An Introduction to Rails

Episode #1

with Gregg Pollack

Prerequisites:

TryRuby.org


Episode #1
Deep in the CRUD
BORN
JUNE 19, 1880
DIED OCT. 15, 1964

twitter

FOR ZOMBIES!!

Home Profile Find People Settings Help Sign out

What's happening? 140

greggpollack
1,520 tweets

148 following 3,028 followers 357 listed

Who to follow view all

mojombo
Tom Preston-Werner
Follow

bryanl
Bryan Liles (ツ)
Follow

Home
@greggpollack
Direct Messages 113
Favorites
Retweets

Search

Lists
greggs-favorites
New list | View all

Trending: Worldwide
Change
Taylor Hotner
Teen Choice Awards
Ted Stevens
#wheniwaslittle
#followerquestion
Ramadhan
#annoyingquestion

CRUD

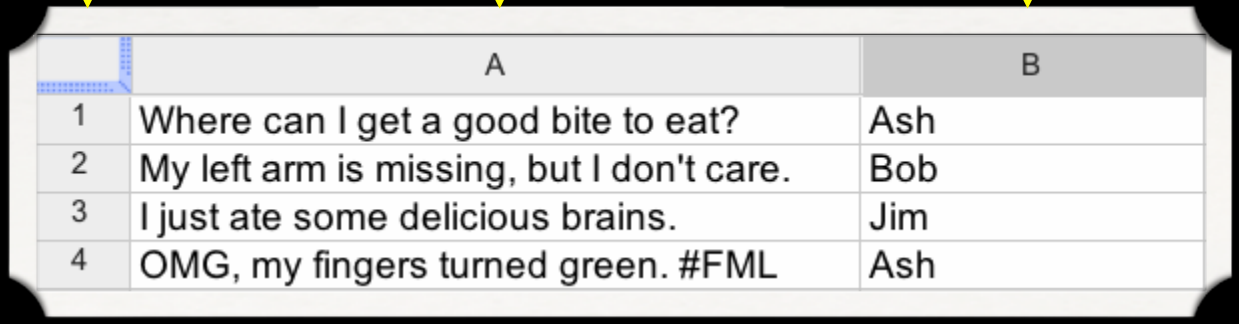
Columns
(we have 3)

tweets

	A	B
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

Rows
(we have 4)

id status zombie



	A	B
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

ZOMBIE CHALLENGE #1

Retrieve a hash of the tweet with id = 3

RESULT

```
b = { :status => "I just ate some delicious brains",
      :zombie => "Jim" }
```

Hash

CRUD

Series of key value pairs

```
b = { :status => "I just ate some delicious brains",
      :zombie => "Jim" }
```

```
puts b[:status]
```

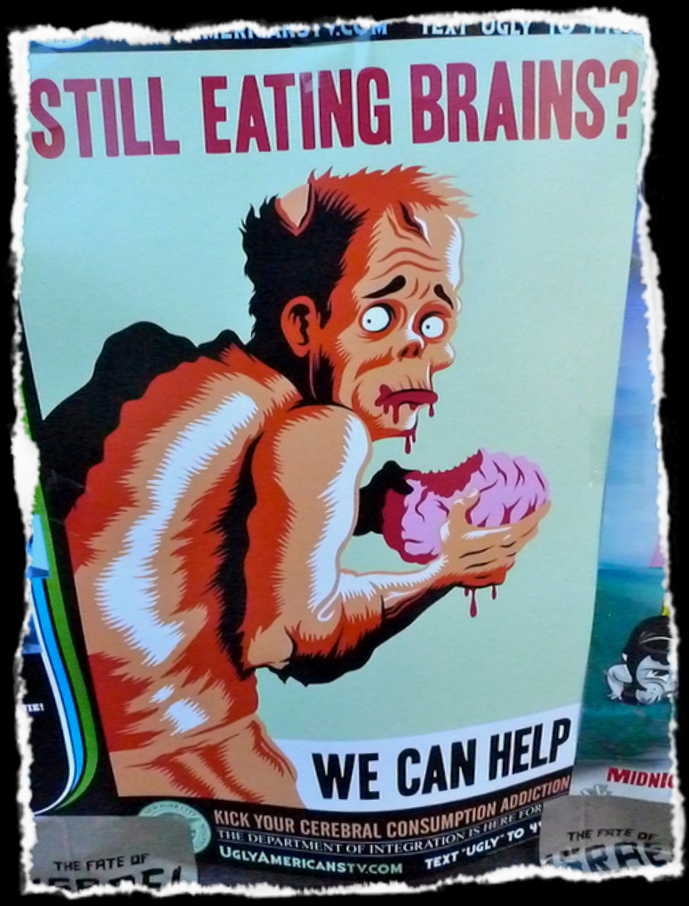
```
"I just ate some delicious brains"
```

```
puts b[:zombie]
```

```
"Jim"
```

```
puts b[:zombie] + " said " + b[:status]
```

```
"Jim said I just ate some delicious brains"
```



CRUD

tweets

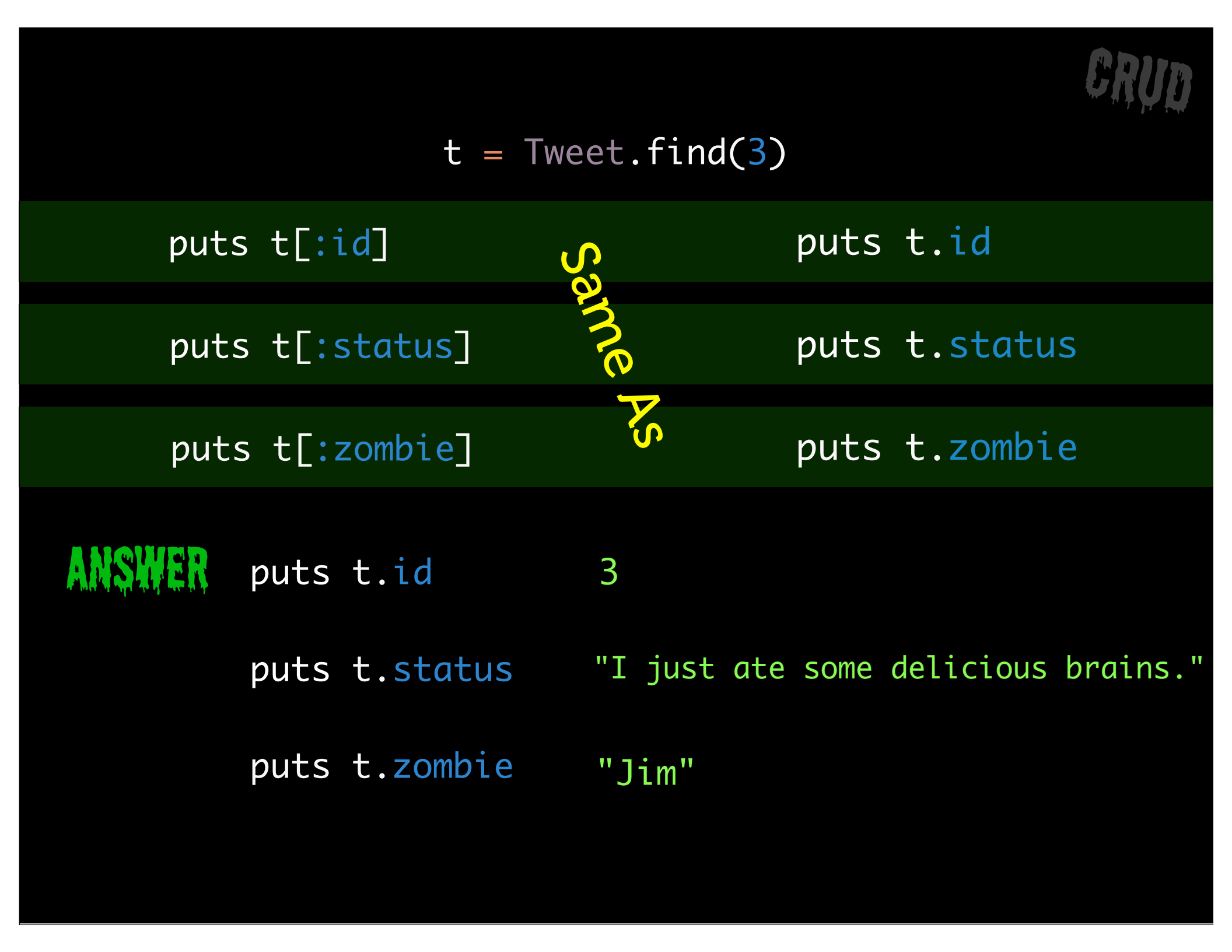
id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

ZOMBIE CHALLENGE #1

Retrieve a hash of the tweet with id = 3

ANSWER

```
t = Tweet.find(3)

puts t[:id]      3

puts t[:status]  "I just ate some delicious brains."

puts t[:zombie]  "Jim"
```


CRUD
t = Tweet.find(3)
puts t[:id]
puts t[:status]
puts t[:zombie]
Same As
puts t.id
puts t.status
puts t.zombie
ANSWER
puts t.id 3
puts t.status "I just ate some delicious brains."
puts t.zombie "Jim"

CRUD

```
t = Tweet.find(3)
```

tweets

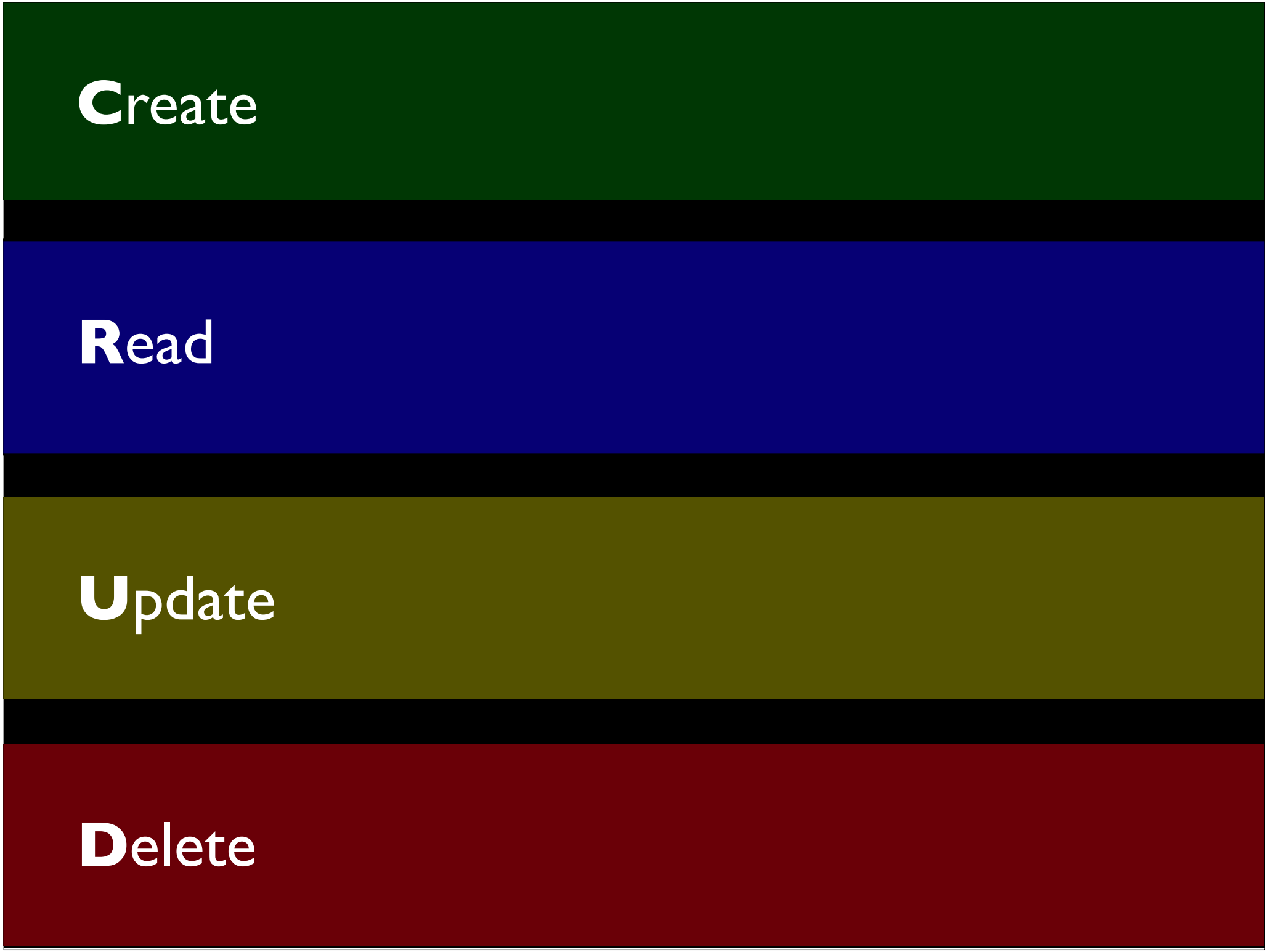
id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

C

R

U

D


Create
Read
Update
Delete

Create

```
t = Tweet.new
t.status = "I <3 brains."
t.save
```

Read

```
Tweet.find(3)
```

Update

```
t = Tweet.find(3)
t.zombie = "EyeballChomper"
t.save
```

Delete

```
t = Tweet.find(3)
t.destroy
```

Create

Syntax

```
t = Tweet.new
t.status = "I <3 brains."
t.zombie = "Jim"
t.save
```

Notice we don't set the id.
The id gets set manually for us.

Alternate Syntax

```
t = Tweet.new(:status => "I <3 brains", :zombie => "Jim")
t.save
```

```
Tweet.create(:status => "I <3 brains", :zombie => "Jim")
```

Read

Syntax

```
Tweet.find(2)                    # Returns a single item
Tweet.find(3, 4, 5)              # Returns an array
Tweet.first                      # Returns the first tweet
Tweet.last                       # Returns the last tweet
Tweet.all                        # Returns all the tweets
Tweet.count                      # Returns number of tweets
Tweet.order(:zombie)             # All ordered by zombie
Tweet.limit(10)                  # Only 10 tweets
Tweet.where(:zombie => "ash")    # Only tweets by Ash
Tweet.where(:zombie => "ash").order(:zombie).limit(10)
```

method chaining

Update

Syntax

```
t = Tweet.find(3)
t.zombie = "EyeballChomper"
t.save
```

```
t = Tweet.find(2)
t.attributes = {
            :status => "Can I munch your eyeballs?",
            :zombie => "EyeballChomper"
            }
t.save
```

```
t = Tweet.find(2)
t.update_attributes(
            :status => "Can I munch your eyeballs?",
            :zombie => "EyeballChomper"
            )
```

Delete

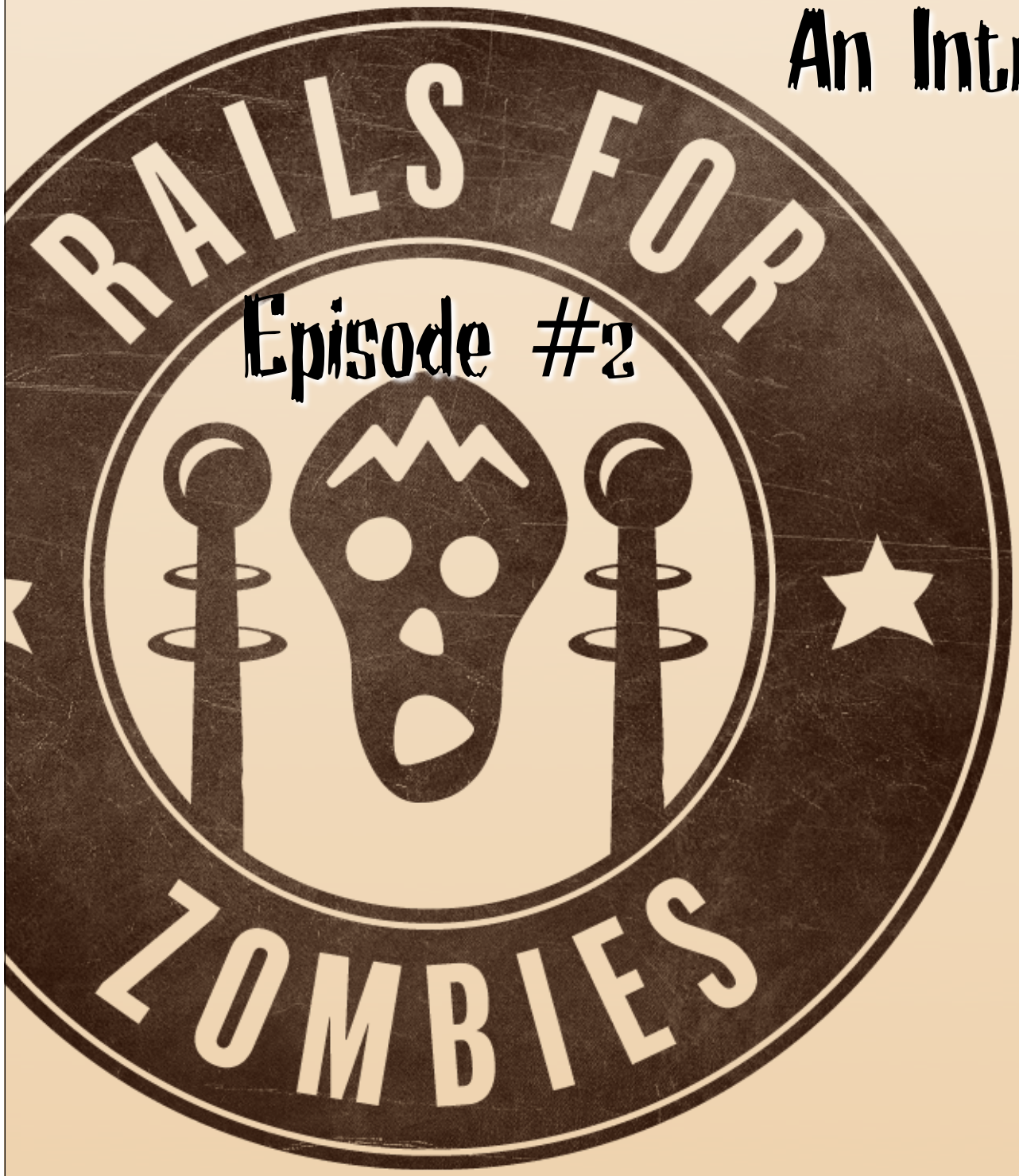
Syntax

```
t = Tweet.find(2)
t.destroy
```

```
Tweet.find(2).destroy
```

```
Tweet.destroy_all
```

