



DEEP IN THE CRUD

LEVEL 1



Prerequisites:

TRYRUBY.ORG



TWITTER FOR ZOMBIES

A screenshot of a Twitter interface. The top navigation bar includes Home, Connect, Discover, Me, Search, and a blue checkmark icon. On the left, a sidebar lists Tweets, Following, Followers, Favorites, and Lists. Below this is a 'Tweet to Olivier Lacan' section with the handle @olivierlacan. A 'Photos and videos' section shows thumbnail images of Morgan Freeman, a lion, a person in a mask, and a yellow cartoon character. At the bottom, a 'Who to follow' section lists Matt "Wilto" Marquis (@wilto) and Scott Jehl (@scottjehl). Overlaid on the entire screen is a large, high-contrast image of a young girl screaming with her mouth wide open, showing fake fangs and blood. She has blonde hair and is wearing a white t-shirt with red vertical stripes.



DB TABLE

Columns
(we have 3)

tweets

ROWS
(we have 4)

	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

↓
id

↓
status

↓
zombie

Zombie Challenge #1

Retrieve the Tweet object with id = 3



Hash

Series of key value pairs

Single key & value

```
b = { id: 3 }
```

Multiple keys & values

```
b = { id: 3,  
      status: "I just ate some delicious brains",  
      zombie: "Jim" }
```



Hash Recipe: variable = { key: value }

HASH



Hash

Series of key value pairs

HASH

keys	values
:id	3
:status	"I just ate some delicious brains"
:zombie	"Jim"

Symbols

```
b = { id: 3,  
      status: "I just ate some delicious brains",  
      zombie: "Jim" }
```



Hash

Read the value

RETRIEVE

```
b = { id: 3,  
      status: "I just ate some delicious brains",  
      zombie: "Jim" }
```

```
b[:status]
```

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

```
b[:zombie]
```

```
=> "Jim"
```

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

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



read recipe: variable[:key] => value



Zombie Challenge #1

Retrieve the Tweet object with id = 3

RETRIEVE

Result

```
{ id: 3,  
  status: "I just ate some delicious brains",  
  zombie: "Jim" }
```

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 the Tweet object with id = 3

RETRIEVE

Answer

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

Result

```
{ id: 3,  
  status: "I just ate some delicious brains",  
  zombie: "Jim" }
```



CASE & TENSE

Accessing tables

tweets Lowercase & Plural Table Name

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

will allow you to access

Answer

t = Tweet.find(3)

Singular & Uppercase
Table Name



RETRIEVE

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

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

```
puts t[:id]
```

```
=> 3
```

```
puts t[:status]
```

```
=> "I just ate some delicious brains."
```

```
puts t[:zombie]
```

```
=> "Jim"
```



Alternate Syntax

SYNTAX

```
puts t[:id]
```

```
puts t.id
```

```
puts t[:status]
```

```
puts t.status
```

```
puts t[:zombie]
```

```
puts t.zombie
```

```
=> "Jim"
```

HASH VS DOT
SYNTAX



Student Question: Should I use the hash or dot syntax?

You can use EITHER syntax. It comes down to personal preference.



CASE & TENSE

Zombie Challenge #2

Retrieve the Weapon object with id = 1

Answer

```
w = Weapon.find(1)
```

weapons

id	name
1	Ash
2	Bob
3	Jim



CRUD

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 a zombie

CREATE

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

- how delicious -

The id gets set for us



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



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



Recipe

- t = TableName.new
- t.key = value
- t.save
-
-
-
-
-

- t = TableName.new(hash)
- t.save
-
-
-
-
-

- TableName.create(hash)
-
-
-
-

READ

Read

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

=> Returns a single tweet with id of 2

```
Tweet.find(3, 4, 5)
```

=> Returns an array of tweets, id of 3, 4, or 5

```
Tweet.first
```

=> Returns the first tweet

```
Tweet.last
```

=> Returns the last tweet

```
Tweet.all
```

=> Returns all the tweets

READ

Recipes to Read

`Tweet.count`

=> Returns total number of tweets

`Tweet.order(:zombie)`

=> Returns all tweets, ordered by zombies

`Tweet.limit(10)`

=> Returns the first 10 tweets

`Tweet.where(zombie: "ash")`

=> Returns all tweets from zombie named ‘ash’

We can combine any of these read methods together to add constraints

Method Chaining

READ

Method Chaining

```
Tweet.where(zombie: "ash").order(:status).limit(10)
```

=> Returns
only tweets from zombie 'ash'
ordered by status
only the first 10

```
Tweet.where(zombie: "ash").first
```

=> Returns
only tweets from 'ash'
just the first one

Update a zombie

CREATE

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



Recipe

- t = TableName.find(id)
- t.key = value
- t.save
-
-
-
-



Alternate Syntax

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

-
-
- t = TableName.find(id)
- t.attributes = hash
- t.save
-
-
-



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

-
-
- t = Tweet.find(2)
- t = TableName.update(hash)
-
-
-
-

Delete a zombie

DELETE



Recipe

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

```
• t = Table.find(id)  
• t.destroy
```



Alternate Syntax

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

• `TableName.find(id).destroy`



`Tweet.destroy_all`

• TableName.destroy_all

ZOMBIE LAB 1



MODELS
LIFEBLOOD OF THE APP

LEVEL 2



Model

How your Rails application communicates with a data store

MODELS

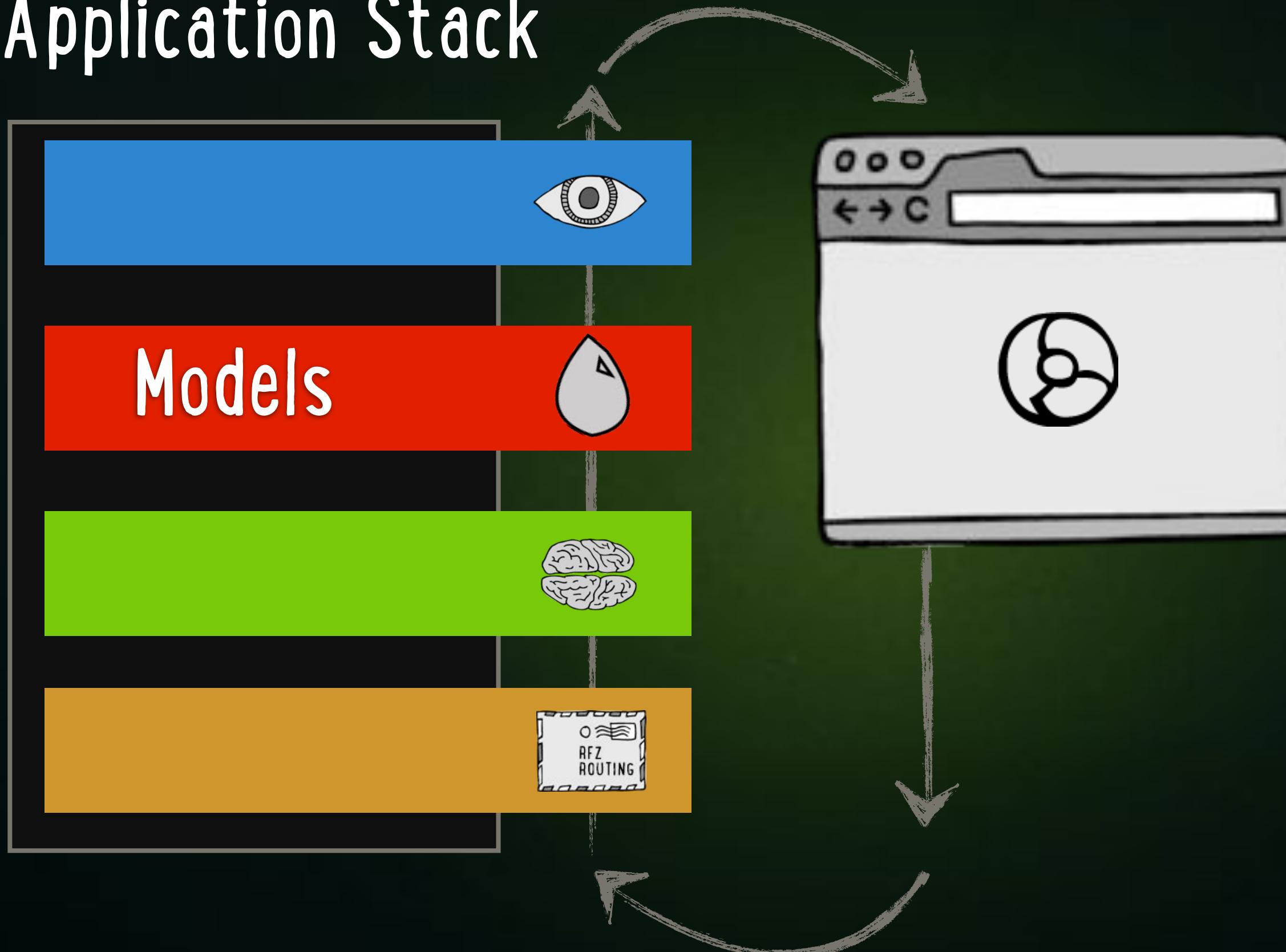


The Lifeblood of the application



Application Stack

ROUTING



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	Ash
2	My left arm is missing, but I	Bob
3	I just ate some delicious	Jim
4	OMG, my fingers turned green.	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	Ash
2	My left arm is missing, but I	Bob
3	I just ate some delicious	Jim
4	OMG, my fingers turned green.	Ash



app/models/tweet.rb

```
class Tweet < ActiveRecord::Base  
end
```

MODELS

(Instance of Tweet #3)

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

Class

tweets

id	status	zombie
1	Where can I get a good bite to	Ash
2	My left arm is missing, but I	Bob
3	I just ate some delicious	Jim
4	OMG, my fingers turned green.	Ash



Validations

VALIDATIONS

tweets

id	status	zombie
1	Where can I get a good bite to	Ash
2	My left arm is missing, but I	Bob
3	I just ate some delicious	Jim
4	OMG, my fingers turned green.	Ash
5		



Ack, we don't want to create a blank Tweet!

