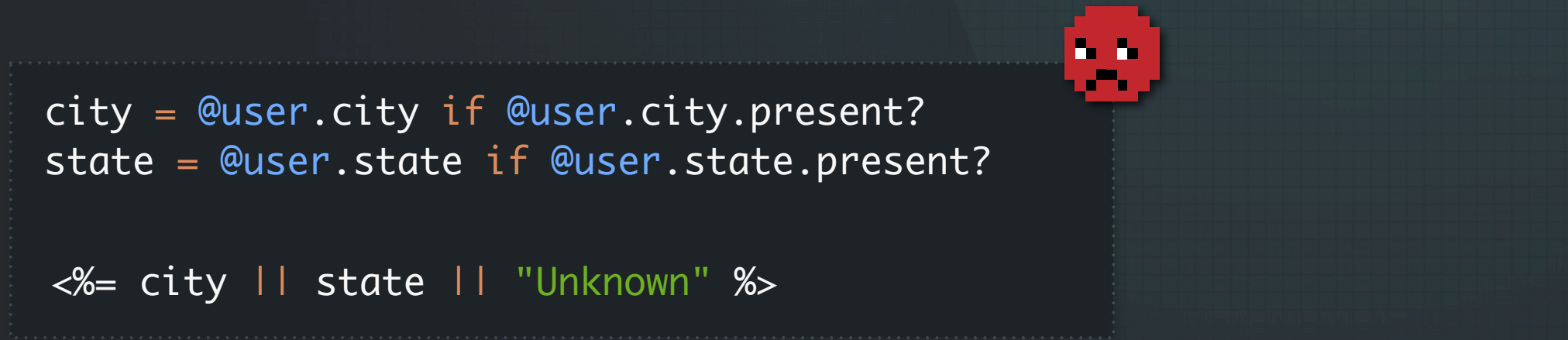
```
<% if @user.email? %>
```

EMPTY STRİNG THİNGS

```
<%= @user.city || @user.state || "Unknown" %>
```

=> “”

IF CITY IS EMPTY “” IT WILL PRINT “”



```
city = @user.city if @user.city.present?
state = @user.state if @user.state.present?

<%= city || state || "Unknown" %>
```

EMPTY STRİNG THİNGS

```
<%= @user.city || @user.state || "Unknown" %>
```

```
<%= @user.city.presence || @user.state.presence || "Unknown" %>
```

EMPTY STRİNG THİNGS

```
<%= @user.city.titleize || "Unknown" %>
```

UNDEFINED METHOD `TITLEIZE' FOR NIL:NILCLASS

CITY IS NIL

EMPTY STRİNG THİNGS

```
<% if @user.city %>
   <%= @user.city.titleize %>
<% else %>
   Unknown
<% end %>
```

```
<%= @user.city ? @user.city.titleize : "Unknown" %>
```

```
<%= @user.city.try(:titleize) || "Unknown" %>
```

THE EXAMPLE


KeithBarrett Keith Barrett
Just got the Twitter "Something is technically wrong" screen. Haven't seen that in ages. Exactly what is that creature w/ the claws & mask?
10 minutes ago Favorite Reply
timbray Tim Bray by whiteafrican
Death to hashbangs! http://goo.gl/uDm2b
21 hours ago Unfavorite Retweet Reply
Sephora_ Sephora
Too hungry to figure out what to whip up for dinner.....shh growly stomach
11 minutes ago

ROCK YOUR BLOCK HELPERS

*/app/views/tweets/**index.html.erb***

```
<% @presenter.tweets.each do |tweet| %>
  <div id="tweet_<%= tweet.id %>"
       class="<%= 'favorite' if tweet.is_a_favorite?(current_user) %>">
    <%= tweet.status %>
  </div>
<% end %>
```

ROCK YOUR BLOCK HELPERS

*/app/views/tweets/**index.html.erb***

```
<% @presenter.tweets.each do |tweet| %>
  <%= tweet_div_for(tweet, current_user) do %>
    <%= tweet.status %>
  <% end %>
<% end %>
```