ZOMBIE LAB 1



An Introduction to Rails

Episode #2

with Gregg Pollack

Chapter 2
Models taste like chicken

MODELS

```
t = Tweet.find(3)
```

```
app/models/tweet.rb
class Tweet < ActiveRecord::Base

end
```

tweets

id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

MODELS

Tweet

app/models/tweet.rb

```
class Tweet < ActiveRecord::Base

end
```

Maps the class to the table

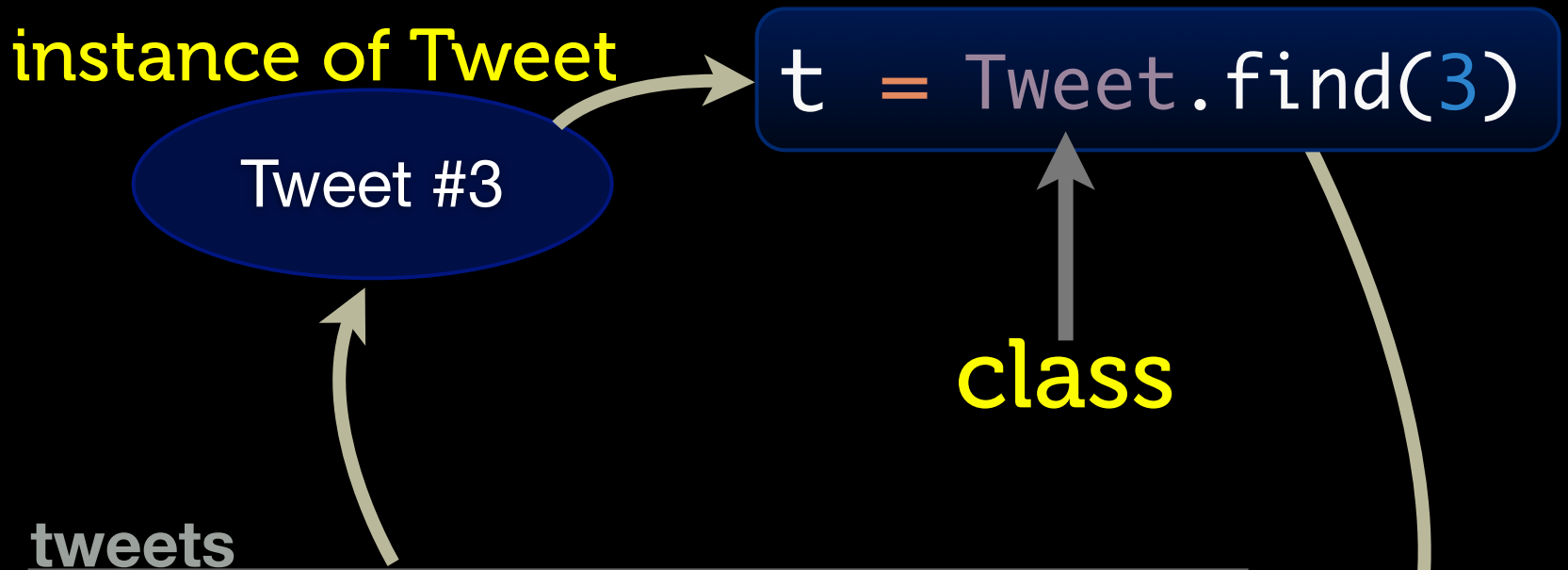
tweets

id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

MODELS

app/models/tweet.rb

```
class Tweet < ActiveRecord::Base

end
```



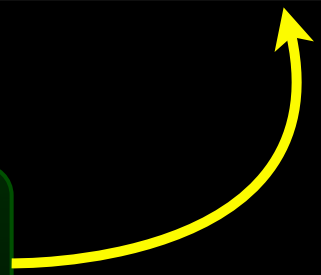
tweets

id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

Validations?

id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash
5		

```
t = Tweet.new
t.save
```



VALIDATIONS

app/models/tweet.rb

```ruby
class Tweet < ActiveRecord::Base

  validates_presence_of :status

end
```

```
> t = Tweet.new
                => #<Tweet id: nil, status: nil, zombie: nil>
> t.save
                => false
> t.errors
                => {:status=>["can't be blank"]}
> t.errors[:status]
                => "can't be blank"
```

VALIDATIONS

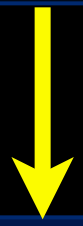
```
validates_presence_of :status

validates_numericality_of :fingers

validates_uniqueness_of :toothmarks

validates_confirmation_of :password

validates_acceptance_of :zombification

validates_length_of :password, :minimum => 3

validates_format_of :email, :with => /regex/i

validates_inclusion_of :age, :in => 21..99

validates_exclusion_of :age, :in => 0...21,
                       :message => “Sorry you must be over 21”
```

VALIDATIONS

Attribute

Validation

```
validates :status, :presence => true
validates :status, :length => { :minimum => 3 }
```



```
validates :status, :presence => true, :length => { :minimum => 3 }
```

```
:presence => true
:uniqueness => true
:numericality => true
:length => { :minimum => 0, :maximum => 2000 }
:format => { :with => /.*/ }
:inclusion => { :in => [1,2,3] }
:exclusion => { :in => [1,2,3] }
:acceptance => true
:confirmation => true
```


RELATIONSHIPS
Because they always travel in packs

RELATIONSHIPS

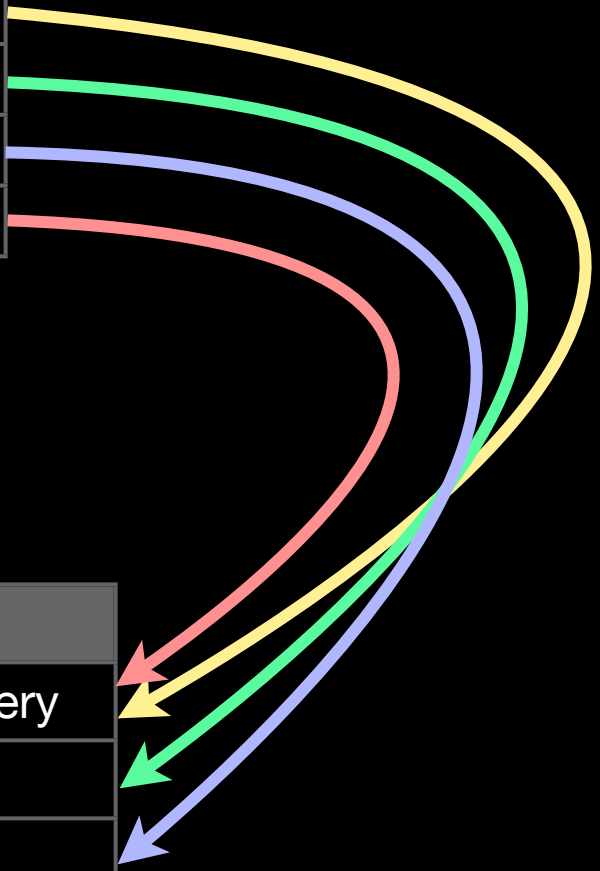
tweets

id	status	zombie
1	Where can I get a good bite to eat?	Ash
2	My left arm is missing, but I don't care.	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

RELATIONSHIPS

tweets

id	status
1	Where can I get a good bite to eat?
2	My left arm is missing, but I don't care.
3	I just ate some delicious brains.
4	OMG, my fingers turned green. #FML



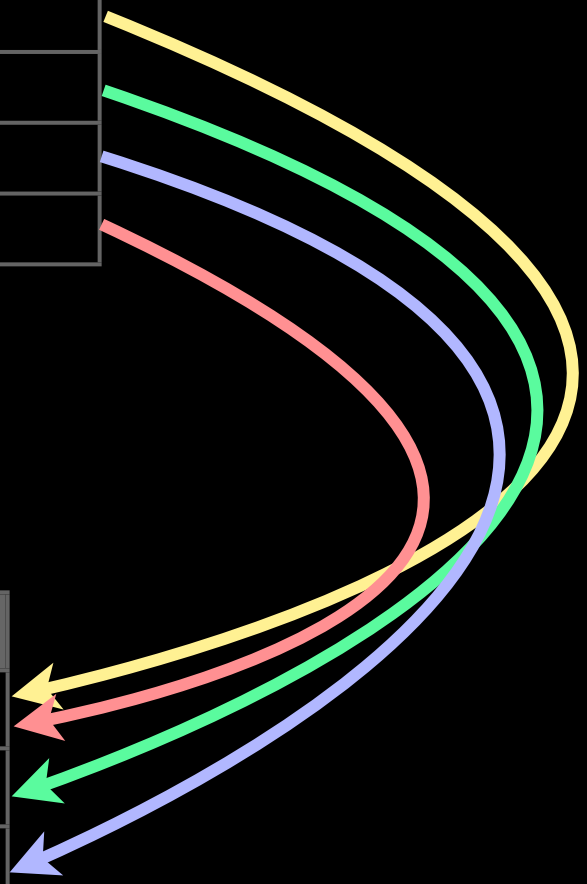
zombies

id	name	graveyard
1	Ash	Glen Haven Memorial Cemetery
2	Bob	Chapel Hill Cemetery
3	Jim	My Father’s Basement

RELATIONSHIPS

tweets

id	status	zombie_id
1	Where can I get a good bite to eat?	1
2	My left arm is missing, but I don't care.	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1



zombies

id	name	graveyard
1	Ash	Glen Haven Memorial Cemetery
2	Bob	Chapel Hill Cemetery
3	Jim	My Father's Basement

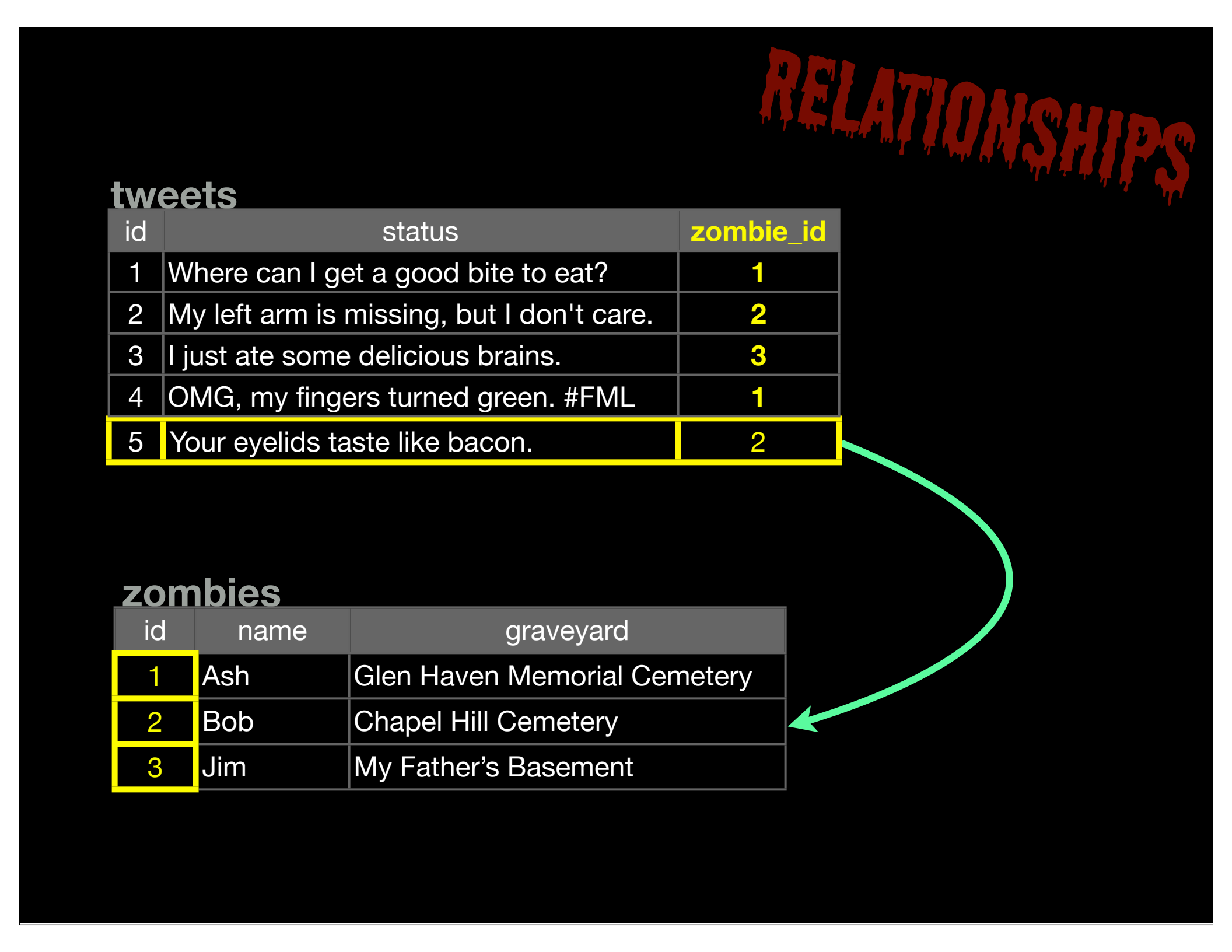
RELATIONSHIPS

tweets

id	status	zombie_id
1	Where can I get a good bite to eat?	1
2	My left arm is missing, but I don't care.	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1
5	Your eyelids taste like bacon.	2

zombies

id	name	graveyard
1	Ash	Glen Haven Memorial Cemetery
2	Bob	Chapel Hill Cemetery
3	Jim	My Father's Basement



A Tweet belongs to a Zombie

app/models/tweet.rb

```
class Tweet < ActiveRecord::Base

  belongs_to :zombie

end
```

Singular

A Zombie has many Tweets

app/models/zombie.rb

```
class Zombie < ActiveRecord::Base

  has_many :tweets

end
```

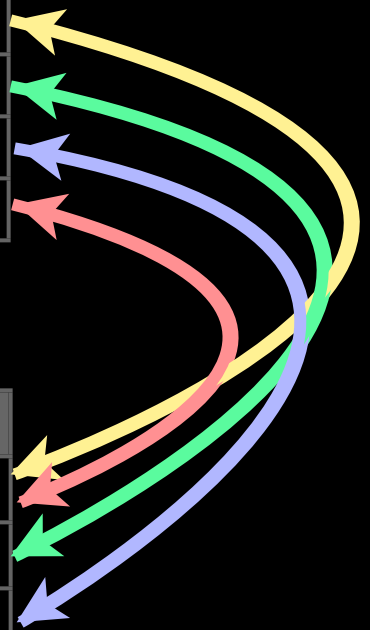
Plural

tweets

id	status	zombie_id
1	Where can I get a good bite to eat?	1
2	My left arm is missing, but I don't care.	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1

zombies

id	name	graveyard
1	Ash	Glen Haven Memorial Cemetery
2	Bob	Chapel Hill Cemetery
3	Jim	My Father's Basement



A Tweet belongs to a Zombie


```
> z = Zombie.find(2)
 => #<Zombie id: 2, name: "Bob", graveyard: "Chapel Hill Cemetery">

> t = Tweet.create(:status => "Your eyelids taste like bacon.", :zombie => z)
 => #<Tweet id: 5, status: "Your eyelids taste like bacon.", zombie_id: 2>

> t.zombie
 => #<Zombie id: 2, name: "Bob", graveyard: "Chapel Hill Cemetery">

> t.zombie.name
 => "Bob"

> ash = Zombie.find(1)
  => #<Zombie id: 1, name: "Ash", graveyard: "Glen Haven Cemetery">

> ash.tweets.count
  => 4

> ash.tweets
  => [#<Tweet id: 1, status: "Where can I get a good bite to eat?",
zombie_id: 1>, #<Tweet id: 4, status: "OMG, my fingers turned green. #FML",
zombie_id: 1>]
```