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



app/models/tweet.rb

```
class Tweet < ActiveRecord::Base
  validates_presence_of :status
end
```

VALIDATIONS

```
>> t = Tweet.new
=> #<Tweet id: nil, status: nil, zombie: nil>
>> t.save
=> false

>> t.errors.messages
=> {status:["can't be blank"]}

>> t.errors[:status][0]
=> "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

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

Validation

Alternate Syntax

```
validates :status,  
presence: true,  
length: { minimum: 3 }  
  
presence: true  
uniqueness: true  
numericality: true  
length: { minimum: 0, maximum: 2000 }  
format: { with: /.*/ }  
acceptance: true  
confirmation: true
```

Additional Options



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	Bob
3	I just ate some delicious brains.	Jim
4	OMG, my fingers turned green. #FML	Ash

👎 We want to store zombies in their own table!



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	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1 ←



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	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



zombies

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

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	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1

RELATIONSHIPS

a Zombie
HAS MANY
Tweets



zombie_id

1

2

3

1

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	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1

a Tweet
BELONGS TO
a Zombie

app/models/tweet.rb

```
class Tweet < ActiveRecord::Base
  belongs_to :zombie
end
```

Singular

zombies

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

Using Relationships

RELATIONSHIPS

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

> t = Tweet.create(status: "Your eyelids taste like bacon.",
                     zombie: ash)
=> #<Tweet id: 5, status: "Your eyelids taste like bacon.", zombie_id: 1>

> ash.tweets.count
=> 3

> 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>,
      #<Tweet id: 5,
      status: "Your eyelids taste like bacon.", zombie_id: 1>]
```

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	2
3	I just ate some delicious brains.	3
4	OMG, my fingers turned green. #FML	1
5	Your eyelids taste like bacon.	1



zombies

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



Using Relationships

RELATIONSHIPS

```
> t = Tweet.find(5)
=> #<Tweet id: 5, status: "Your eyelids taste like bacon.",  
zombie_id: 1>

> t.zombie
=> #<Zombie id: 1,  
      name: "Ash",  
      graveyard: "Glen Haven Memorial Cemetery">

> t.zombie.name
=> "Ash"
```

ZOMBIE LAB 2



VIEW

VISUAL REPRESENTATION

LEVEL 3



View

User Interface. What we see.

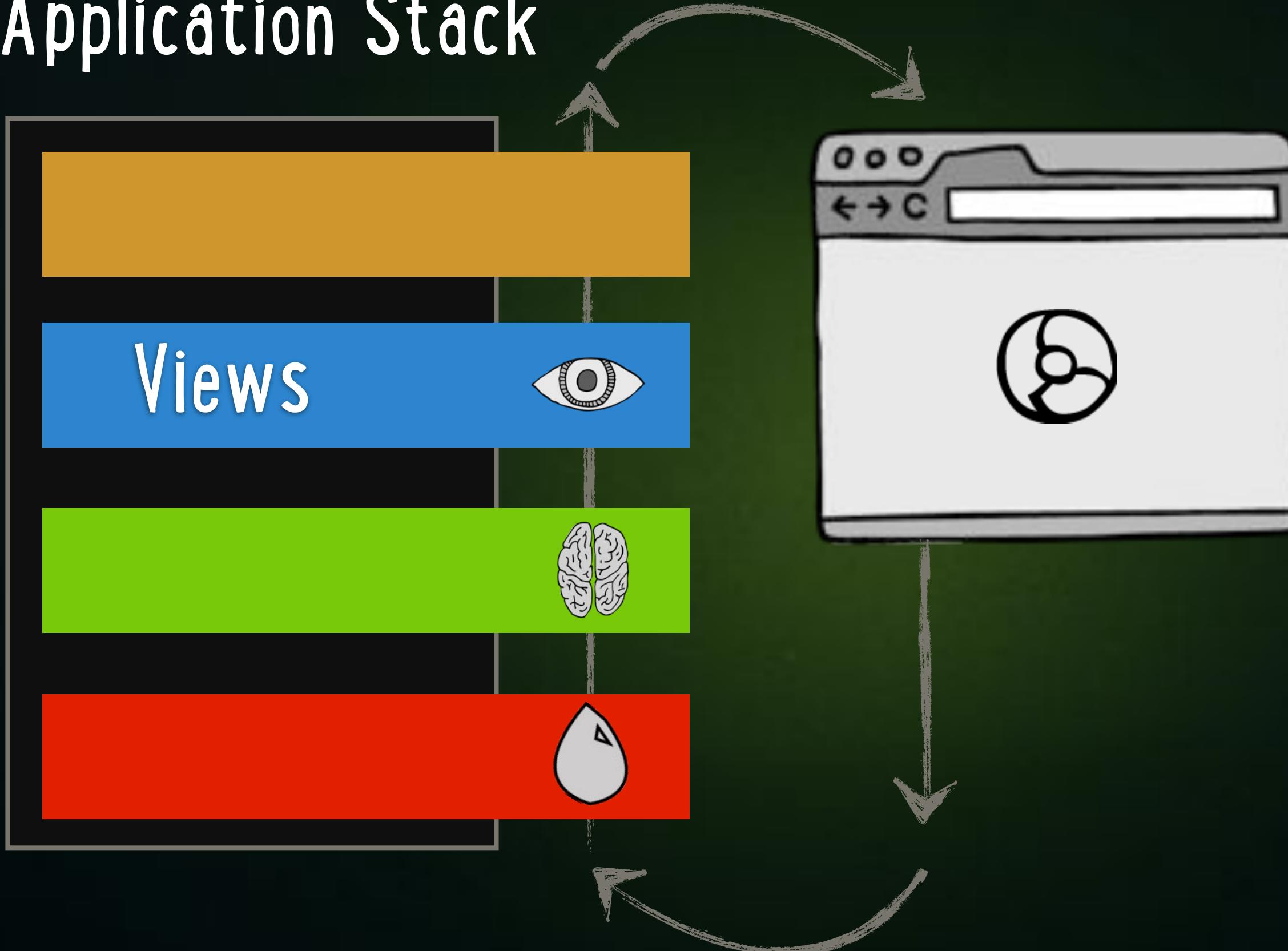
VIEWS



The Visual Representation
of the application



Application Stack



VIEWS



ERB



Edible Rotting Bodies

Ruby inside HTML

Embedded Ruby

List all tweets

/ index.html.erb

show.html.erb

View a tweet



Show a tweet

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

```
<!DOCTYPE html>
<html>
  <head><title>Twitter for Zombies</title>
  <body>
    <header>...</header>

    <% tweet = Tweet.find(1) %>
    <h1><%= tweet.status %></h1>
    <p>Posted by <%= tweet.zombie.name %><
  </body>
</html>
```



Evaluate Ruby: <% . . . %> Evaluate Ruby & Print Result: <%=

SHOW

Show a tweet

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

```
<!DOCTYPE html>
<html>
  <head><title>Twitter for Zombies</title></head>
  <body>
    <header>...</header>

    <% tweet = Tweet.find(1) %>
    <h1><%= tweet.status %></h1>
    <p>Posted by <%= tweet.zombie.name %></p>
  </body>
</html>
```



- Rotten Code -
We are repeating this in
multiple views.



Evaluate Ruby: <% . . . %> Evaluate Ruby & Print Result: <%= . . . %>



Show a tweet

SHOW

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

```
<!DOCTYPE html>
<html>
  <head><title>Twitter for Zombies</title></head>
  <body>
    <header>...</header>
    <%= yield %>
  </body>
</html>
```

- Tasty Code -
Every page we create
uses this template by default

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

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= tweet.zombie.name %></p>
```



Show a tweet

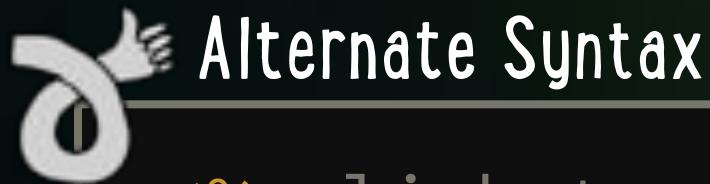
/app/views/tweets/show.html.erb

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= tweet.zombie.name %></p>
```

How do we
make this a link ??

Create a Link

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



Alternate Syntax

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

As with tweets,
shorter is better.