*/app/helpers/**tweets_helper.rb***

```
def tweet_div_for(tweet, user, &block)
  klass = 'favorite' if tweet.is_a_favorite?(user)

  content_tag tweet, :class => klass do
    yield
  end
end
```

```
id="tweet_<%= tweet.id %>"
```

YIELD TO THE CONTENT_FOR

*/app/views/layouts/**application.html.erb***

NEED TO INSERT CONTENT HERE!

```
<!DOCTYPE html>
<html>
<body>
  <h1>Twitter</h1>
  <% if flash[:notice] %>
    <span style="color: green"><%= flash[:notice] %></span>
  <% end %>
  <%= yield %>
</body>
</html>
```

YIELD TO THE CONTENT_FOR

*/app/views/layouts/**application.html.erb***

```
<!DOCTYPE html>
<html>
<body>
  <h1>Twitter</h1>

  <%= yield :sidebar %>

  <% if flash[:notice] %>
    <span style="color: green"><%= flash[:notice] %></span>
  <% end %>
  <%= yield %>
</body>
</html>
```

*/app/views/tweets/**index.html.erb***

```
<% content_for(:sidebar) do %>
  ... html here ...
<% end %>
```

WHAT IF ALL ACTIONS IN THE TWEET CONTROLLER NEED THE SIDEBAR?

*/app/views/layouts/**application.html.erb***

```
<%= yield :sidebar %>

<% if flash[:notice] %>
  <span style="color: green"><%= flash[:notice] %></span>
<% end %>
<%= yield %>
```

3

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  layout 'with_sidebar'
end
```

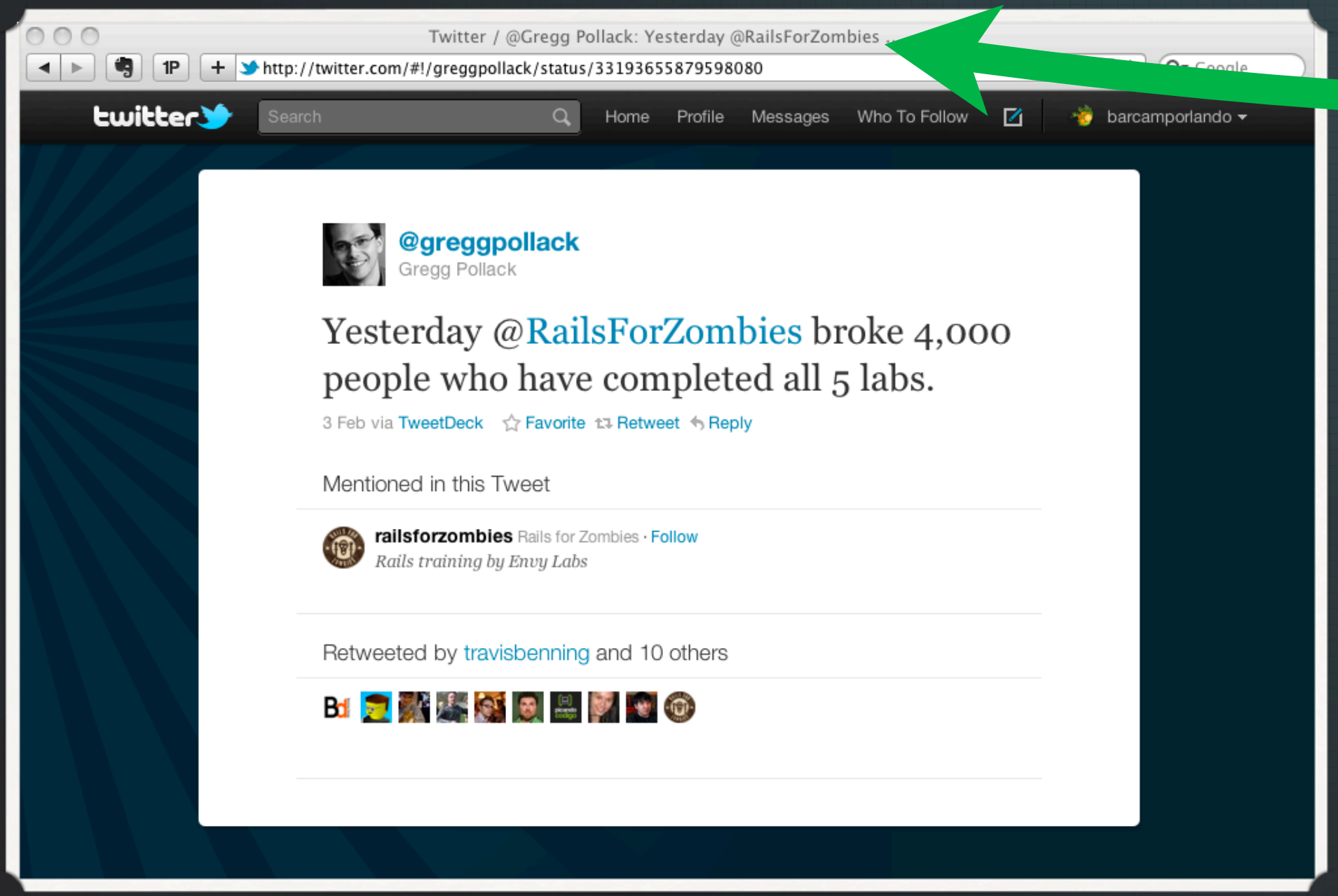
1

*/app/views/layouts/**with_sidebar.html.erb***