ZOMBIE LAB 2

An Introduction to Rails

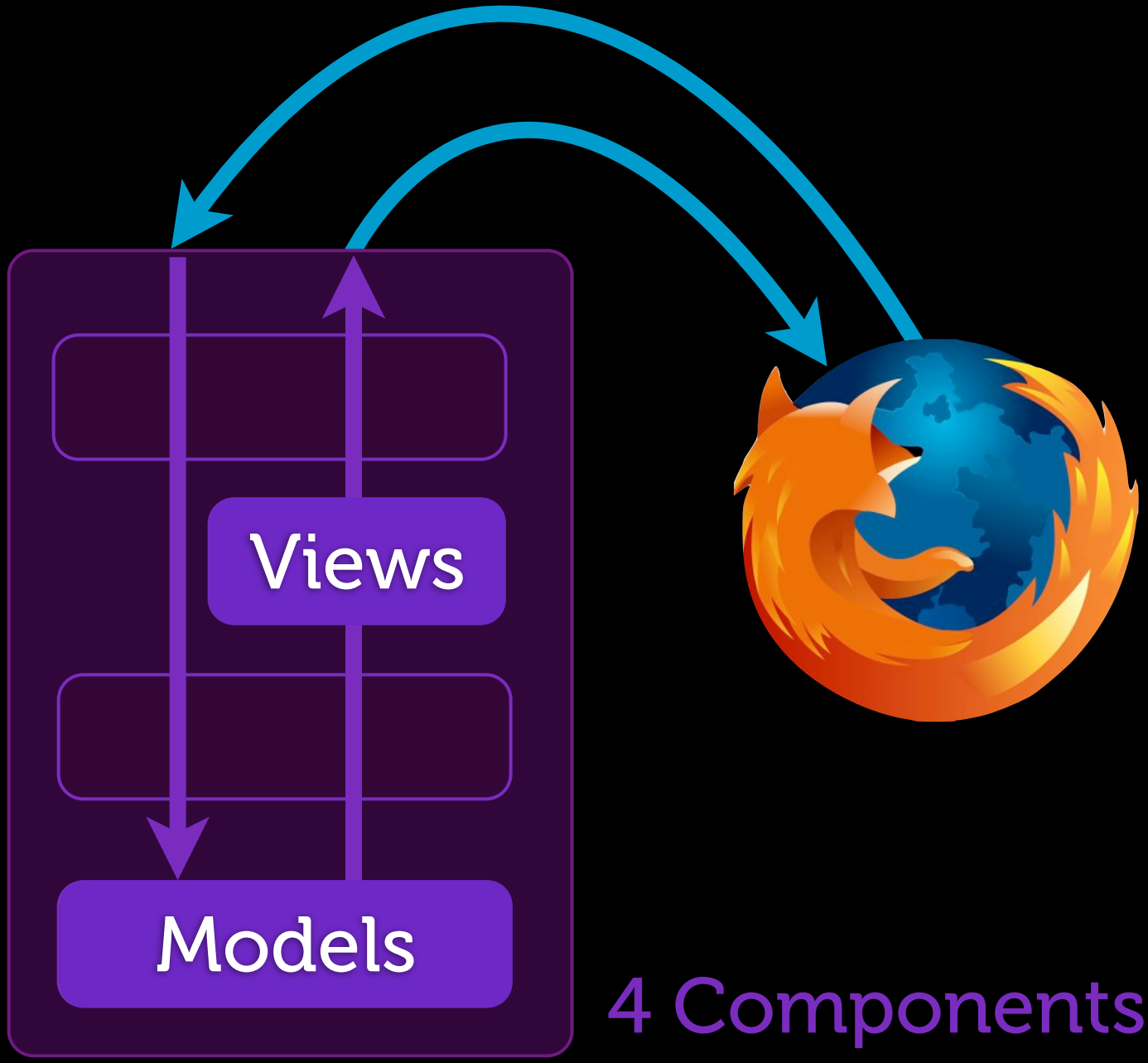
Episode #3

with Gregg Pollack

Chapter 3
The View ain't always pretty

Our Application Stack

VIEWS



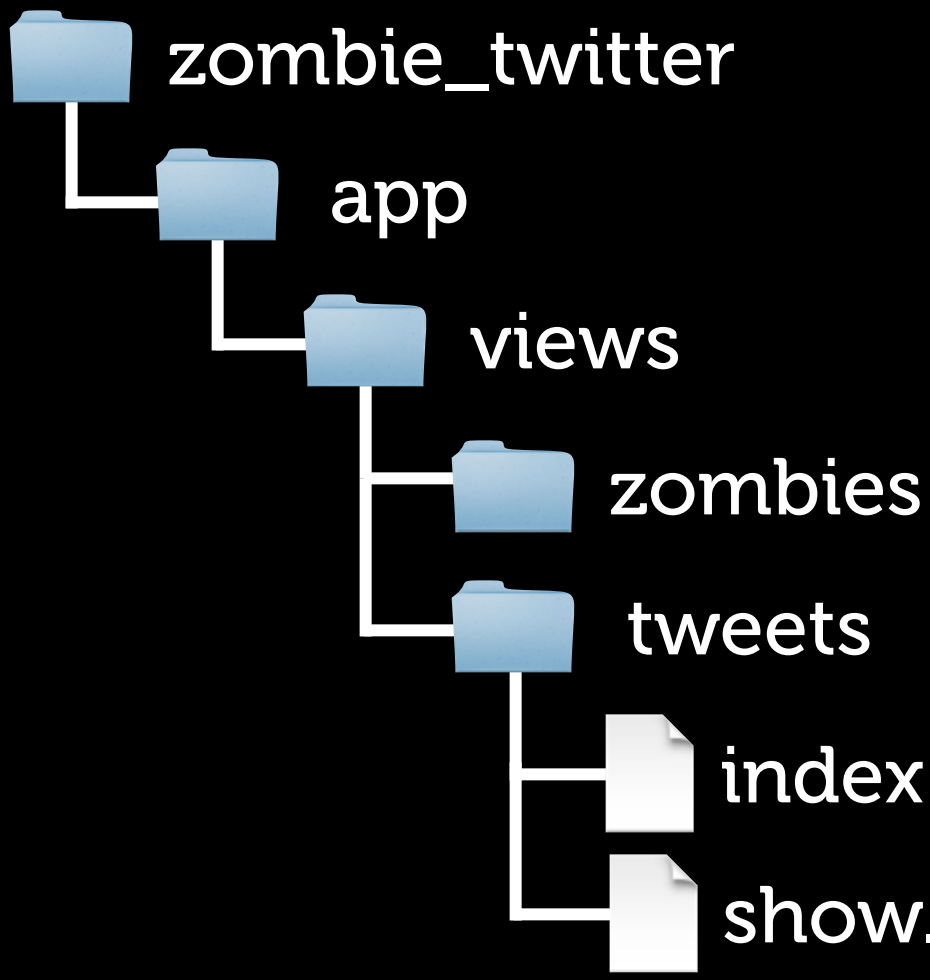
4 Components

~~Edible Rotting Bodies~~

VIEWS

Embedded Ruby

Ruby inside HTML



List all tweets

View a tweet

Show a tweet

`<% ... %>`

Evaluate Ruby

`<%= ... %>`

Eval and print result

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

```
<!DOCTYPE html>
<html>
 <head><title>Twitter for Zombies</title></head>
<body>
  <img src="/images/twitter.png" />

  <% tweet = Tweet.find(1) %>

  <h1><%= tweet.status %></h1>

  <p>Posted by <%= tweet.zombie.name %></p>

</body></html>
```

Show a tweet

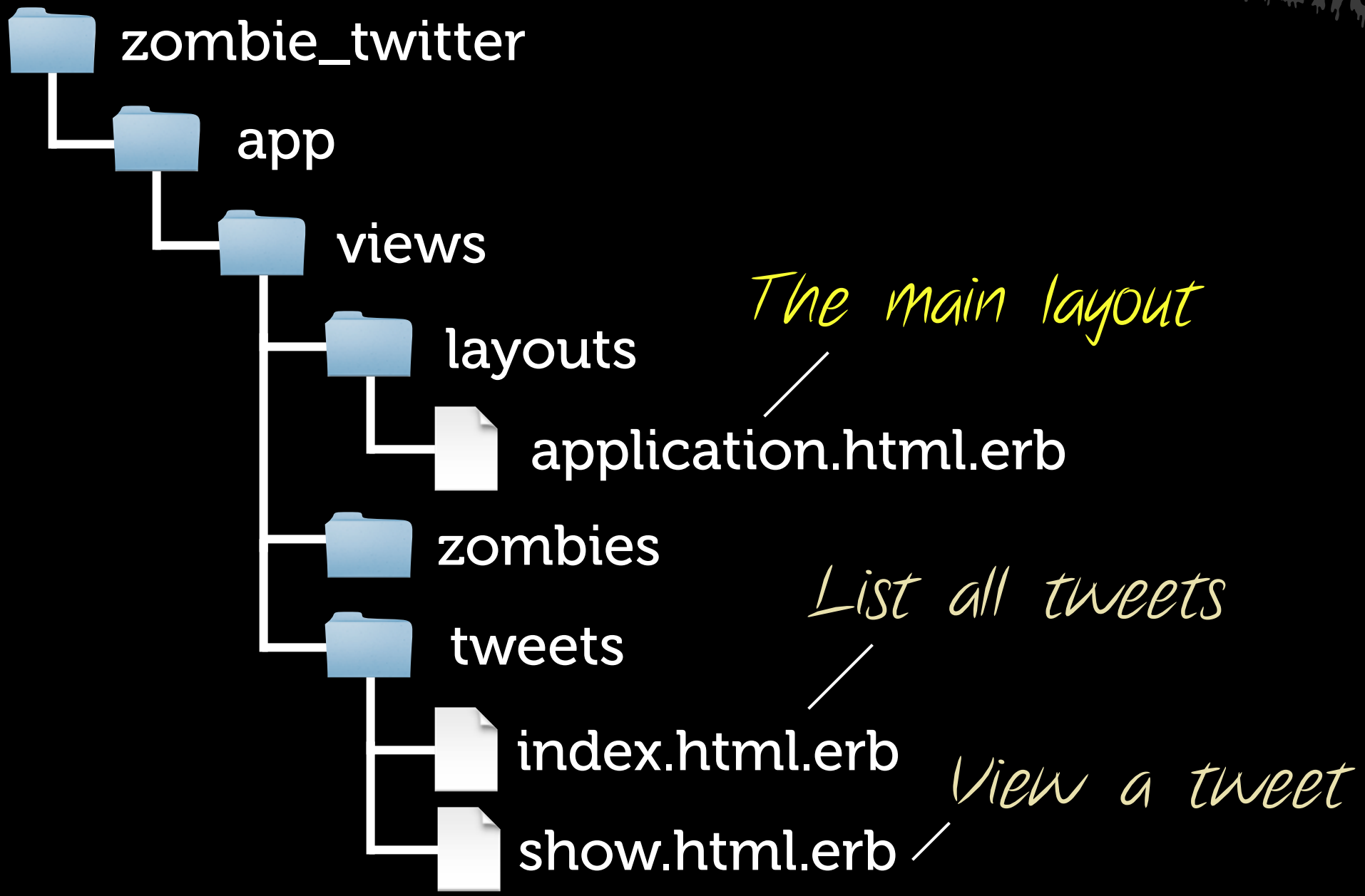
/app/views/layouts/application.html.erb

```
<!DOCTYPE html>
<html>
 <head><title>Twitter for Zombies</title></head>
<body>
  <img src="/images/twitter.png" />

  <%= yield %>

</body></html>
```

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

```
<% tweet = Tweet.find(1) %>

<h1><%= tweet.status %></h1>

<p>Posted by <%= tweet.zombie.name %></p>
```

VIEWS


zombie_twitter
app
views
layouts
application.html.erb
zombies
tweets
index.html.erb
show.html.erb
The main layout
List all tweets
View a tweet

Additional Layout Components

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Twitter for Zombies</title>
</head>
<body>
  <img src="/images/twitter.png" />

  <%= yield %>

</body></html>
```

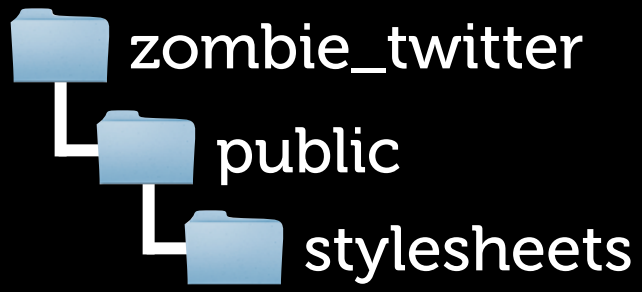
Additional Layout Components

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Twitter for Zombies</title>
  <%= stylesheet_link_tag :all %>
  <%= javascript_include_tag :defaults %>
  <%= csrf_meta_tag %>
</head>
<body>
  <img src="/images/twitter.png" />

  <%= yield %>

</body></html>
```

Additional Layout Components

```
<%= stylesheet_link_tag :all %>
```



Include all stylesheets

Renders

```
<link href="/stylesheets/scaffold.css" media="screen" rel="stylesheet" type="text/css" />
```

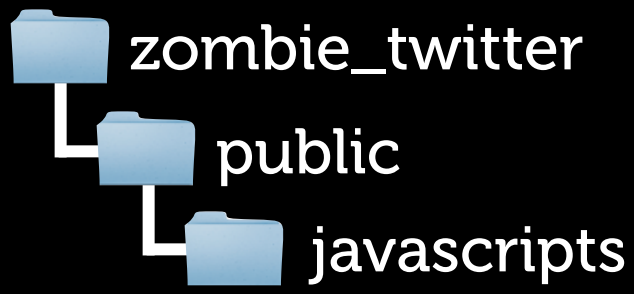
```
<%= javascript_include_tag :defaults %>
```

```
<%= csrf_meta_tag %>
```

Additional Layout Components

```
<%= stylesheet_link_tag :all %>
```

```
<%= javascript_include_tag :defaults %>
```



Include default JS

Renders

```
<script src="/javascripts/prototype.js" t
<script src="/javascripts/effects.js" typ
<script src="/javascripts/dragdrop.js" t
<script src="/javascripts/controls.js" ty
<script src="/javascripts/rails.js" type="text/
<script src="/javascripts/application.js" type="text/javascr
```

Prototype Javascript Framework

```
<%= csrf_meta_tag %>
```

Additional Layout Components

```
<%= stylesheet_link_tag :all %>
```

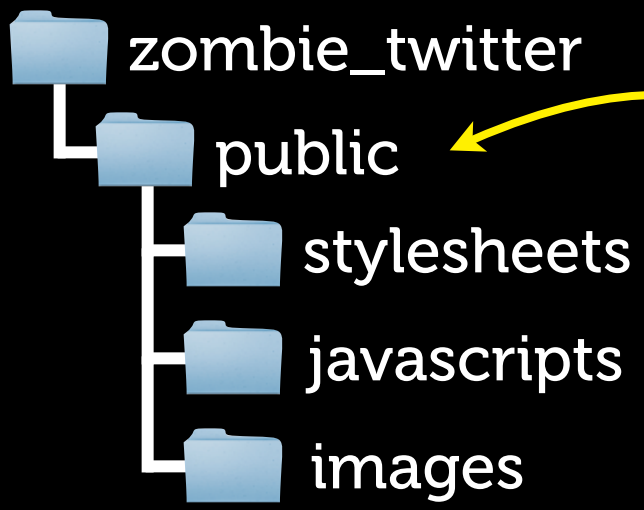
```
<%= javascript_include_tag :defaults %>
```

```
<%= csrf_meta_tag %>
```

Automatically adds this to forms

Root path and images

http://ZombieTwitter.com/[something]



If exists in /public use it, otherwise go to Rails

Example

```
<img src="/images/twitter.png" />
```

Adding a Link

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

```
...
<p>Posted by <%= tweet.zombie.name %></p>
```

Make zombie a link

```
<%= link_to tweet.zombie.name , zombie_path(tweet.zombie)  %>
```

Link Text — Link Path (URL)

Renders → `<a href="/zombies/1">Ash</a>`

We can also write as

```
<%= link_to tweet.zombie.name, tweet.zombie %>
```

Link Text — Object to Show

Method

```
link_to
```

What options can you use with it?

1. Look in the source

Command Line

```
git clone http://github.com/rails/rails.git
cd rails
```

Open your editor and search for “def link_to”

Method

`link_to`

What options can you use with it?

2. Look at api.rubyonrails.org

(and search for link_to)


Files
actionmailer/CHANGELOG
actionmailer/README
actionmailer/lib/action_mailer/base.rb
actionpack/CHANGELOG
actionpack/README
Classes
ActionController::Assertions
ActionController::Assertions::DomAssertions
ActionController::Assertions::ModelAssertions
ActionController::Assertions::ResponseAssertions
ActionController::Assertions::RoutingAssertions
ActionController::Assertions::SelectorAssertions
ActionController::Assertions::TagAssertions
ActionController::Base
ActionController::Benchmarking::ClassMethods
ActionController::CGIHandler
ActionController::CGIHandler::ProperStream
ActionController::Caching
ActionController::Caching::Actions
ActionController::Caching::Actions::ClassMethods
ActionController::Caching::Fragments
ActionController::Caching::Pages
ActionController::Caching::Pages::ClassMethods
ActionController::Caching::Sweeping
ActionController::CgiExt
ActionController::CgiExt::Stdinput
ActionController::Cookies
ActionController::Dispatcher
ActionController::Failsafe
ActionController::Filters::ClassMethods
ActionController::Flash
ActionController::Flash::FlashHash
ActionController::Helpers::ClassMethods
ActionController::Http
ActionController::Http::Headers
ActionController::HttpAuthentication
ActionController::HttpAuthentication::Basic
ActionController::HttpAuthentication::Digest
ActionController::Integration::Runner
ActionController::Integration::Session
ActionController::IntegrationTest
ActionController::Layout::ClassMethods
ActionController::MiddlewareStack
ActionController::MiddlewareStack::Middleware
ActionController::MimeResponds::InstanceMethods
ActionController::ParamsParser
label_tag (ActionView::Helpers::FormTagHelper)
last (ActiveRecord::NamedScope::Scope)
last (ActiveRecord::Base)
last_modified (ActionController::Response)
last_modified= (ActionController::Response)
last_modified? (ActionController::Response)
layout (ActionController::Layout::ClassMethods)
length (ActiveSupport::Multibyte::Chars)
length (ActiveRecord::Errors)
length (ActiveResource::Errors)
link_to (ActionView::Helpers::UrlHelper)
link_to_function (ActionView::Helpers::JavaScriptHe
link_to_if (ActionView::Helpers::UrlHelper)
link_to_unless (ActionView::Helpers::UrlHelper)
link_to_unless_current (ActionView::Helpers::UrlHel
links (RecursiveHTTPFetcher)
ljust (ActiveSupport::Multibyte::Chars)
load (Rails::Plugin)
load (Rails::GemDependency)
load (ActiveResource::Base)
link_to(*args, &block)
Creates a link tag of the given name using a URL created by the set of options. See the valid options in the documentation for url_for. It's also possible to pass a string instead of an options hash to get a link tag that uses the value of the string as the href for the link, or use :back to link
the referrer - a JavaScript back link will be used in place of a referrer if none exists. If nil is passed as a name, the link itself will become the na
Signatures
link_to(name, options = {}, html_options = nil)
link_to(options = {}, html_options = nil) do
name
end
Options
:confirm => 'question?' - This will add a JavaScript confirm prompt with the question specified. If the user accepts, the link is processed
normally, otherwise no action is taken.
:popup => true || array of window options - This will force the link to open in a popup window. By passing true, a default browser windo
will be opened with the URL. You can also specify an array of options that are passed-thru to JavaScripts window.open method.
:method => symbol of HTTP verb - This modifier will dynamically create an HTML form and immediately submit the form for processing u
the HTTP verb specified. Useful for having links perform a POST operation in dangerous actions like deleting a record (which search bots
follow while spidering your site). Supported verbs are :post, :delete and :put. Note that if the user has JavaScript disabled, the request w
fall back to using GET. If you are relying on the POST behavior, you should check for it in your controller's action by using the request obj
methods for post?, delete? or put?.
The html_options will accept a hash of html attributes for the link tag.
Note that if the user has JavaScript disabled, the request will fall back to using GET. If :href => '#' is used and the user has JavaScript disablec
clicking the link will have no effect. If you are relying on the POST behavior, your should check for it in your controller's action by using the req
object's methods for post?, delete? or put?.
You can mix and match the html_options with the exception of :popup and :method which will raise an ActionView::ActionViewError exception.
Examples
Because it relies on url_for, link_to supports both older-style controller/action/id arguments and newer RESTful routes. Current Rails style favo
RESTful routes whenever possible, so base your application on resources and use
link_to "Profile", profile_path(@profile)
=> Profile
or the even pithier
link_to "Profile", @profile
=> Profile
in place of the older more verbose, non-resource-oriented
link_to "Profile", :controller => "profiles", :action => "show", :id => @profile
=> Profile
Similarly,
link_to "Profiles", profiles_path
=> Profiles
is better than
You can use a bloc as well if your link target is hard to fit into the name parameter. ERb example:
http://api.rubyonrails.org

Method

`link_to`

What options can you use with it?