Link Recipe

```
<%= link_to text_to_show, object_to_show %>
```

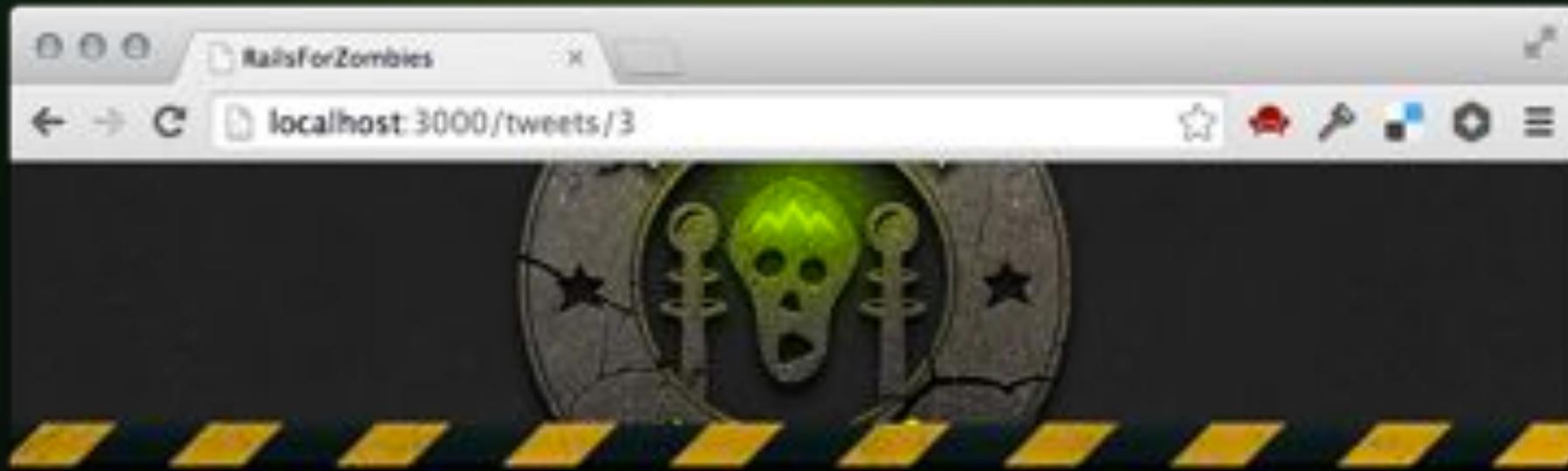
SHOW



Show a tweet

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

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= link_to tweet.zombie.name, tweet.zombie %> </p>
```



Where can I get a good bite to eat?

Posted by: Ash



Show a tweet

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

SHOW

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= link_to tweet.zombie.name, tweet.zombie %> </p>
```

Student Question:

What options
can we use
with link_to?

Well Billy, we are glad you asked!
Our next topic happens to be:

Looking up Documentation



Options for link_to

1. Look in the source

Open your editor and search for “def link_to”

Command Line

```
git clone http://github.com/rails/rails.git  
cd rails  
grep -rin "def link_to"
```



LINK

Options for link_to

1. Look in the source

Open your editor and search for "def link_to"

2. Look at api.rubyonrails.org

(and search for link_to)



Ruby on Rails API

← → C api.rubyonrails.org

Q link_to

link_to (name = nil, options = {}, html_options = {})
ActionView::Helpers::UrlHelper
<code>Creates a link tag of the given <code>name</code> with

link_to function(name, function, html_options = {})
ActionView::Helpers::JavaScriptHelper
<code>Returns a link whose <code>:onclick</code> handler is

link_to_if(condition, name, options = {}, html_options = {})
ActionView::Helpers::UrlHelper
<code>Creates a link tag of the given <code>name</code> if

link_to_unless(condition, name, options = {}, html_options = {})
ActionView::Helpers::UrlHelper
<code>Creates a link tag of the given <code>name</code> unless

link_to_unless_current(name, options = {}, html_options = {})
ActionView::Helpers::UrlHelper
<code>Creates a link tag of the given <code>name</code> unless

Options

- `:data` - This option can be used to add custom data attributes.
- `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 record (which search bots can follow while spidering your site).

Data attributes

- `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(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 :method, is passed to url_  
  
link_to(options = {}, html_options = {}) do  
  # options  
end
```



Options for link_to

LINK

1. Look in the source

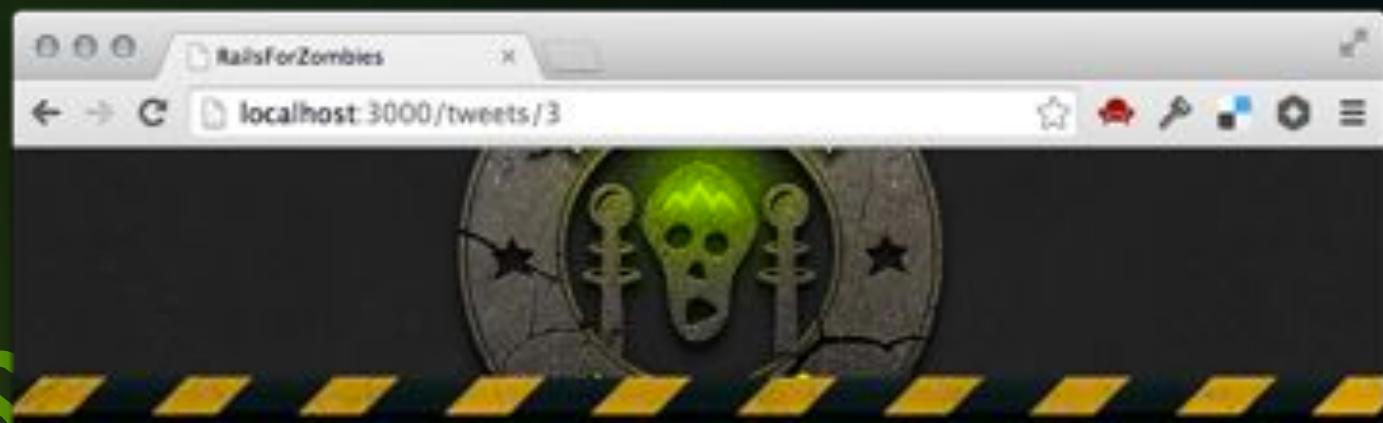
Open your editor and search for "def link_to"

2. Look at api.rubyonrails

(and search for link_to)

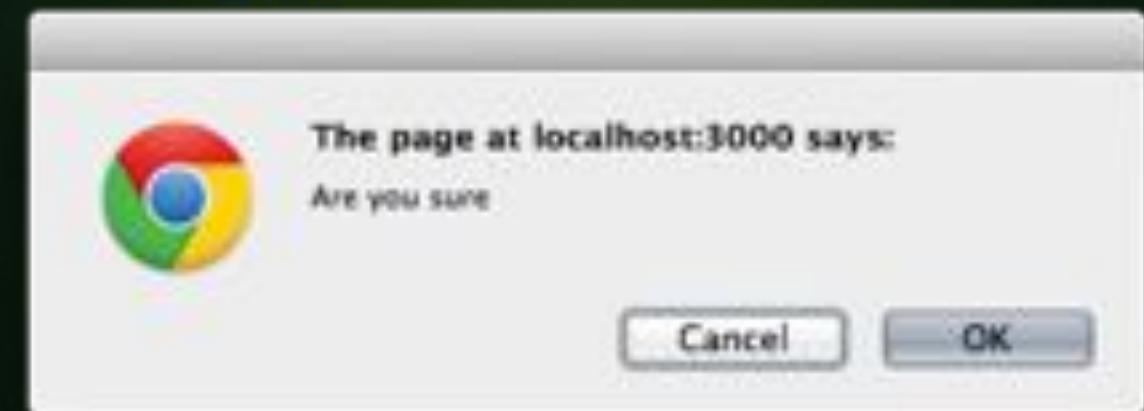
/app/views/tweets/show.html.erb

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

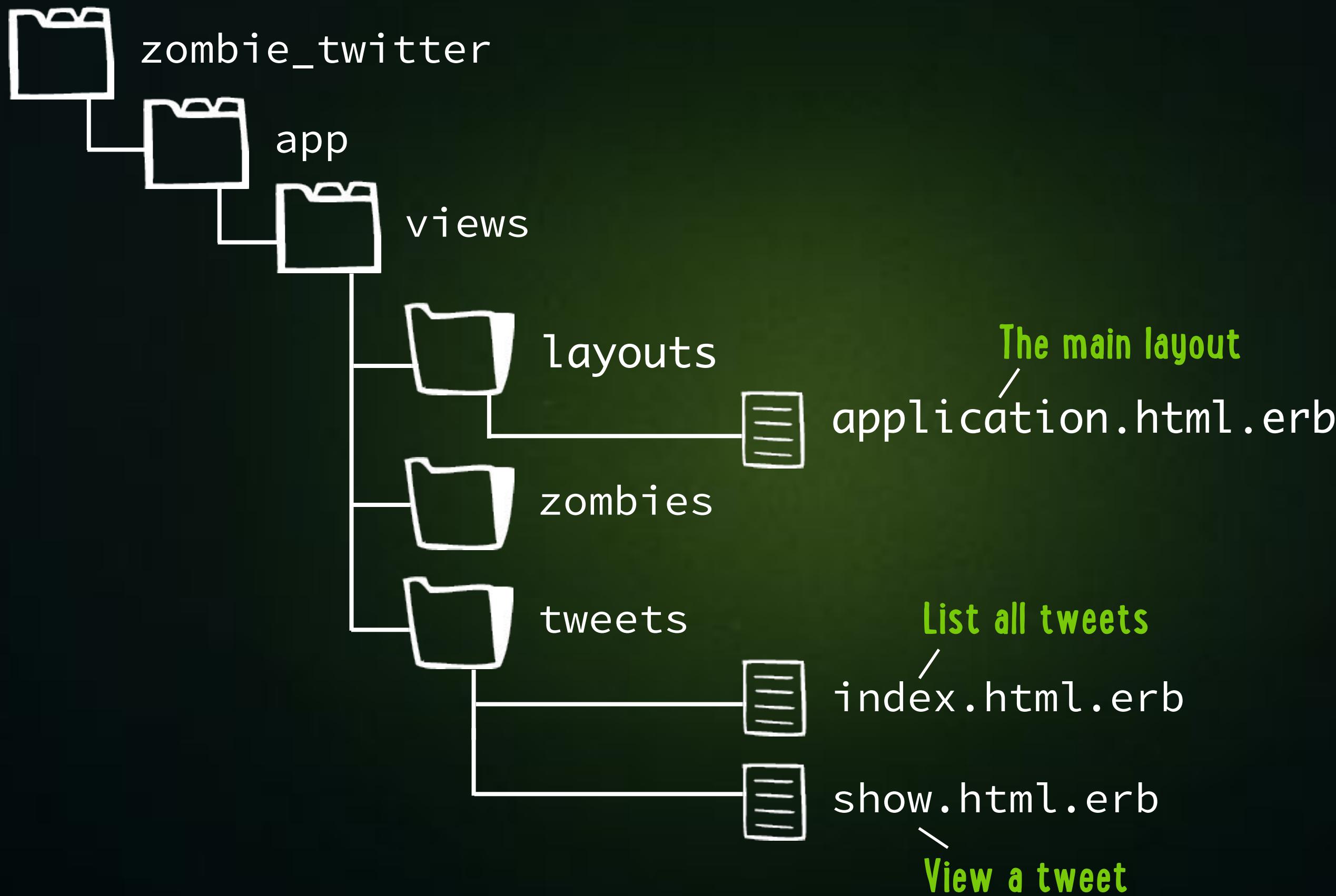


Where can I get a good bite to eat?

Posted by: Ash



VIEWS



List Tweets

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