```
<% content_for(:sidebar) do %>
  ... html here ...
<% end %>
<%= render :file => 'layouts/application' %>
```

2

THE EXAMPLE


Twitter / @Gregg Pollack: Yesterday @RailsForZombies
http://twitter.com/#!/greggpollack/status/33193655879598080
twitter
Search
Home
Profile
Messages
Who To Follow
barcamporlando
@greggpollack
Gregg Pollack
Yesterday @RailsForZombies broke 4,000 people who have completed all 5 labs.
3 Feb via TweetDeck
Favorite
Retweet
Reply
Mentioned in this Tweet
railsforzombies Rails for Zombies · Follow
Rails training by Envy Labs
Retweeted by travisbenning and 10 others

META YIELD

*/app/views/layouts/**application.html.erb***

```
<!DOCTYPE html>
<html>
<head>
  <title>Twitter <%= @title %></title>
  <meta name="description"
        content="<%= @description || "The best way ..." %>">
  <meta name ="keywords"
        content="<%= @keywords || "social,tweets ..." %>">
  ...
```

*/app/controllers/**tweets_controller.rb***

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(params[:id])
    @title = @tweet.user.name
    @description = @tweet.status
    @keywords = @tweet.hash_tags.join(",")
  end
end
```

CLUTTERING YOUR CONTROLLER
& POLLUTING WITH VIEW CONCERNS

META YİELD

*/app/views/layouts/**application.html.erb***

```
<!DOCTYPE html>
<html>
<head>
  <title>Twitter <%= yield(:title) %></title>
  <meta name="description"
        content="<%= yield(:description) || "The best way ..." %>">
  <meta name ="keywords"
        content="<%= yield(:keywords) || "social,tweets ..." %>">
  ...
```

*/app/views/tweets/**show.html.erb***

```
<%
content_for(:title, @tweet.user.name)
content_for(:description, @tweet.status)
content_for(:keywords, @tweet.hash_tags.join(","))
%>
```

META YIELD

*/app/helpers/**application_helper.rb***

```ruby
def title(title)
  content_for(:title, title)
end

def description(description)
  content_for(:description, description)
end

def keywords(keywords)
  content_for(:keywords, keywords)
end
```

*/app/views/tweets/**show.html.erb***

```erb
<%
  title       @tweet.user.name
  description @tweet.status
  keywords    @tweet.hash_tags.join(",")
%>
```