3. API Dock

(online Ruby and Rails docs)

http://apidock.com/rails

project

Ruby on Rails - Browse - Search

What's this?

Latest events

- stevo posted note Preserve order of elements within fields_for — about 7 hours ago
- himn1 posted note This can be useful for conditional validation — about 10 hours ago
- himn1 posted note Custom validator with i18n support — about 22 hours ago
- tokland posted note Anchor option — 8 days ago
- soulim posted note Mistake in example for Rails 3 — 9 days ago
- tordans posted note Nice german translation independet of structure of a sentence — 10 days ago
- bradwerth posted note Or maybe... — 15 days ago
- Imported version v3.0.0 — 15 days ago
- Git repository cloned from git://github.com/rails/rails.git — 15 days ago
- tokland posted note About the options argument — 15 days ago
- tom2cjp posted note using joins, group, having — 17 days ago

Project README

Welcome to Rails

Rails is a web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Control pattern.

Project details

- 759 notes
- 242 good notes
- 683 modules
- 938 classes
- 9659 methods

Show more project details

Latest good notes

- Use the current URL, with changes — by foliosus at 29 Sep
- This method will still rewrite all the values of the table — by railsmonk at 16 Sep
- Easy workaround for missing :through option — by szeryf at 27 Aug
- Passing optional arguments with defaults to a named_scope — by insane-dreamer at 25 Aug
- default_scope on create — by squil at 25 Aug
- Not really deprecated — by rxcfc at 23 Aug
- :conditions examples — by mihserf at 21 Aug
- Calling migrations within migrations — by RISCfuture at 20 Aug

All good notes - RSS
All notes - RSS

Top contributors of last 30 days

1. rgolkosky with 6 received thanks
2. leente with 2 received thanks
3. yonosoytu with 2 received thanks

Method

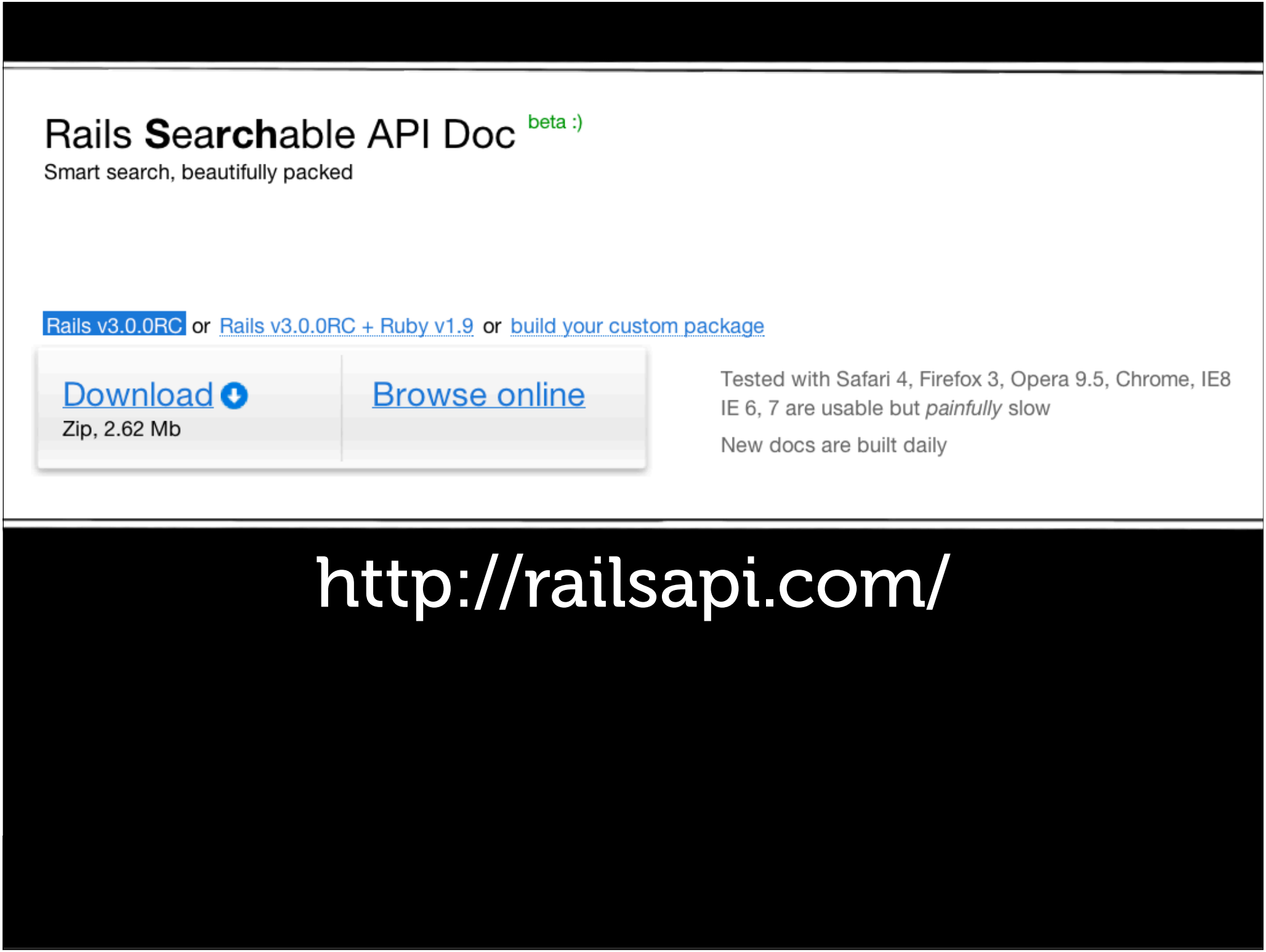
`link_to`

What options can you use with it?

4. Rails Searchable API Doc

(local download or online)

http://railsapi.com/


Rails Searchable API Doc
beta :)
Smart search, beautifully packed
Rails v3.0.0RC or Rails v3.0.0RC + Ruby v1.9 or build your custom package
Download
Zip, 2.62 Mb
Browse online
Tested with Safari 4, Firefox 3, Opera 9.5, Chrome, IE8
IE 6, 7 are usable but painfully slow
New docs are built daily
http://railsapi.com/

link_to

link_to(*args, &block)
ActionView::Helpers::UrlHelper
Creates a link tag of the given +name+ using a URL crea

link_to_function(*name, *args, &block)
ActionView::Helpers::JavaScriptHelper
Returns a link of the given +name+ that will trigger a Java

link_to_if(condition, name, options = {}, html_o
ActionView::Helpers::UrlHelper
Creates a link tag of the given +name+ using a URL crea

link_to_unless(condition, name, options = {}, ht
ActionView::Helpers::UrlHelper
Creates a link tag of the given +name+ using a URL crea

link_to_unless_current(name, options = {}, html
ActionView::Helpers::UrlHelper
Creates a link tag of the given +name+ using a URL crea

link_to(*args, &block)

Creates a link tag of the given `name` using a URL created by the set of `options`. See the valid options in the documentation for `url_for`. also possible to pass a string instead of an options hash to get a link tag that uses the value of the string as the href for the link, or use `:back` to link to the referrer - a JavaScript back link will be used in place of a referrer if none exists. If `nil` is passed as a name, the link itself will become the name.

Signatures

```
link_to(body, url, html_options = {})
  # url is a String; you can use URL helpers like
  # posts_path

link_to(body, url_options = {}, html_options = {})
  # url_options, except :confirm or :method,
  # is passed to url_for

link_to(options = {}, html_options = {}) do
  # name
end

link_to(url, html_options = {}) do
  # name
end
```

Options

- `:confirm => 'question?'` - This will allow the unobtrusive JavaScript driver to prompt with the question specified. If the user acce the link is processed normally, otherwise no action is taken.
- `:method => symbol of HTTP verb` - This modifier will dynamically create an HTML form and immediately submit the form for processing using the HTTP verb specified. Useful for having links perform a POST operation in dangerous actions like deleting a rec (which search bots can follow while spidering your site). Supported verbs are `:post`, `:delete` and `:put`. Note that if the user has JavaScript disabled, the request will fall back to using GET. If `:href => '#'` is used and the user has JavaScript disabled clicking link will have no effect. If you are relying on the POST behavior, you should check for it in your controller's action by using the reque object's methods for `post?`, `delete?` or `put?`.
- `:remote => true` - This will allow the unobtrusive JavaScript driver to make an Ajax request to the URL in question instead of follo the link. The drivers each provide mechanisms for listening for the completion of the Ajax request and performing JavaScript operati once they're complete

Examples

Because it relies on `url_for`, `link_to` supports both older-style controller/action/id arguments and newer RESTful routes. Current Rails favors RESTful routes whenever possible, so base your application on resources and use

```
link_to "Profile", profile_path(@profile)
# => <a href="/profiles/1">Profile</a>
```

or the even pithier

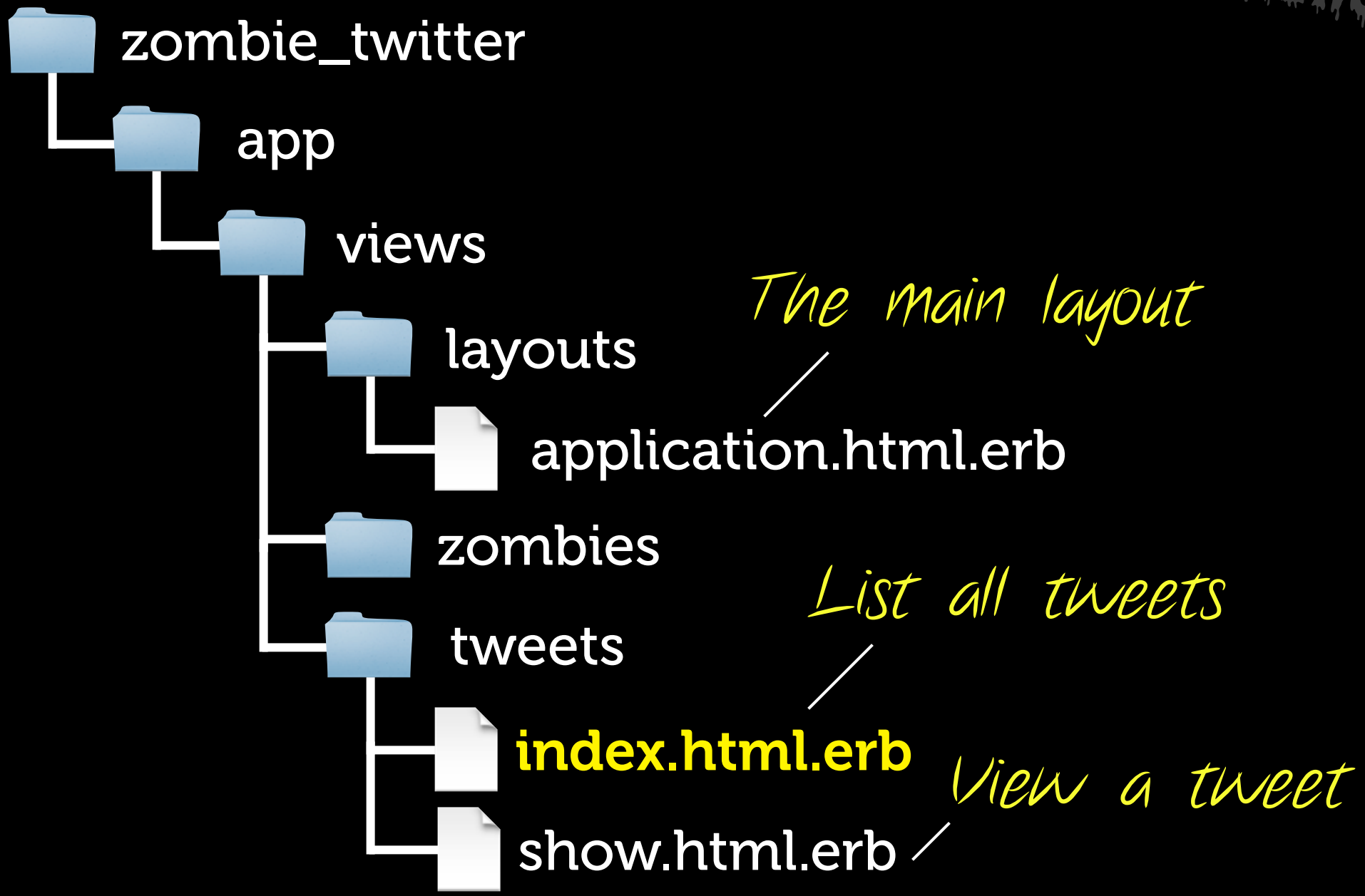
```
link_to "Profile", @profile
# => <a href="/profiles/1">Profile</a>
```

Options

- `:confirm => 'question?'` - This will allow the unobtrusive JavaScript driver to prompt with the question specified. If the user accepts, the link is processed normally, otherwise no action is taken.

```
<%= link_to @tweet.zombie.name, @tweet.zombie, :confirm => "Are you sure?" %>
```

VIEWS


zombie_twitter
app
views
layouts
application.html.erb
The main layout
zombies
tweets
index.html.erb
List all tweets
show.html.erb
View a tweet

List Tweets

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

```
<h1>Listing tweets</h1>

<table>
  <tr>
    <th>Status</th>
    <th>Zombie</th>
  </tr>

<% Tweet.all.each do |tweet| %>
  <tr>
    <td><%= tweet.status %></td>
    <td><%= tweet.zombie.name %></td>
  </tr>
<% end %>
</table>
```

What they return

`Tweet`	class
`Tweet.all`	array of tweets
`tweet`	single tweet

Create Links

VIEWS

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

```
<% Tweet.all.each do |tweet| %>
  <tr>
    <td><%= tweet.status %></td>
    <td><%= tweet.zombie.name %></td>
  </tr>
<% end %>
```

Create Links

VIEWS

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

```
<% Tweet.all.each do |tweet| %>
  <tr>
    <td><%= link_to tweet.status, tweet %></td>
    <td><%= tweet.zombie.name %></td>
  </tr>
<% end %>
```

Create Links

VIEWS

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

```
<% Tweet.all.each do |tweet| %>
  <tr>
    <td><%= link_to tweet.status, tweet %></td>
    <td><%= link_to tweet.zombie.name, tweet.zombie %></td>
  </tr>
<% end %>
```

Empty Table?

VIEWS


```
<% Tweet.all.each do |tweet| %>
   <tr>
     <td><%= link_to tweet.status, tweet %></td>
     <td><%= link_to tweet.zombie.name, tweet.zombie %></td>
   </tr>
<% end %>
```

Empty Table?

VIEWS

Listing tweets

Status Zombie

No tweets found

```
<% Tweet.all.each do |tweet| %>
   ...
<% end %>
```

Empty Table?

VIEWS


```
<% tweets = Tweet.all %>

<%          .each do |tweet| %>
   ...
<% end %>
```

Empty Table?

VIEWS


```
<% tweets = Tweet.all %>

<% tweets.each do |tweet| %>
   ...
<% end %>


<% if tweets.size == 0 %>
   <em>No tweets found</em>
<% end %>
```

Empty Table?