```
<h1>Listing tweets</h1>
<table>
  <tr>
    <th>Status</th>
    <th>Zombie</th>
  </tr>
<% Loop through each tweet %>
  <tr>
    <td><%= tweet.status %></td>
    <td><%= tweet.zombie.name %></td>
  </tr>
<% end %>
</table>
```

List Tweets

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

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


| Status              | Zombie                   |
|---------------------|--------------------------|
| <%= tweet.status %> | <%= tweet.zombie.name %> |


```

What they return

Tweet

class

Tweet.all

array of tweets

tweet

single tweet



Listing tweets

Status

Twitter is for the living... for now. Jim

Such Hunger. Much Brains.

Zombie

Bob

This code tastes like rotted brains. Ash

LINK

Create Links

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

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

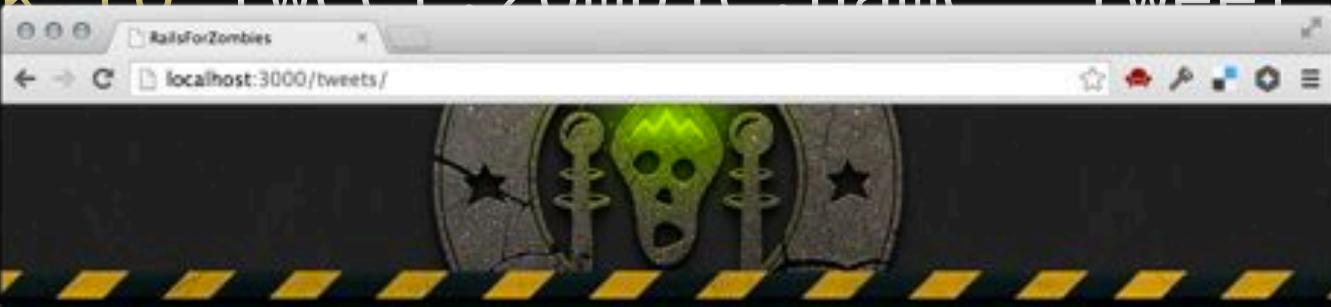


LINK

Create Links

/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 %>
```



Listing tweets

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



Empty Table?

VIEWS

A screenshot of a web browser window titled "RailsForZombies". The URL bar shows "localhost:3000/tweets". The page content is a table with the following structure:

Status	Zombie
No Tweets Found	

The word "Status" is preceded by a thumbs-down icon, and "Zombie" is preceded by a thumbs-up icon. The entire table row is highlighted with a green arrow pointing from the code block below.



Listing tweets

Status	Zombie
No Tweets Found	

Listing tweets
Status Zombie

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



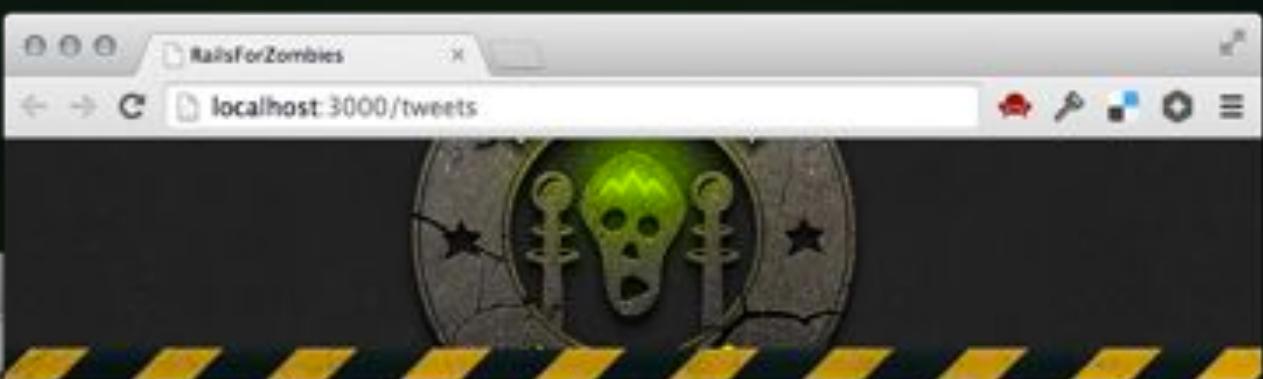
Empty Table?

10

Listing tweets

Status Zombie

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



Listing tweets

Status Zombie

No Tweets Found



Empty Table?

```
<% tweets = Tweet.all %>  
  
<% tweets.each do |tweet| %>  
  ...  
<% end %>  
  
<% if tweets.size == 0 %>  
  <em>No tweets found</em>  
<% end %>
```

Listing tweets

Status Zombie



Listing tweets

Status Zombie

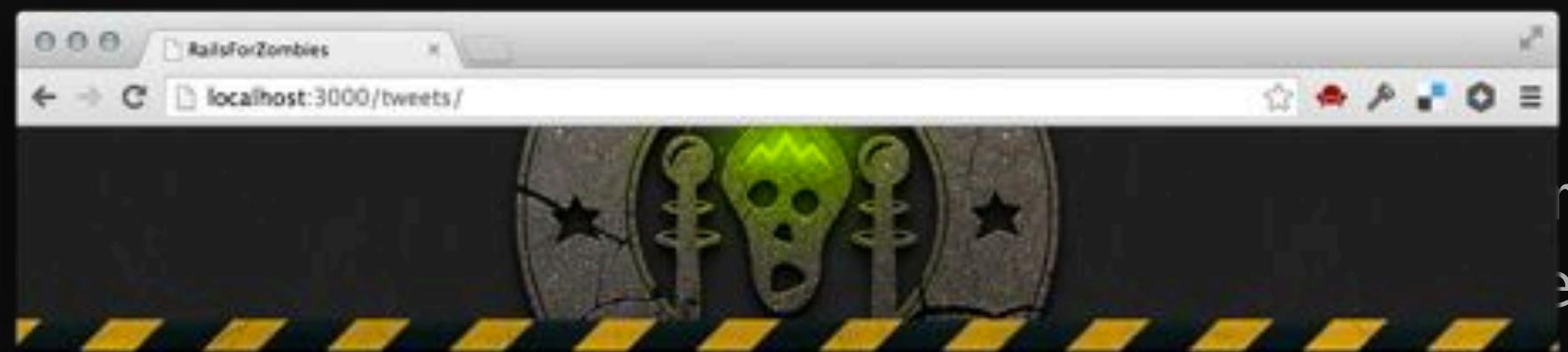
No Tweets Found

VIEWS

VIEWS

Edit & Delete Links

```
<% tweets.each do |tweet| %>
<tr>
<td><%=
```



Listing tweets

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



All Links For Tweets

VIEWS

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

`tweet = Tweet.find(1)` 

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



Link Recipe: `<%= link_to text_to_show, code %>`

ZOMBIE LAB 3



CONTROLLERS
BRAINS OF THE APP

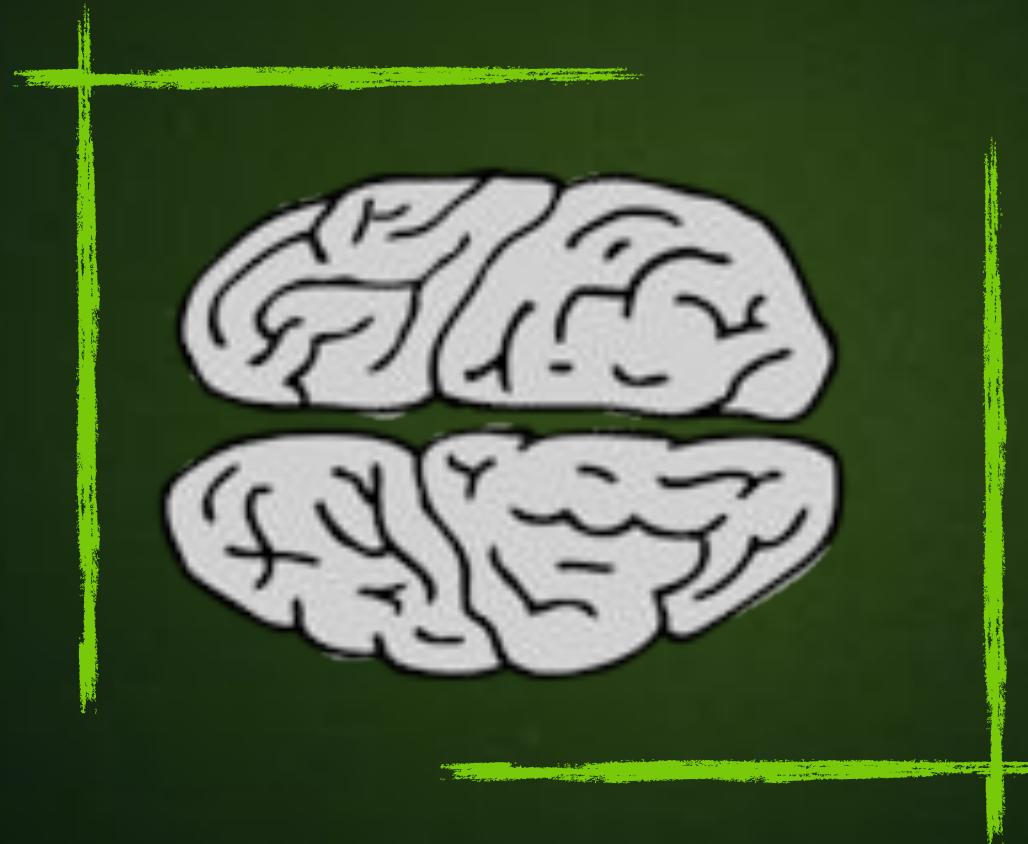
LEVEL 4



Controller

The binding between the Model and the View

CONTROLLERS

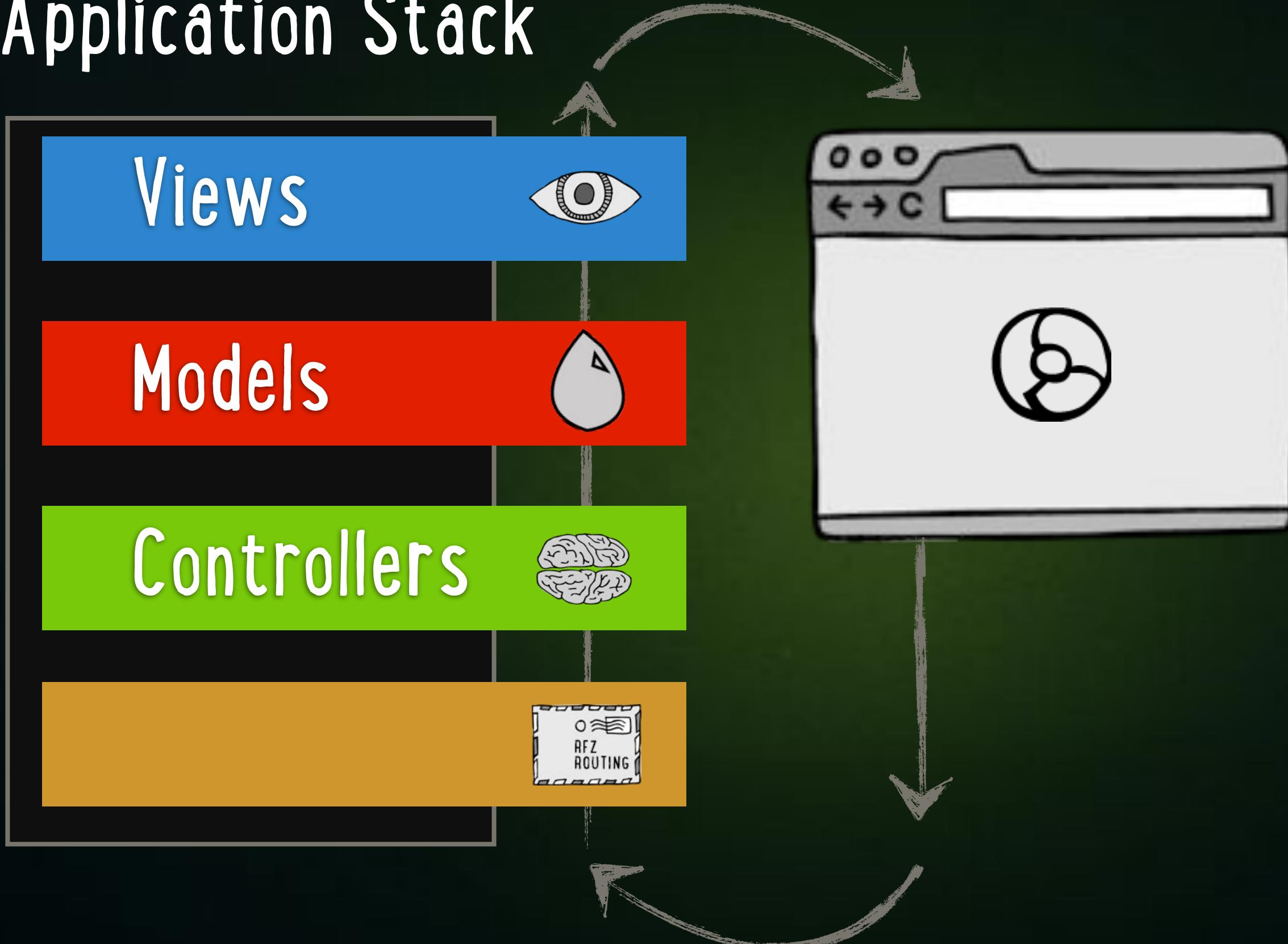


The Brains of the application



Application Stack

ROUTING



Show A tweet

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

```
<!DOCTYPE html>
<html>
  <head><title>Twitter for Zombies</title>
  <body>
    <header>...</header>

    <% tweet = Tweet.find(1) %>
    <h1><%= tweet.status %></h1>
    <p>Posted by <%= tweet.zombie.name %>
  </body>
</html>
```



This code tastes like rotted brains. We will fix it later.

CONTROLLERS

Request /tweets/1

/app/controllers/tweets_controller.rb

Controllers



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

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= tweet.zombie.name %></p>
```



zombie_twitter



app



controllers



tweets_controller.rb



CONTROLLERS

Request /**tweets/1**

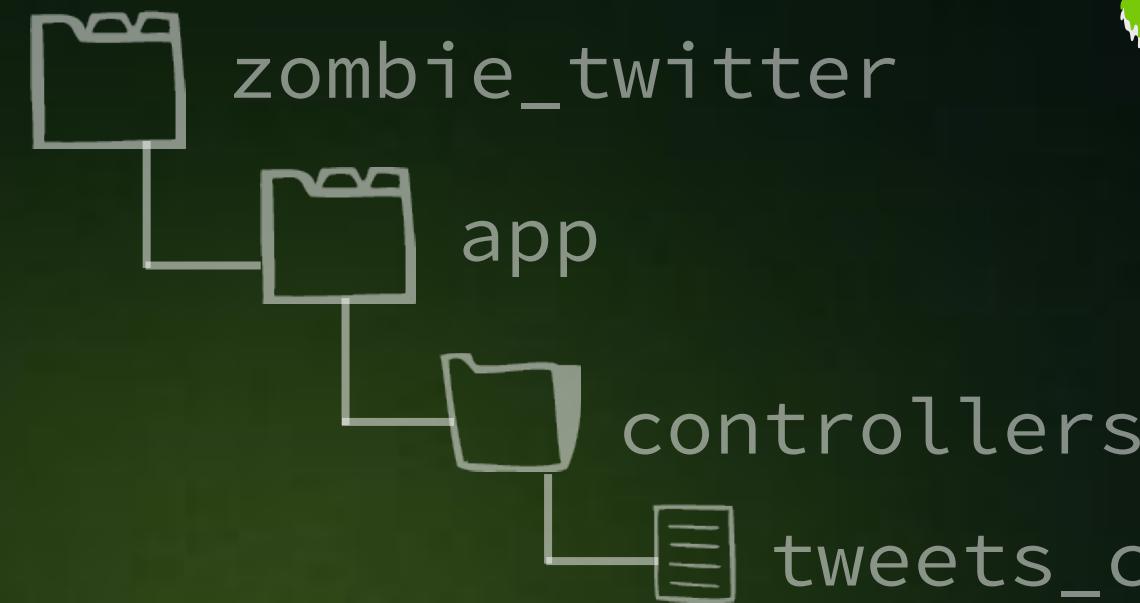
/app/controllers/**tweets_controller.rb**

Controllers



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

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= tweet.zombie.name %></p>
```



It is no coincidence that the word 'tweets' is found in the URL, the controller name, and the view folder.



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



REQUEST



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



REQUEST



Request /tweets/1

REQUEST

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



Request /tweets/1

REQUEST

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
  end
end
```

This is where we typically call our models.
So let's fix our code!