VIEWS

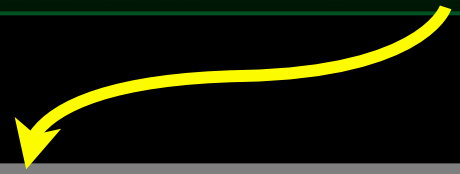

```
<% tweets = Tweet.all %>

<% tweets.each do |tweet| %>
   ...
<% end %>


<% if tweets.empty? %>
   <em>No tweets found</em>
<% end %>
```

Edit & Delete Links?

```
<% tweets.each do |tweet| %>
  <tr>
    <td><%= link_to tweet.status, tweet %></td>
    <td><%= link_to tweet.zombie.name, tweet.zombie %></td>
    <td><%= link_to "Edit", edit_tweet_path(tweet) %></td>
    <td><%= link_to "Delete", tweet, :method => :delete %></td>
  </tr>
<% end %>
```

All links for Tweets

```
<%= link_to "<link text>", <code> %>
```



Action	Code	The URL Generated
List all tweets	`tweets_path`	/tweets
New tweet form	`new_tweet_path`	/tweets/new

```
tweet = Tweet.find(1)
```

These paths need a tweet

Action	Code	The URL Generated
Show a tweet	`tweet`	/tweets/1
Edit a tweet	`edit_tweet_path(tweet)`	/tweets/1/edit
Delete a tweet	`tweet, :method => :delete`	/tweets/1

ZOMBIE LAB 3

An Introduction to Rails

Episode #4

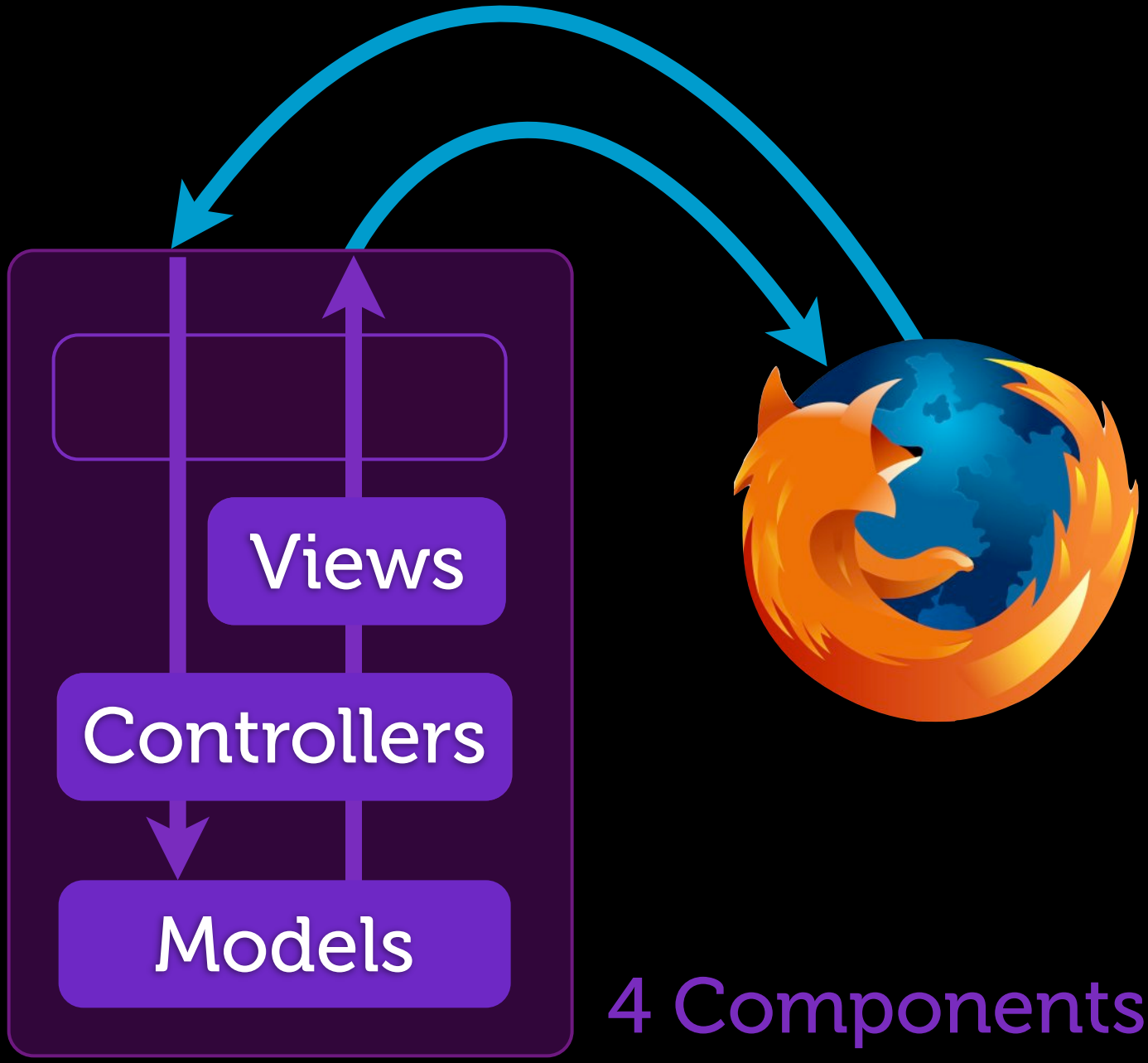
with Gregg Pollack

Chapter 4

Controllers (brains) must be eaten

Our Application Stack

VIEWS



4 Components

Show a tweet

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

```
<!DOCTYPE html>
<html>
 <head><title>Twitter for Zombies</title></head>
<body>
  <img src="/images/twitter.png" />

  <% tweet = Tweet.find(1) %>

  <h1><%= tweet.status %></h1>

  <p>Posted by <%= tweet.zombie.name %></p>

</body></html>
```

Request
/tweets/1

/app/controllers/tweets_controller.rb

Controller

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

```
<% tweet = Tweet.find(1) %>

<h1><%= tweet.status %></h1>

<p>Posted by <%= tweet.zombie.name %></p>
```

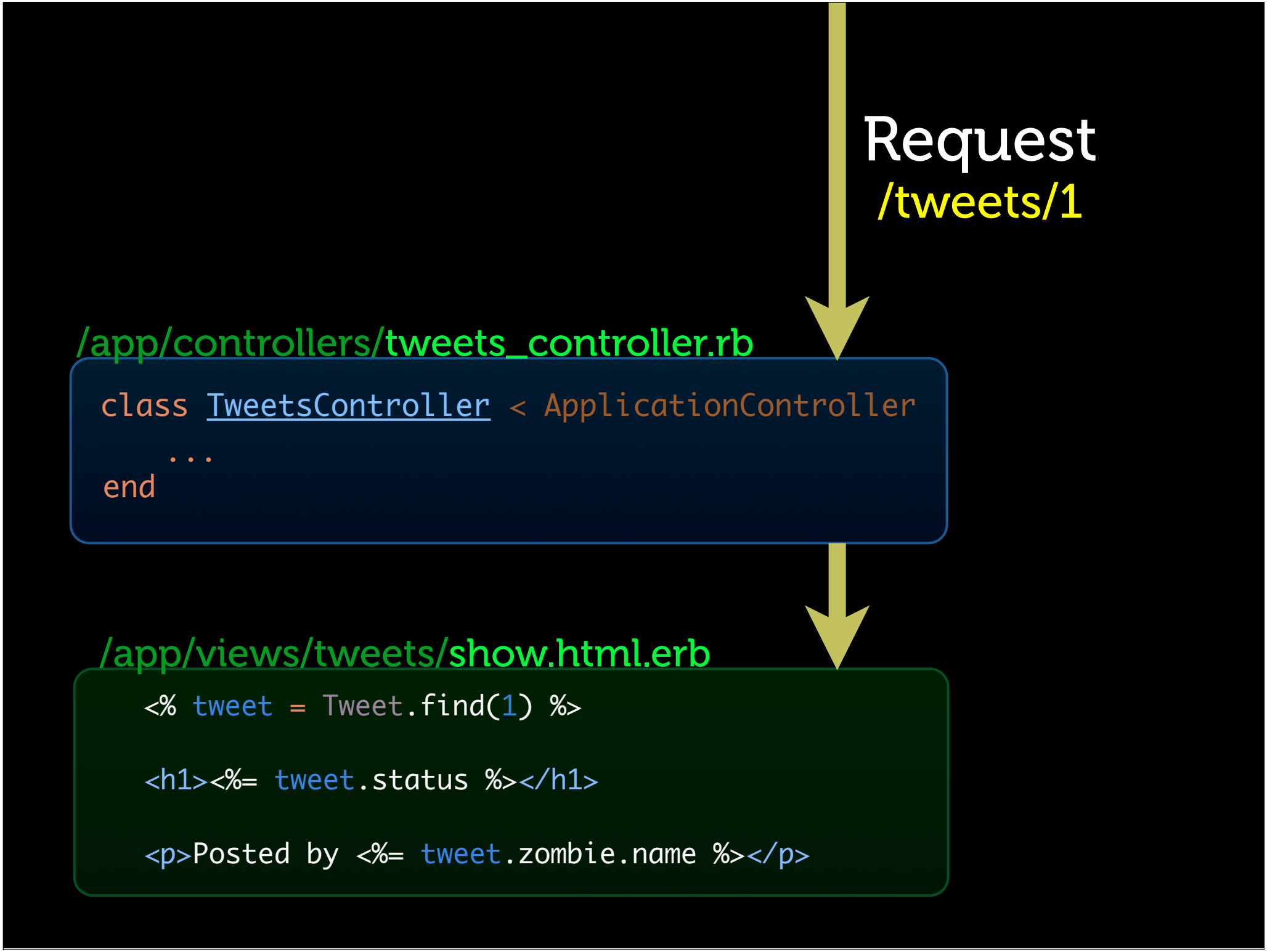
Request
/tweets/1

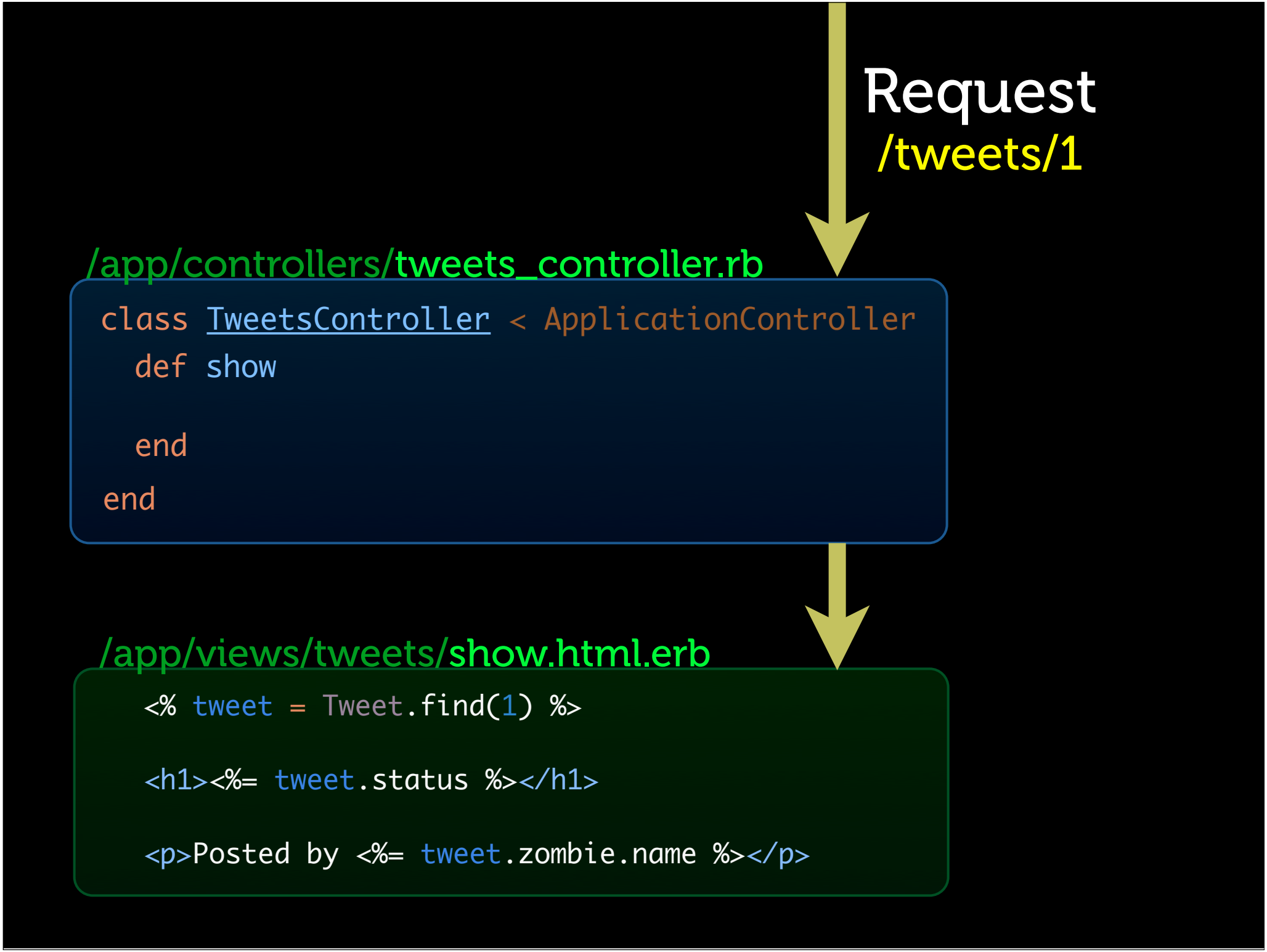
/app/controllers/tweets_controller.rb

Controller

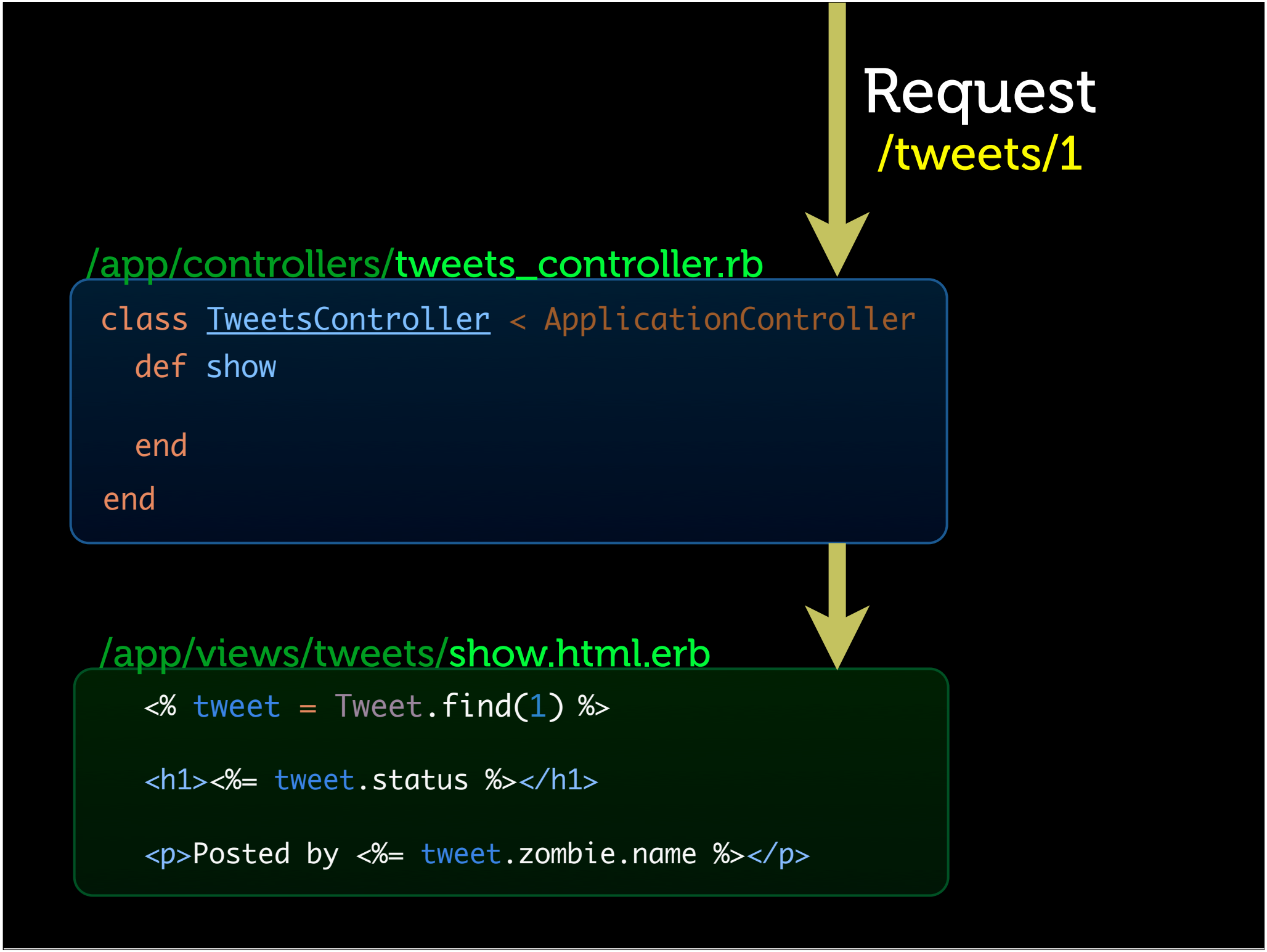
/app/views/tweets/show.html.erb

```
<% tweet = Tweet.find(1) %>

<h1><%= tweet.status %></h1>

<p>Posted by <%= tweet.zombie.name %></p>
```


Request
/tweets/1
/app/controllers/tweets_controller.rb
class TweetsController < ApplicationController
...
end
/app/views/tweets/show.html.erb
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
Posted by <%= tweet.zombie.name %>


Request
/tweets/1
/app/controllers/tweets_controller.rb
class TweetsController < ApplicationController
def show
end
end
/app/views/tweets/show.html.erb
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
Posted by <%= tweet.zombie.name %>


Request
/tweets/1
/app/controllers/tweets_controller.rb
class TweetsController < ApplicationController
def show
end
end
/app/views/tweets/show.html.erb
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
Posted by <%= tweet.zombie.name %>


Request
/tweets/1
/app/controllers/tweets_controller.rb
class TweetsController < ApplicationController
def show
end
end
/app/views/tweets/show.html.erb
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
Posted by <%= tweet.zombie.name %>

Request
/tweets/1

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    tweet = Tweet.find(1)
  end
end
```

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

```
<h1><%= tweet.status %></h1>

<p>Posted by <%= tweet.zombie.name %></p>
```

Instance Variable

@

Request

/tweets/1

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(1)
  end
end
```

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

```
<h1><%= @tweet.status %></h1>

<p>Posted by <%= @tweet.zombie.name %></p>
```

Render

Request
/tweets/1

/app/controllers/tweets_controller.rb

```ruby
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(1)
  end
end
```

/app/views/tweets/status.html.erb

```erb
<h1><%= @tweet.status %></h1>

<p>Posted by <%= @tweet.zombie.name %></p>
```

Render

Request
/tweets/1

/app/controllers/tweets_controller.rb

```ruby
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(1)
    render :action => 'status'
  end
end
```

/app/views/tweets/status.html.erb

```erb
<h1><%= @tweet.status %></h1>

<p>Posted by <%= @tweet.zombie.name %></p>
```

Render

Request

/tweets/1
/tweets/2
/tweets/3
/tweets/4
/tweets/5

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(1)
    render :action => 'status'
  end
end
```

/app/views/tweets/status.html.erb

```
<h1><%= @tweet.status %></h1>

<p>Posted by <%= @tweet.zombie.name %></p>
```

Render

Request

/tweets/1
/tweets/2
/tweets/3
/tweets/4
/tweets/5

params[:id]

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(1)
    render :action => 'status'
  end
end
```

/app/views/tweets/status.html.erb

```
<h1><%= @tweet.status %></h1>

<p>Posted by <%= @tweet.zombie.name %></p>
```

Render

```
params = { :id => "1" }
```

Request

/tweets/1
/tweets/2
/tweets/3
/tweets/4
/tweets/5

params[:id]

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(params[:id])
    render :action => 'status'
  end
end
```

/app/views/tweets/status.html.erb

```
<h1><%= @tweet.status %></h1>

<p>Posted by <%= @tweet.zombie.name %></p>
```

Parameters

Request

```
params = { :status => "I'm dead" }
```

```
@tweet = Tweet.create(:status => params[:status])
```

```
params = {:tweet => { :status => "I'm dead" }}
```

```
@tweet = Tweet.create(:status => params[:tweet][:status])
```

We can also write as

```
@tweet = Tweet.create(params[:tweet])
```

Request

/tweets/1

xml? json?

xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tweet>
  <id type="integer">1</id>
  <status>Where can I get a good bite to eat?</status>
  <zombie-id type="integer">1</zombie-id>
</tweet>
```

json

```
{"tweet":{"id":1,"status":"Where can I get a good bite to eat?","zombie_id":1}}
```

```ruby
class TweetsController < ApplicationController

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

    respond_to do |format|
      format.html # show.html.erb
      format.xml  { render :xml => @tweet }
      format.json { render :json => @tweet }
    end

  end
```

Request

/tweets/1

xml? json?

/tweets/1.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tweet>
  <id type="integer">1</id>
  <status>Where can I get a good bite to eat?</status>
  <zombie-id type="integer">1</zombie-id>
</tweet>
```

/tweets/1.json

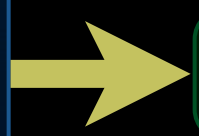
```json
{"tweet":{"id":1,"status":"Where can I get a good bite to eat?","zombie_id":1}}
```

Controller Actions

Request

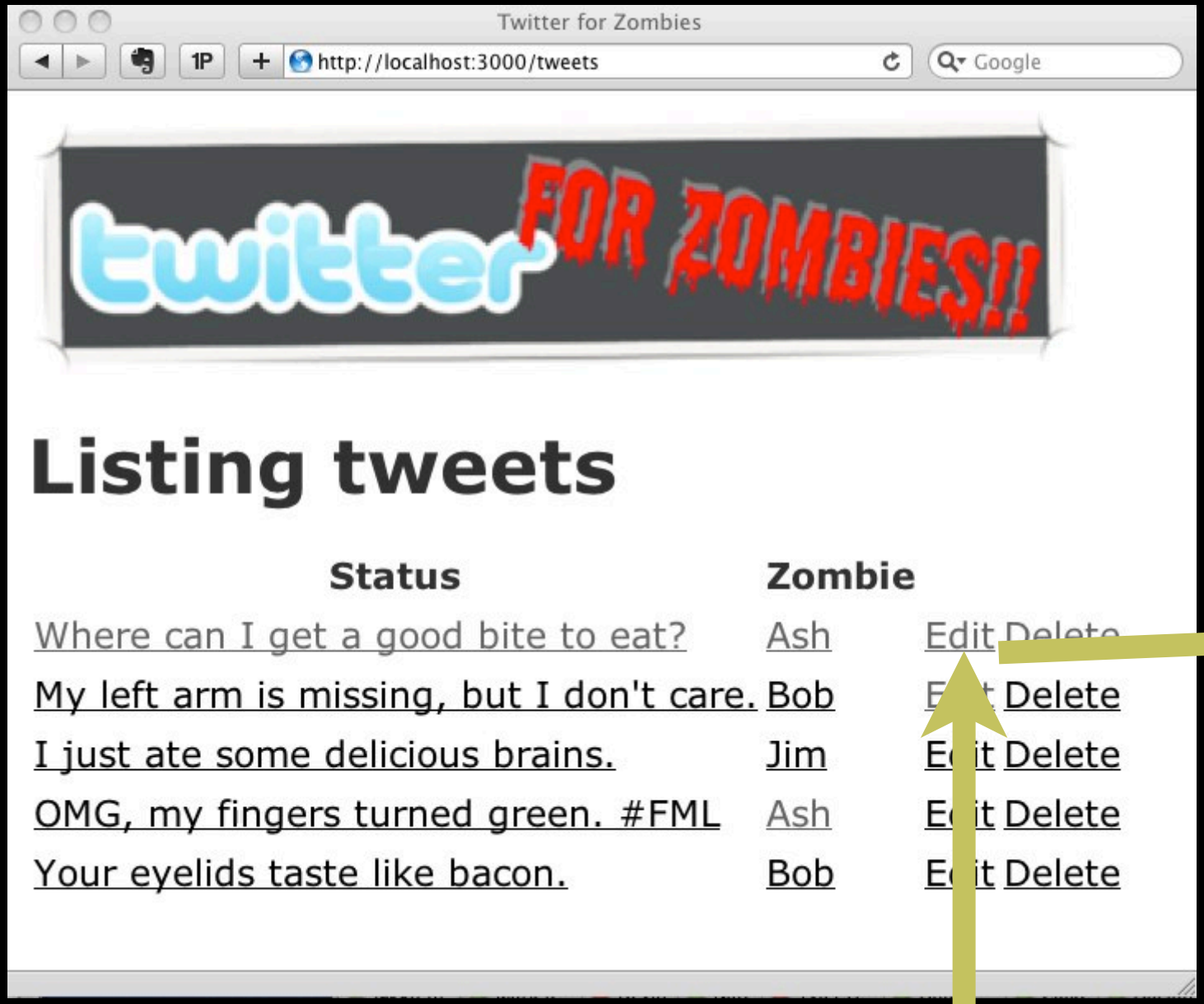
/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def index    List all tweets
  def show     Show a single tweet
  def new      Show a new tweet form
  def edit     Show an edit tweet form
  def create   Create a new tweet
  def update   Update a tweet
  def destroy  Delete a tweet
end
```



View

Adding some authorization

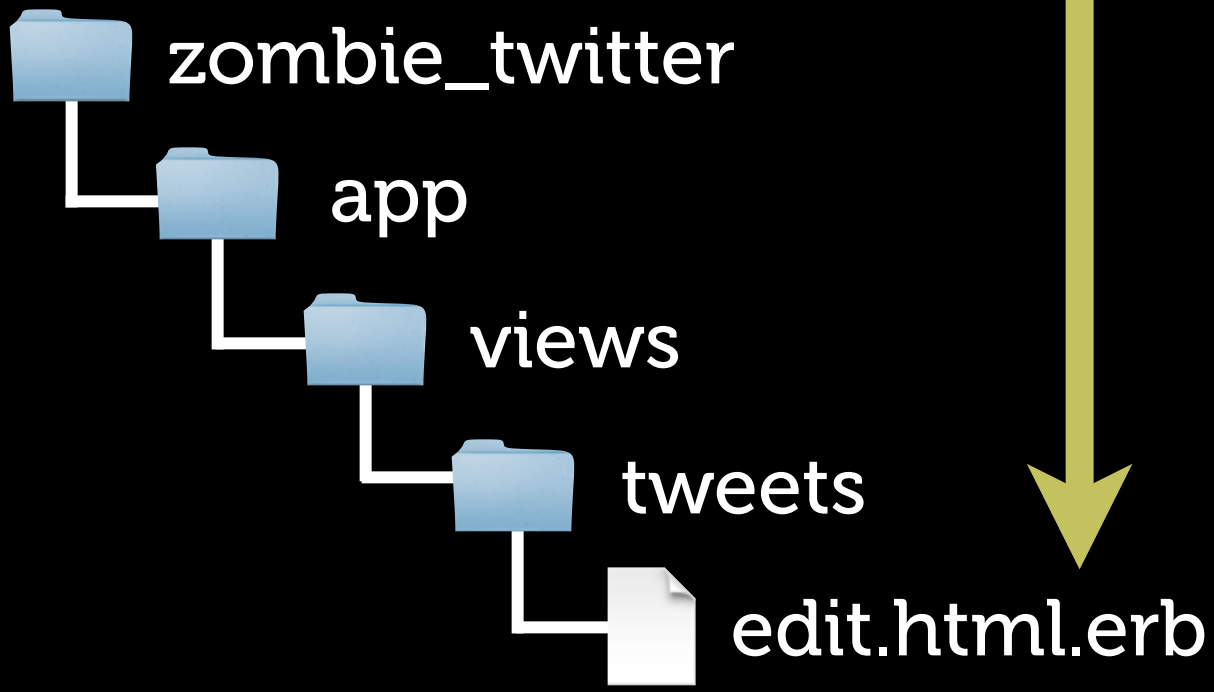



```
def edit
```

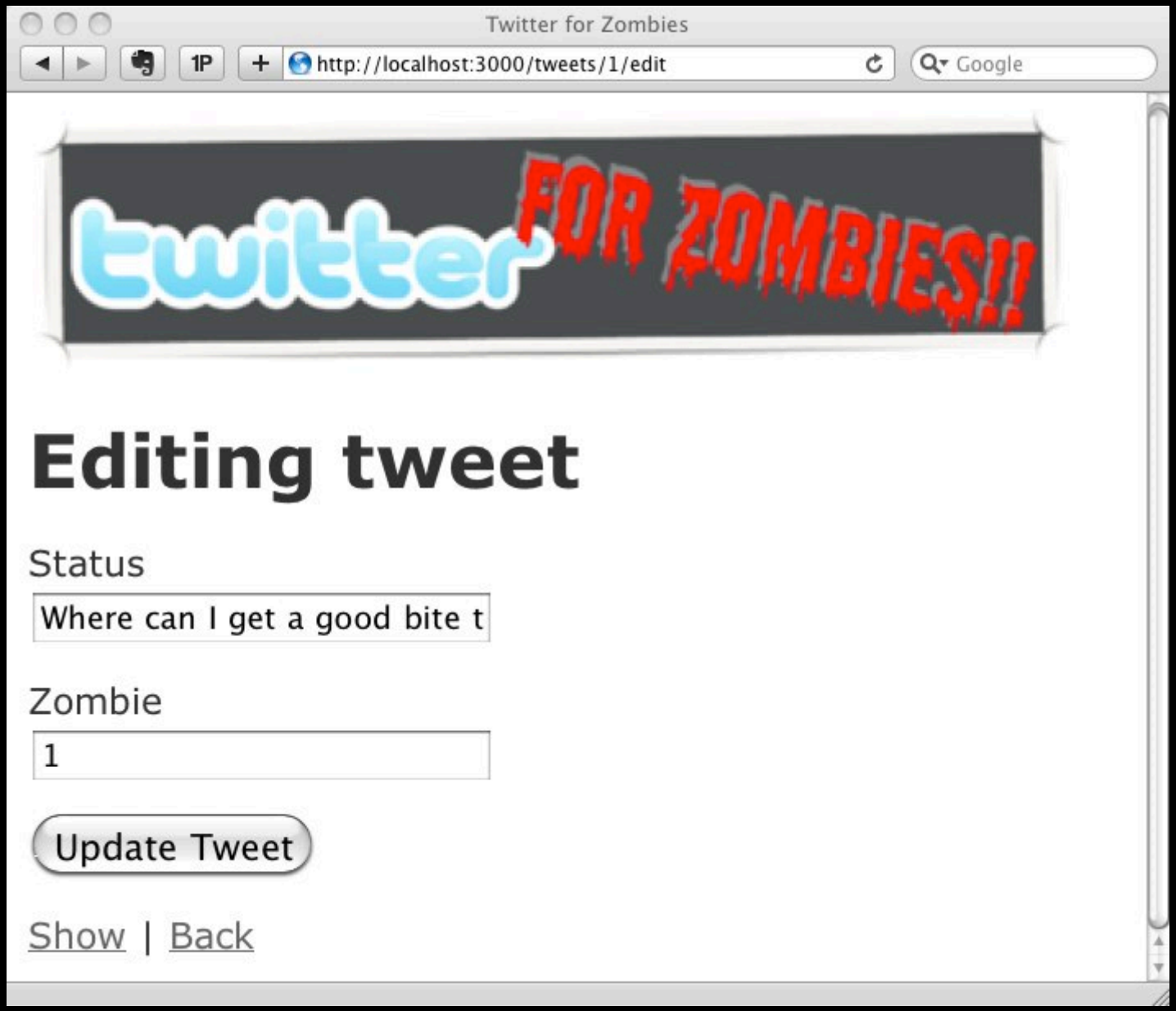
Adding some authorization

/app/controllers/tweets_controller.rb

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

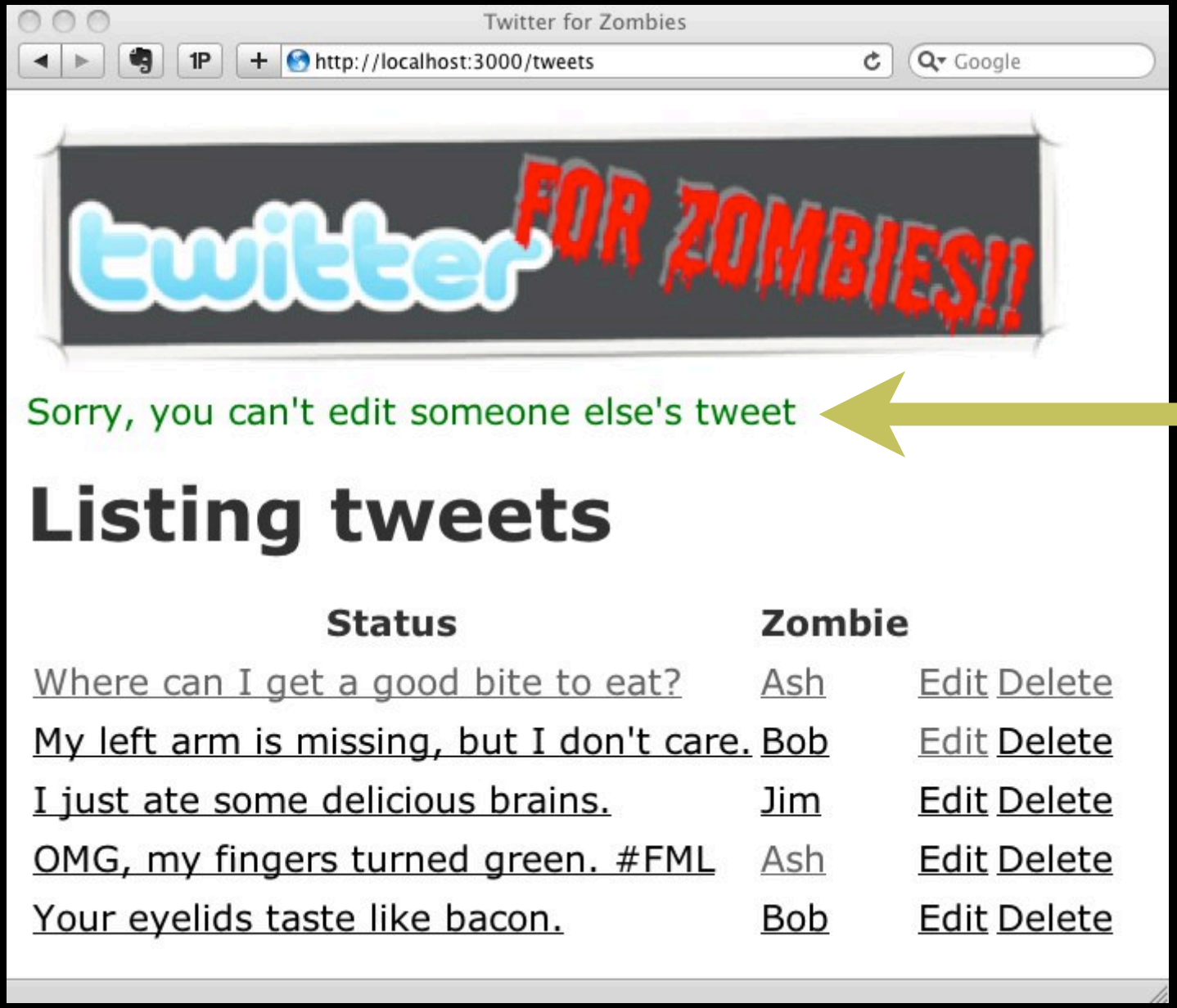


Adding some authorization



Adding some authorization

Adding some authorization



Redirect and Flash

Request

/app/controllers/tweets_controller.rb

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

    if session[:zombie_id] != @tweet.zombie_id
      flash[:notice] = "Sorry, you can't edit this tweet"
      redirect_to(tweets_path)
    end
  end
end
```

session	Works like a per user hash
flash[:notice]	To send messages to the user
redirect <path>	To redirect the request

Redirect and Flash

Request

/app/controllers/tweets_controller.rb

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

    if session[:zombie_id] != @tweet.zombie_id
      redirect_to(tweets_path,
          :notice => "Sorry, you can't edit this tweet")
    end
  end
end
```

Alternate Syntax combining redirect & flash

Notice for Layouts

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Twitter for Zombies</title>
  <%= stylesheet_link_tag :all %>
  <%= javascript_include_tag :defaults %>
  <%= csrf_meta_tag %>
</head>
<body>
  <img src="/images/twitter.png" />