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

```
<% tweet = Tweet.find(1) %>
<h1><%= tweet.status %></h1>
<p>Posted by <%= tweet.zombie.name %></p>
```



REQUEST

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>
```



Student Question: What about variable scope?



REQUEST

Request /tweets/1

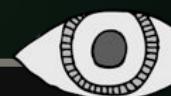
/app/controllers/tweets_controller.rb

```
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: grants view access to variables with @



Rendering a Different View

REQUEST

/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>
```



Accepting Parameters

/app/controllers/tweets_controller.rb

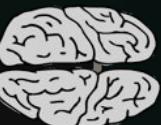
```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find(params[:id]) ←
    render action: 'status'
  end
end
```



Params Recipe: params = { id: "1" }

/tweets/1 /tweets/4
/tweets/2 /tweets/5
/tweets/3 /tweets/6

REQUEST



PARAMS

Parameters

```
/tweets?status=I'm dead  
params = { status: "I'm dead" }  
@tweet = Tweet.create(status: params[:status])
```

```
/tweets?tweet[status]=I'm dead  
params = { tweet: {status: "I'm dead"} }
```

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

Alternate Syntax

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



PARAMS

Parameters

```
/tweets?tweet[status]=I'm dead
```

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

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

- Rotten Code - 

This could be dangerous! Users might
be able to set any attributes!

In Rails 4 we are required to use Strong Parameters

We need to specify the parameter key we require

```
require(:tweet)
```

And the attributes we will permit to be set

```
permit(:status)
```



PARAMS

Strong Parameters

```
/tweets?tweet[status]=I'm dead
```

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

```
@tweet = Tweet.create(params.require(:tweet).permit(:status))
```



If there were multiple things we needed to permit, we could use an array

```
params.require(:tweet).permit[:status, :location]
```

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

Strong Params Required only when:

CREATING or UPDATING

with MULTIPLE Attributes



Respond with XML or 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>
```

/tweets/1

PARSE

JSON

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

Respond with HTML or JSON

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find( params[:id] )
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @tweet }
    end
  end
end
```

/tweets/1.json

PARSE



JSON

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

Respond with HTML, JSON, XML

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  def show
    @tweet = Tweet.find( params[:id] )
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @tweet }
      format.xml { render xml: @tweet }
    end
  end
end
```

/tweets/1.xml

PARSE



XML

```
<?xml version="1.0" encoding="UTF-8"?> ...
```



Controller Actions

/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
```



REQUEST

Controller Actions

EDIT

/app/controllers/tweets_controller.rb

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

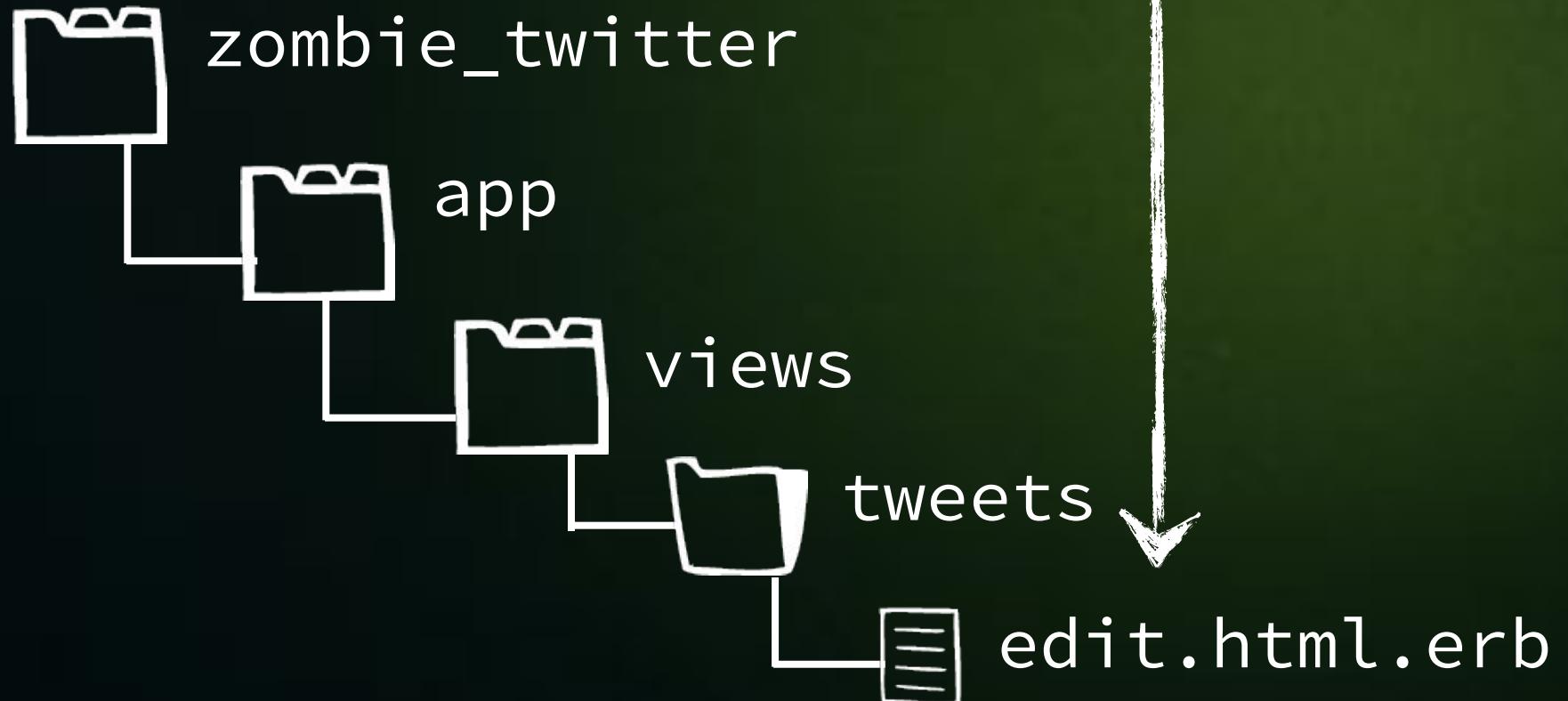
Listing tweets		
Status	Zombie	
<u>Where can I get a good bite to eat?</u>	Ash	Edit Destroy
<u>My left arm is missing, but I dont care.</u> Bob	Bob	Edit Destroy
<u>I just ate some delicious brains.</u> Jim	Jim	Edit Destroy
<u>OMG, my fingers turned green. #FML</u> Ash	Ash	Edit Destroy
<u>Your eyelids taste like bacon.</u> Bob	Bob	Edit Destroy

EDIT

The Edit Action

/app/controllers/tweets_controller.rb

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



EDIT

Adding Some Authorization

A screenshot of a web browser window titled "RailsForZombies". The address bar shows "localhost:3000/tweets/3/edit". The main content area displays a form for editing a tweet. At the top is a decorative header featuring a green skull and crossbones on a dark background with yellow diagonal stripes. Below it, the title "Editing tweet" is displayed in large white font. The form has two text input fields: one for "Status" containing the text "Where can I get a good bite" and another for "Zombie" containing the number "1". A small "Update Tweet" button is located at the bottom left of the form.

Show | Back



Adding Some Authorization

DELETE

RailsForZombies

localhost:3000/tweets/

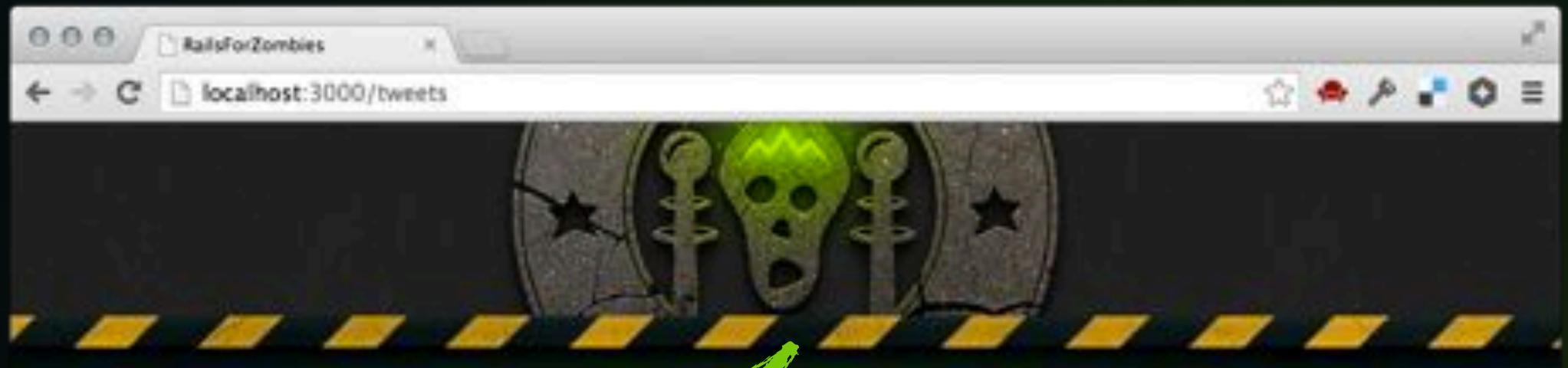
Listing tweets

Status	Zombie	
Where can I get a good bite to eat?	Ash	Edit Destroy
My left arm is missing, but I dont care.	Bob	Edit Destroy



Adding Some Authorization

DELETE



Listing tweets

Status	Zombie	
Where can I get a good bite to eat?	Ash	Edit Destroy
My left arm is missing, but I don't care.	Bob	Edit Destroy
I just ate some delicious brains.	Jim	Edit Destroy



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

flash[:notice]

redirect <path>

Works like a per user hash

To send messages to the user

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
      flash[:notice] = "Sorry, you can't edit this tweet"
      redirect_to(tweets_path)
    end
  end
end
```