  <%= yield %>

</body></html>
```

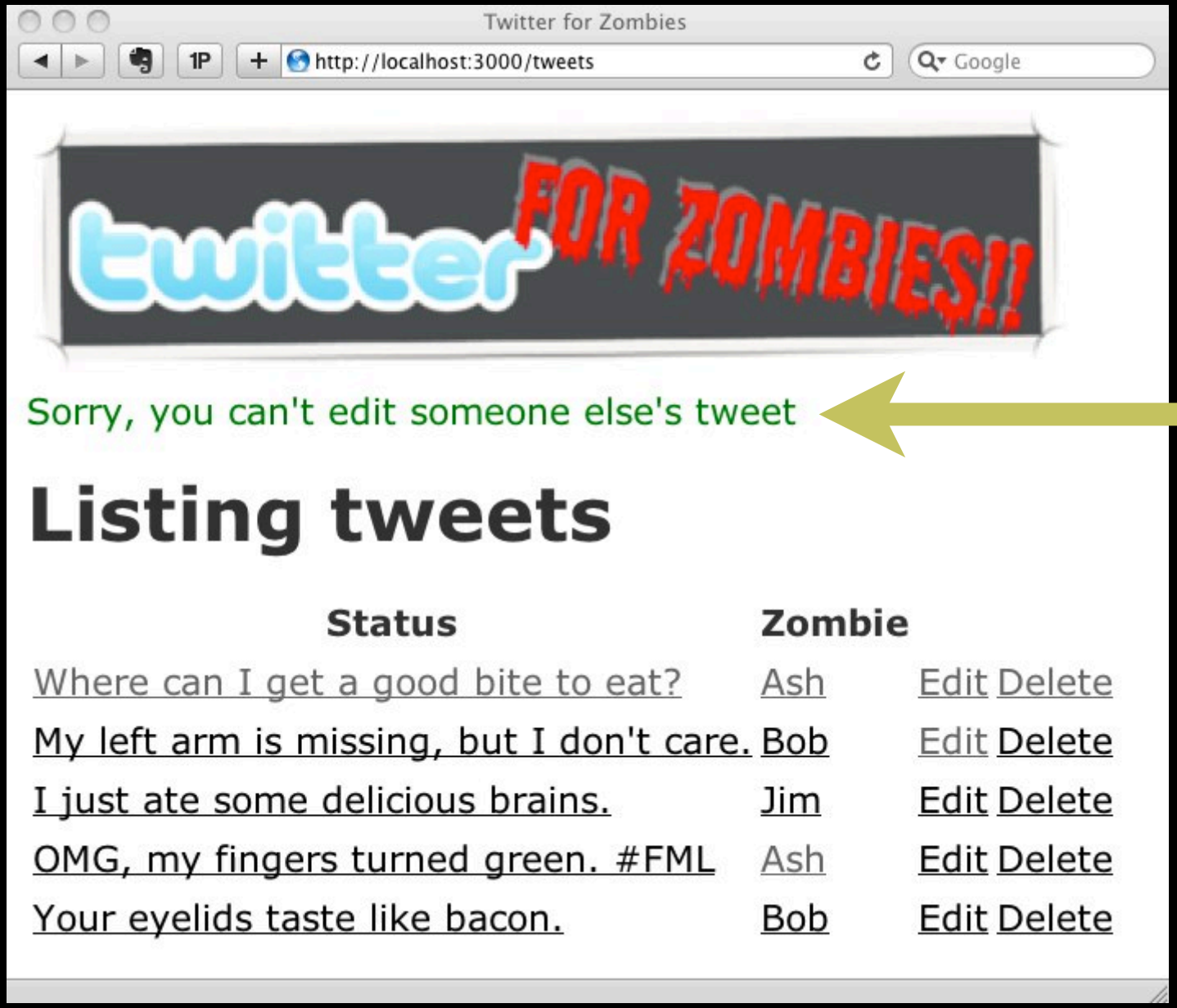
Notice for Layouts

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Twitter for Zombies</title>
  <%= stylesheet_link_tag :all %>
  <%= javascript_include_tag :defaults %>
  <%= csrf_meta_tag %>
</head>
<body>
  <img src="/images/twitter.png" />

  <% if flash[:notice] %>
    <div id="notice"><%= flash[:notice] %></div>
  <% end %>

  <%= yield %>

</body></html>
```

Adding some authorization

Adding some authorization

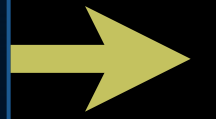


Controller Actions

Request

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def index    List all tweets
  def show     Show a single tweet
  def new      Show a new tweet form
  def edit     Show an edit tweet form
  def create   Create a new tweet
  def update   Update a tweet
  def destroy  Delete a tweet
end
```



View

Before Filters

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController

  def edit
  def update
  def destroy
end
```

Show an edit tweet form

Update a tweet

Delete a tweet

Need Authorization

Before Filters

Request

/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
```

Before Filters

Request

/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
```

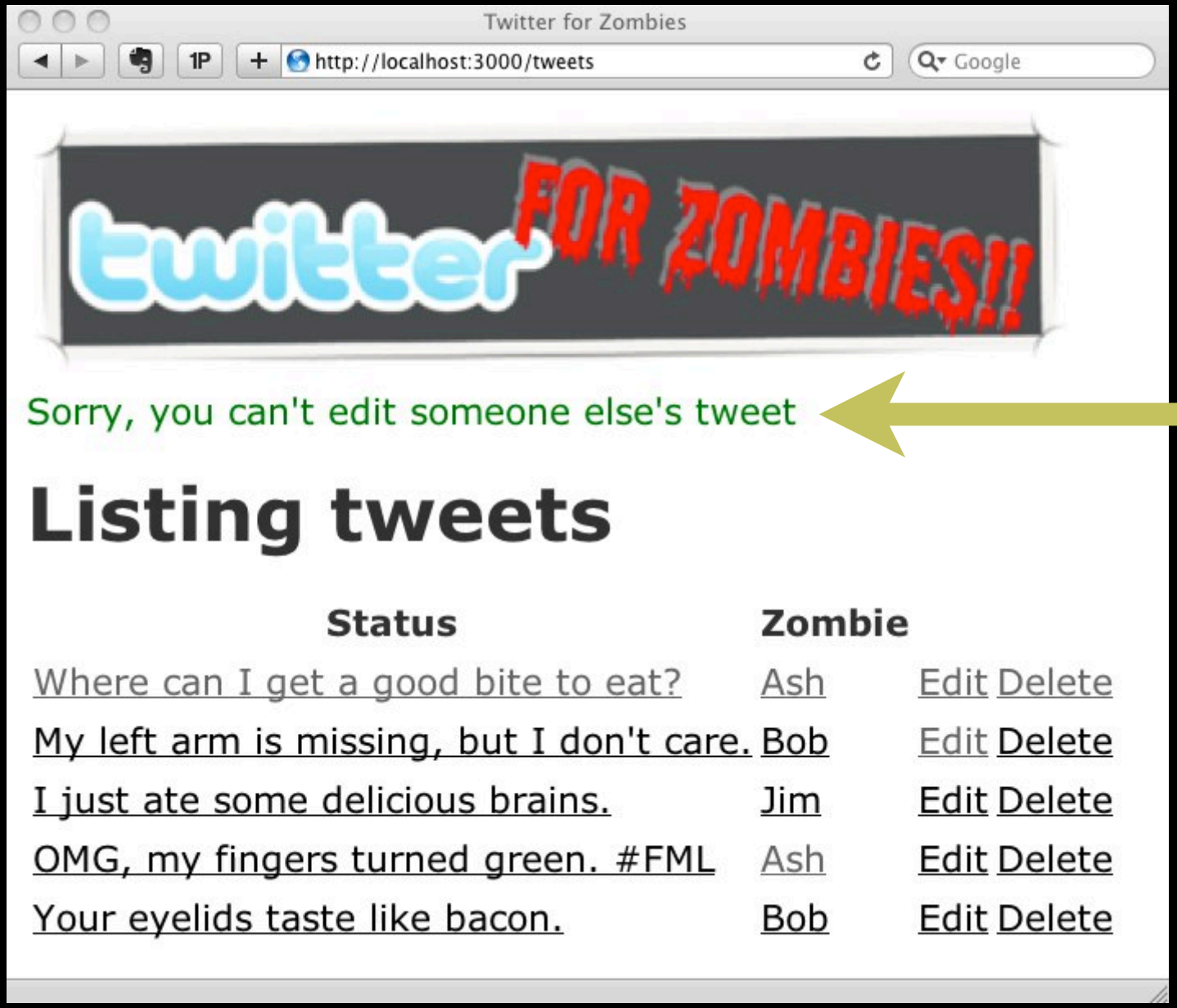
Before Filters

Request

/app/controllers/tweets_controller.rb

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

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

  def check_auth
    if session[:zombie_id] != @tweet.zombie_id
      flash[:notice] = "Sorry, you can't edit this tweet"
      redirect_to tweets_path
    end
  end

  def edit
  def update
  def destroy
end
```

Adding some authorization



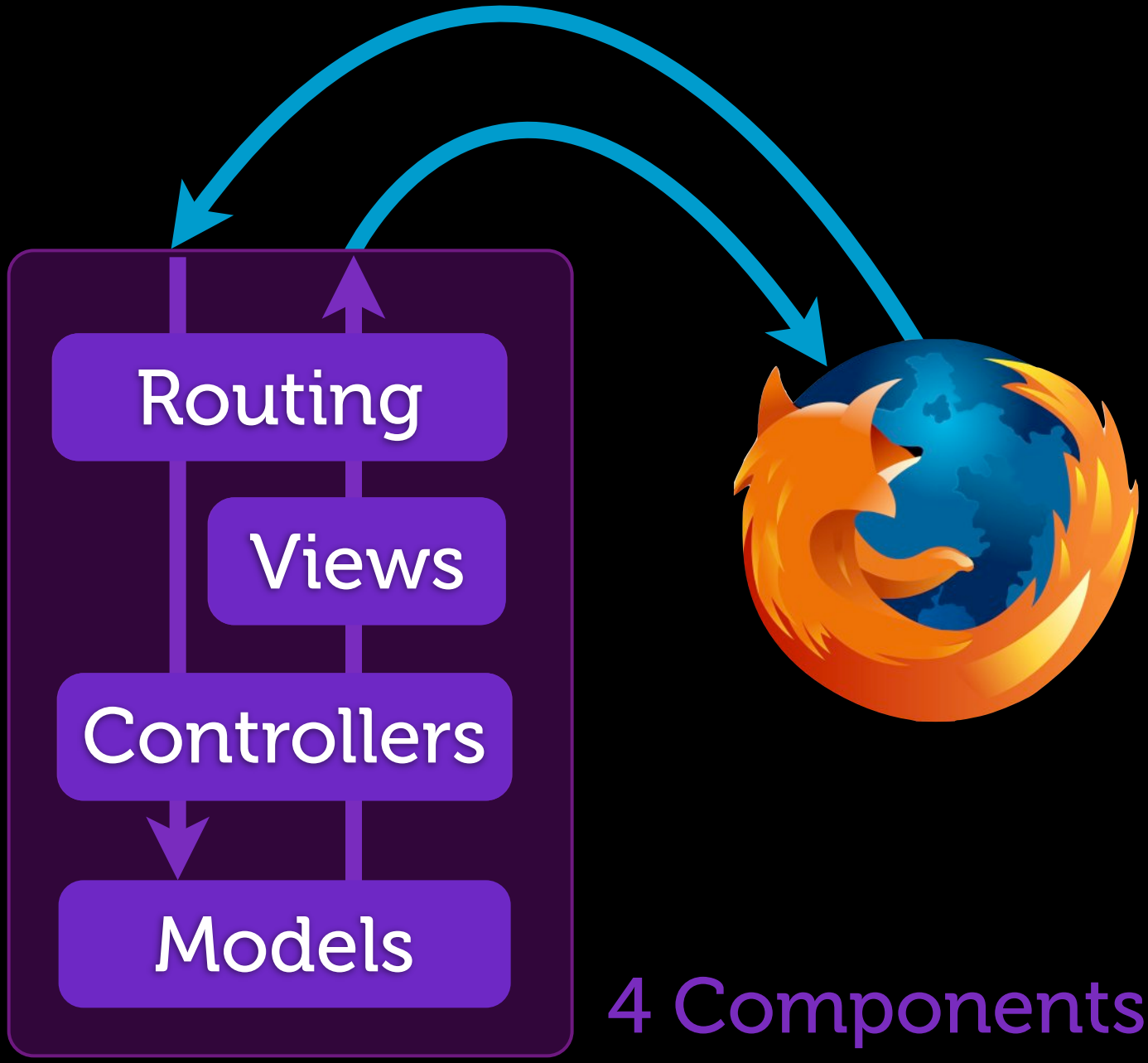
ZOMBIE LAB 4

An Introduction to Rails

with Gregg Pollack

Chapter 5
Routing into darkness

Our Application Stack



4 Components

In order to properly find these paths

```
<%= link_to "<link text>", <code> %>
```

Action	Code	The URL Generated
List all tweets	`tweets_path`	/tweets
New tweet form	`new_tweet_path`	/tweets/new

```
tweet = Tweet.find(1)
```

These paths need a tweet

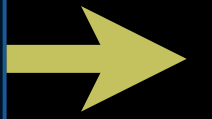
Action	Code	The URL Generated
Show a tweet	`tweet`	/tweets/1
Edit a tweet	`edit_tweet_path(tweet)`	/tweets/1/edit
Delete a tweet	`tweet, :method => :delete`	/tweets/1

and these actions

Request

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def index    List all tweets
  def show     Show a single tweet
  def new      Show a new tweet form
  def edit     Show an edit tweet form
  def create   Create a new tweet
  def update   Update a tweet
  def destroy  Delete a tweet
end
```



View

We need to define routes

Creates what we like to call a "REST"ful resource

```
ZombieTwitter::Application.routes.draw do |map|
  resources :tweets
end
```

Code	The URL Generated	TweetsController action
`tweets_path`	/tweets	`def index`
`tweet`	/tweet/<id>	`def show`
`new_tweet_path`	/tweets/new	`def new`
`edit_tweet_path(tweet)`	/tweets/<id>/edit	`def edit`

and a few more....

Custom Routes

http://localhost:3000/new_tweet —render→ http://localhost:3000/tweets/new

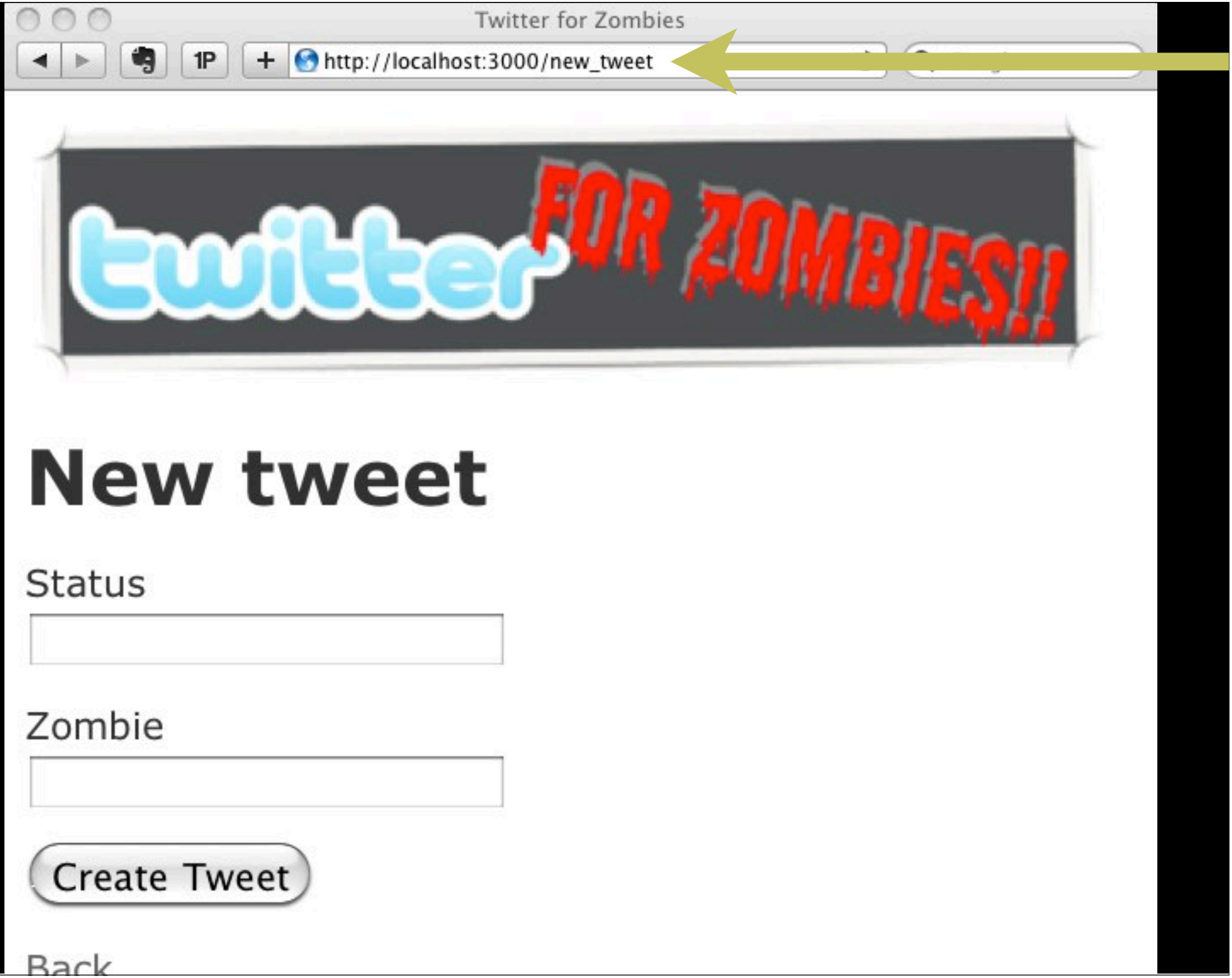
Controller name	Tweets
Action name	new

```
class TweetsController
  def new
    ...
  end
end
```

/config/routes.rb

```
ZombieTwitter::Application.routes.draw do |map|
  resources :tweets
  match 'new_tweet' => "Tweets#new"
end
```

Path — Controller — Action


Twitter for Zombies
http://localhost:3000/new_tweet
twitter
FOR ZOMBIES!!
New tweet
Status
Zombie
Create Tweet
Back

Named Routes

http://localhost:3000/all *render* http://localhost:3000/tweets

Controller name	Tweets
Action name	index

```
class TweetsController
  def index
    ...
  end
end
```

```
match 'all' => "Tweets#index"
```

```
<%= link_to "All Tweets", ? %>
```

tweets_path wouldn't work

Named Routes

http://localhost:3000/all 

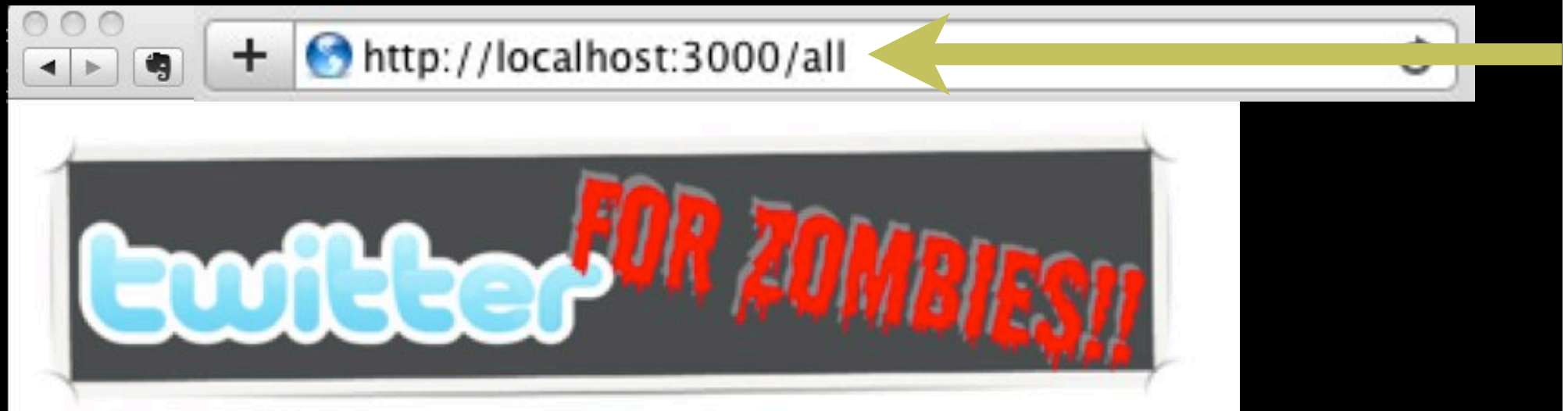
http://localhost:3000/tweets

Controller name	Tweets
Action name	index

```
class TweetsController
  def index
    ...
  end
end
```

```
match 'all' => "Tweets#index", :as => "all_tweets"
```

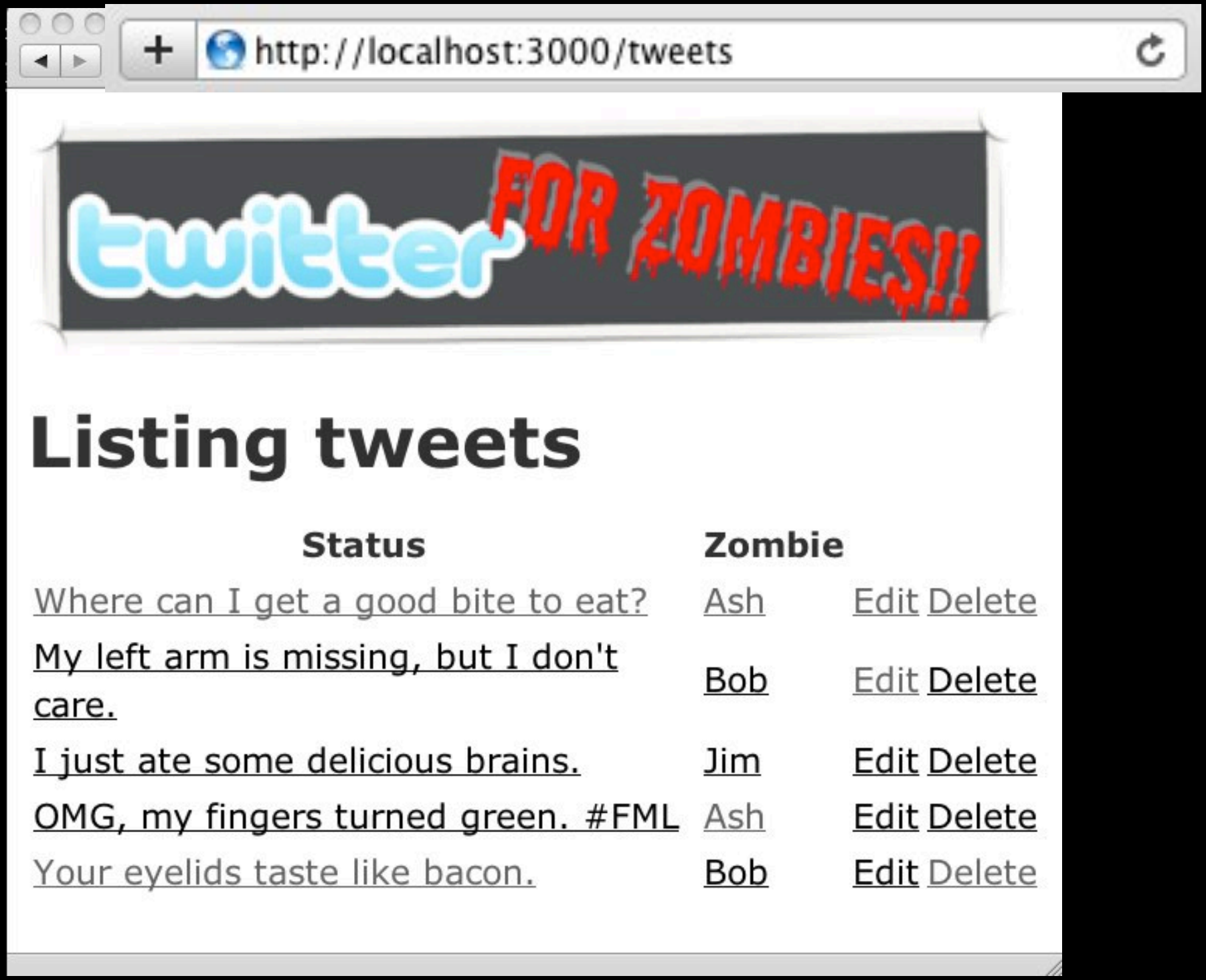
```
<%= link_to "All Tweets", all_tweets_path %>
```



Listing tweets

Status	Zombie	
Where can I get a good bite to eat?	Ash	Edit Delete
My left arm is missing, but I don't care.	Bob	Edit Delete
I just ate some delicious brains.	Jim	Edit Delete
OMG, my fingers turned green. #FML	Ash	Edit Delete
Your eyelids taste like bacon.	Bob	Edit Delete


http://localhost:3000/all
redirect to
http://localhost:3000/tweets


http://localhost:3000/tweets
twitter
FOR ZOMBIES!!
Listing tweets
Status
Zombie
Where can I get a good bite to eat?
Ash
Edit Delete
My left arm is missing, but I don't care.
Bob
Edit Delete
I just ate some delicious brains.
Jim
Edit Delete
OMG, my fingers turned green. #FML
Ash
Edit Delete
Your eyelids taste like bacon.
Bob
Edit Delete

Redirect

http://localhost:3000/all redirect to http://localhost:3000/tweets

```
match 'all' => redirect('/tweets')
```

```
match 'google' => redirect('http://www.google.com/')
```

Root Route

http://localhost:3000/ render http://localhost:3000/tweets

```
root :to => "Tweets#index"
```

Controller Action

```
<%= link_to "All Tweets", root_path %>
```

Route Parameters

/local_tweets/32828

/local_tweets/32801

Find all zombie tweets in this zipcode

/app/controllers/tweets_controller.rb

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

  respond_to do |format|
    format.html # index.html.erb
    format.xml  { render :xml => @tweets }
  end
end
```

Route Parameters

/local_tweets/32828

/local_tweets/32801

Find all zombie tweets in this zipcode

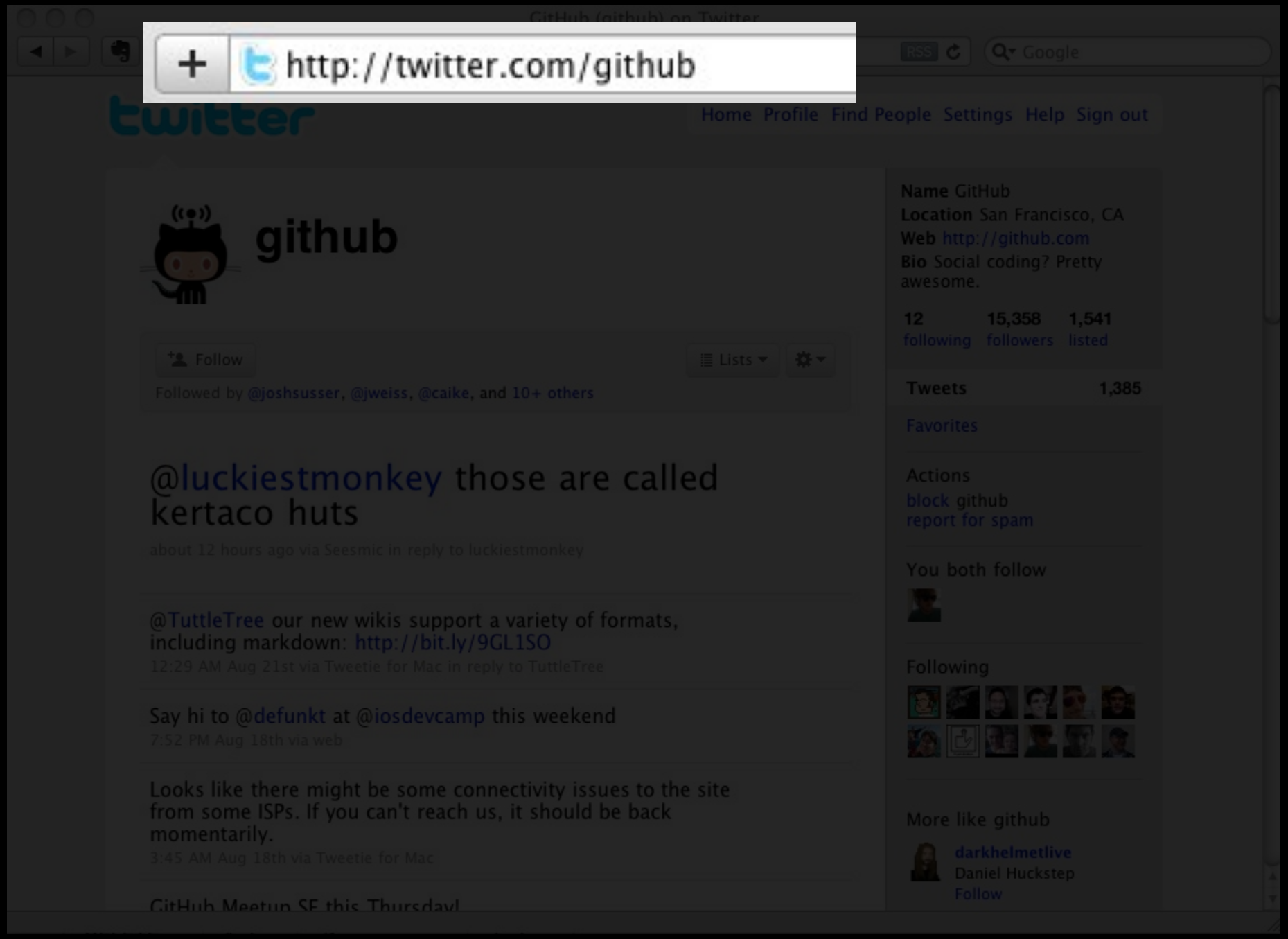
```
match 'local_tweets/:zipcode' => 'Tweets#index'
```

referenced by params[:zipcode] in controller

```
match 'local_tweets/:zipcode' => 'Tweets#index', :as => 'local_tweets'
```

```
<%= link_to "Tweets in 32828", local_tweets_path(32828) %>
```


GitHub (github) on Twitter
http://twitter.com/github
RSS
Google
twitter
Home Profile Find People Settings Help Sign out
github
Follow
Lists
Followed by @joshsusser, @jweiss, @caike, and 10+ others
@luckiestmonkey those are called kertaco huts
about 12 hours ago via Seesmic in reply to luckiestmonkey
@TuttleTree our new wikis support a variety of formats, including markdown: http://bit.ly/9GL1SO
12:29 AM Aug 21st via Tweetie for Mac in reply to TuttleTree
Say hi to @defunkt at @iosdevcamp this weekend
7:52 PM Aug 18th via web
Looks like there might be some connectivity issues to the site from some ISPs. If you can't reach us, it should be back momentarily.
3:45 AM Aug 18th via Tweetie for Mac
GitHub Meetup SF this Thursday!
Name GitHub
Location San Francisco, CA
Web http://github.com
Bio Social coding? Pretty awesome.
12 following
15,358 followers
1,541 listed
Tweets 1,385
Favorites
Actions
block github
report for spam
You both follow
Following
More like github
darkhelmetlive
Daniel Huckstep
Follow


GitHub (github) on Twitter
http://twitter.com/github
Google
twitter
Home Profile Find People Settings Help Sign out
github
Follow
Lists
Followed by @joshsusser, @jweiss, @caike, and 10+ others
@luckiestmonkey those are called kertaco huts
about 12 hours ago via Seesmic in reply to luckiestmonkey
@TuttleTree our new wikis support a variety of formats, including markdown: http://bit.ly/9GL1SO
12:29 AM Aug 21st via Tweetie for Mac in reply to TuttleTree
Say hi to @defunkt at @iosdevcamp this weekend
7:52 PM Aug 18th via web
Looks like there might be some connectivity issues to the site from some ISPs. If you can't reach us, it should be back momentarily.
3:45 AM Aug 18th via Tweetie for Mac
GitHub Meetup SF this Thursday!
Name GitHub
Location San Francisco, CA
Web http://github.com
Bio Social coding? Pretty awesome.
12 following
15,358 followers
1,541 listed
Tweets 1,385
Favorites
Actions
block github
report for spam
You both follow
Following
More like github
darkhelmetlive
Daniel Huckstep
Follow

Route Parameters

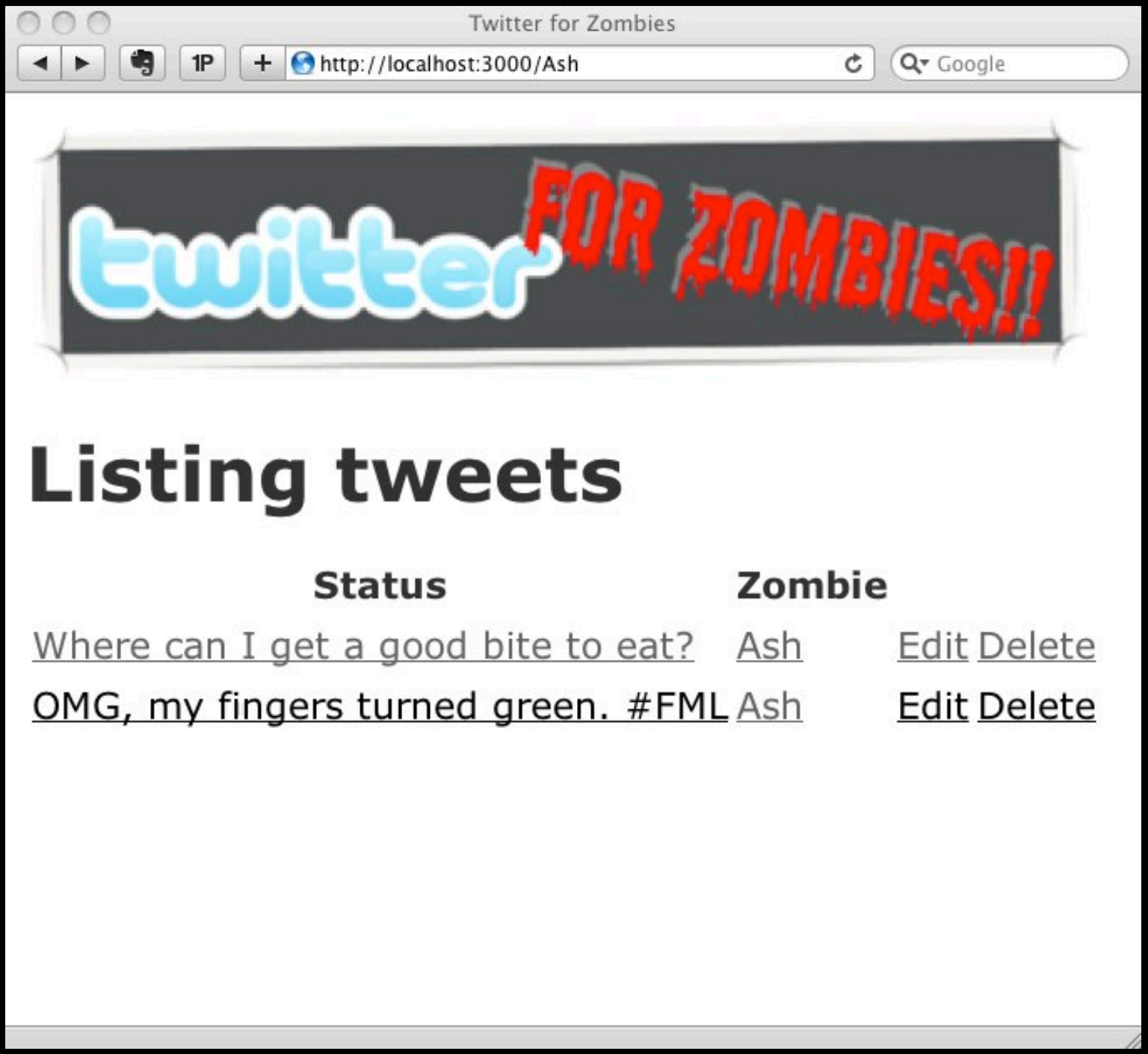
/github
/greggpollack
/eallam
/envylabs


```
match ':name' => 'Tweets#index', :as => 'zombie_tweets'
```

```
    <%= link_to "Gregg", zombie_tweets_path('greggpollack') %>
```

/app/controllers/tweets_controller.rb

```
def index
  if params[:name]
    @zombie = Zombie.where(:name => params[:name]).first
    @tweets = @zombie.tweets
  else
    @tweets = Tweet.all
  end
end
```


Twitter for Zombies
http://localhost:3000/Ash
Google
twitter
FOR ZOMBIES!!
Listing tweets
Status
Zombie
Where can I get a good bite to eat?
Ash
Edit
Delete
OMG, my fingers turned green. #FML
Ash
Edit
Delete

ZOMBIE LAB 5