Flash + Redirect



Alternate Recipe: `redirect_to(tweets_path, :notice => "Sorry, you can't edit this tweet")`

Notice for Layouts

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

```
<!DOCTYPE html>
<head>
  <title>Twitter for Zombies</title>
</head>
<body>
  
  <%= yield %>
</body>
</html>
```



Notice for Layouts

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

```
<!DOCTYPE html>
<head>
  <title>Twitter for Zombies</title>
</head>
<body>
  
  <% if flash[:notice] %>
    <div id="notice"><%= flash[:notice] %></div>
  <% end %>
  <%= yield %>
</body>
</html>
```



Adding Some Authorization

DELETE

RailsForZombies

localhost:3000/tweets/

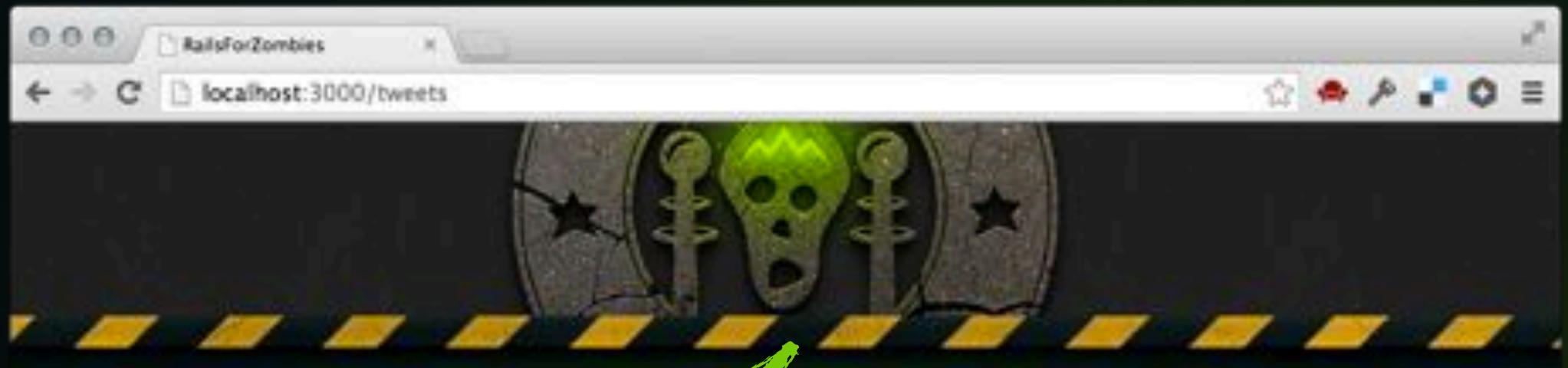
Listing tweets

Status	Zombie	
<u>Where can I get a good bite to eat?</u>	Ash	Edit Destroy
<u>My left arm is missing, but I dont care.</u>	Bob	Edit Destroy



Adding Some Authorization

DELETE



Listing tweets

Status	Zombie	
Where can I get a good bite to eat?	Ash	Edit Destroy
My left arm is missing, but I don't care.	Bob	Edit Destroy
I just ate some delicious brains.	Jim	Edit Destroy



Controller Actions

/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
```



Controller Actions

/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
```



Need Authorization



Before Actions

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 ACTIONS

/app/controllers/tweets_controller.rb

```
class TweetsController < ApplicationController
  before_action :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
```

/app/controllers/tweets_controller.rb

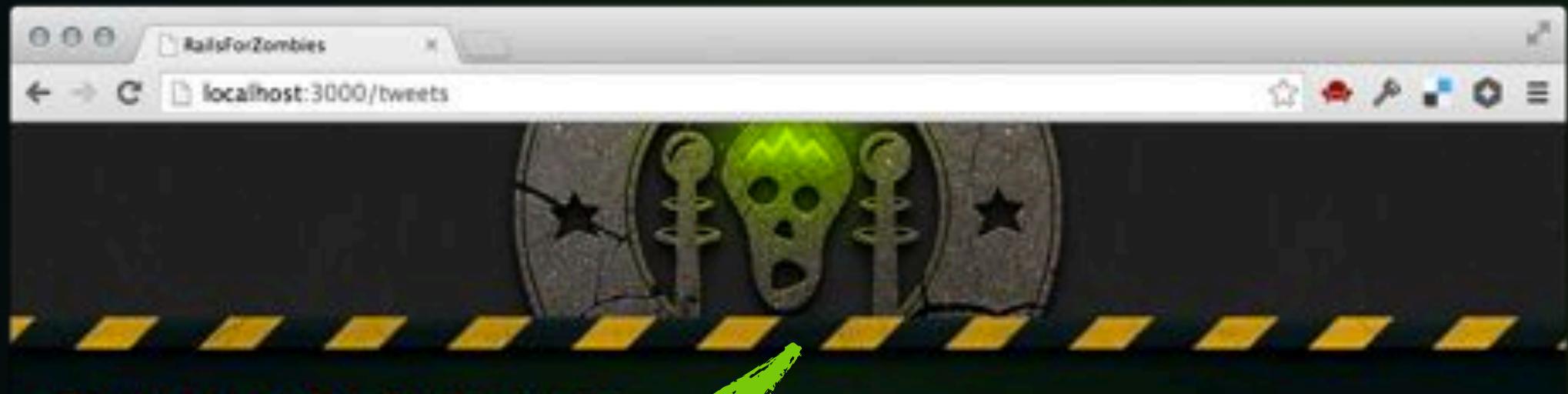
BEFORE ACTIONS

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



Adding Some Authorization

DELETE



Listing tweets

Status	Zombie	
Where can I get a good bite to eat?	Ash	Edit Destroy
My left arm is missing, but I don't care.	Bob	Edit Destroy
I just ate some delicious brains.	Jim	Edit Destroy



ZOMBIE LAB 4



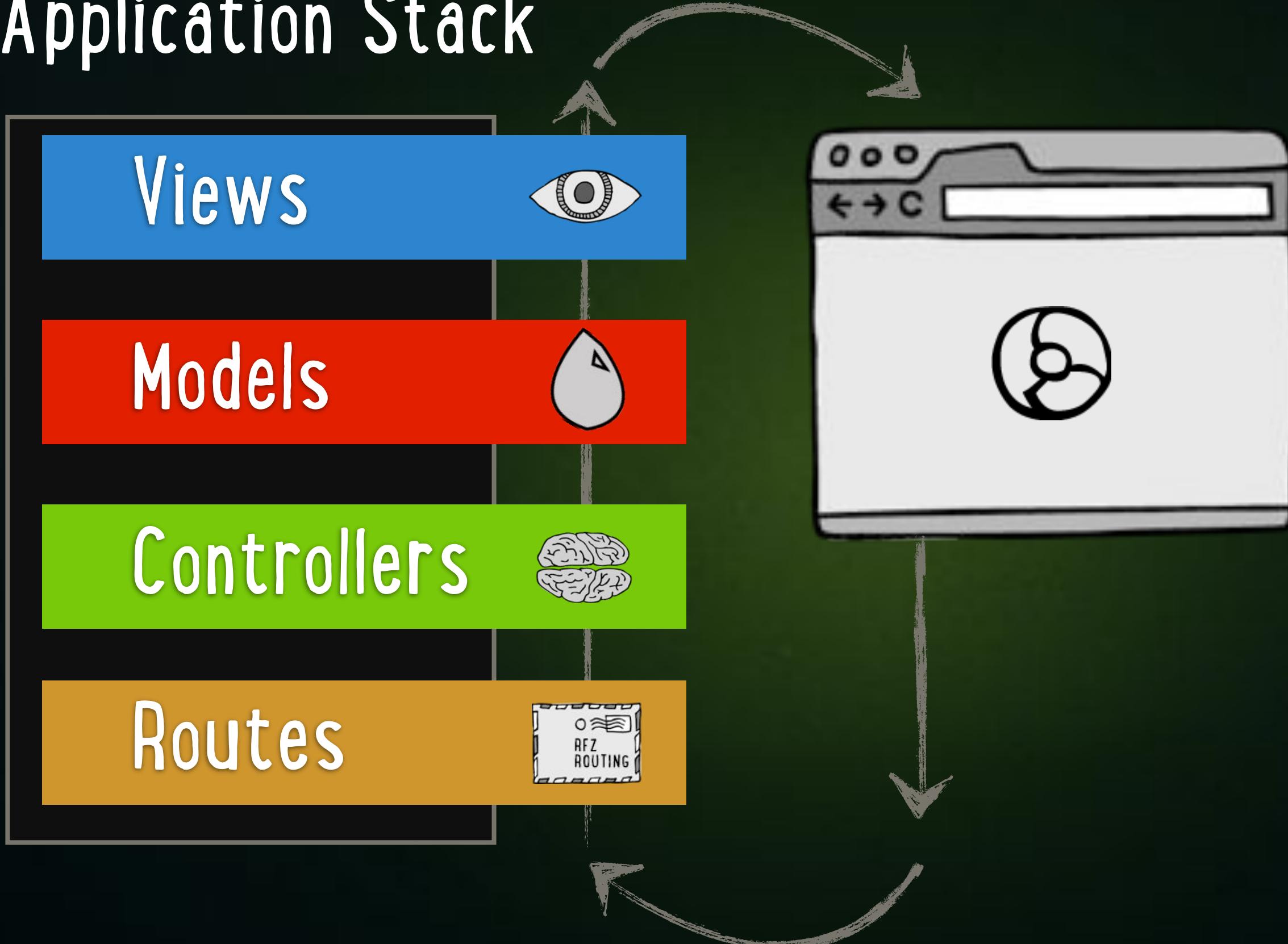
ROUTING THROUGH RAILS

LEVEL 5



Application Stack

ROUTING



ROUTING

In order to properly find these paths...

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



Action	Code	The URL Generated
List all tweets	<code>tweets_path</code>	<code>/tweets</code>
New tweet form	<code>new_tweet_path</code>	<code>/tweets/new</code>

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

These paths need a tweet

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

REQUEST

and these actions...

/app/controllers/tweets_controller.rb

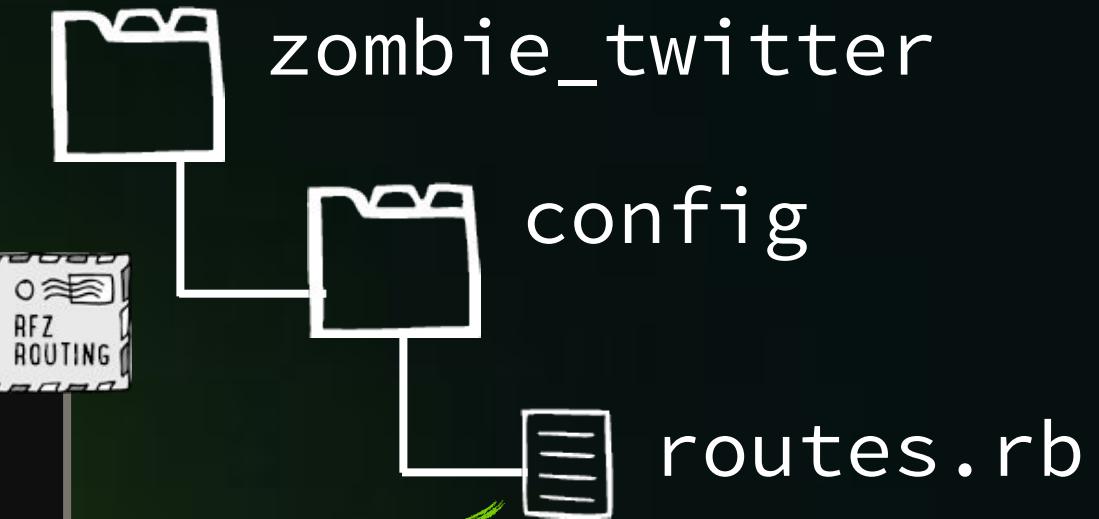
```
class TweetsController < ApplicationController  
  def index    List all tweets  
  def show     Show a single tweet  
  def new      Show a new tweet form  
  def edit      Show an edit tweet form  
  def create    Create a new tweet  
  def update    Update a tweet  
  def destroy   Delete a tweet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



We need to define Routes

Creating what we like to call a "REST"FUL resource

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



Code	The URL	TweetsController
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....		

ROUTING

Custom Routes

`http://localhost:3000/new_tweet`

Controller name	Tweets
Action name	new

render

`http://localhost:3000/tweets/new`

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



`/config/routes.rb`

```
ZombieTwitter::Application.routes.draw do  
  resources :tweets  
  get '/new_tweet' => 'tweets#new'  
end
```

Path

Controller

Action



ROUTING

RailsForZombies

localhost:3000/tweets/new_tweet

New tweet

Status

Zombie



ROUTING

Named Routes

`http://localhost:3000/all`

Controller name	Tweets
Action name	index

render

`http://localhost:3000/tweets`

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



`/config/routes.rb`

```
get '/all' => 'tweets#index'
```



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



`tweets_path` wouldn't work



ROUTING

Named Routes

`http://localhost:3000/all`

Controller name	Tweets
Action name	index

render

`http://localhost:3000/tweets`

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



`/config/routes.rb`

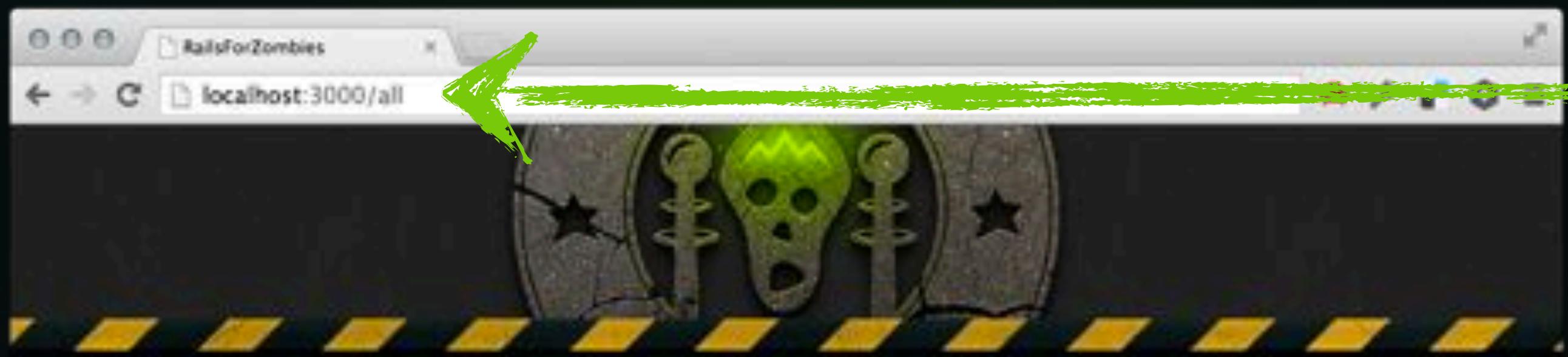
```
get '/all' => 'tweets#index', as: 'all_tweets'
```



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



ROUTING



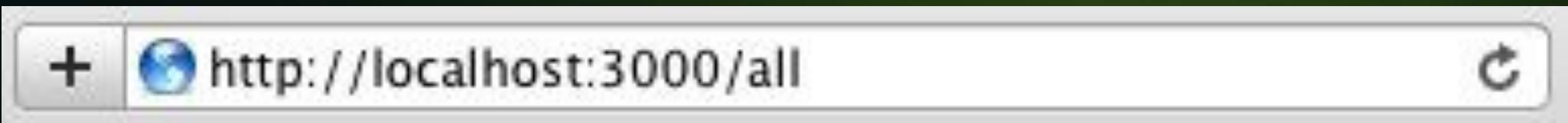
Listing tweets

Status	Zombie	
Where can I get a good bite to eat?	Ash	Edit Destroy
My left arm is missing, but I don't care. Bob		Edit Destroy
I just ate some delicious brains.	Jim	Edit Destroy

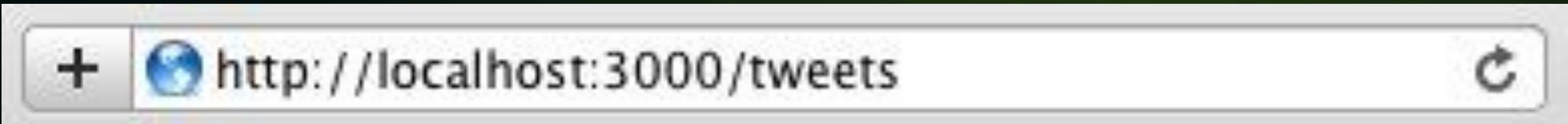


What if our tweets used to be found at `/all`,
but now our definitive URL is `/tweets`
What can we do?

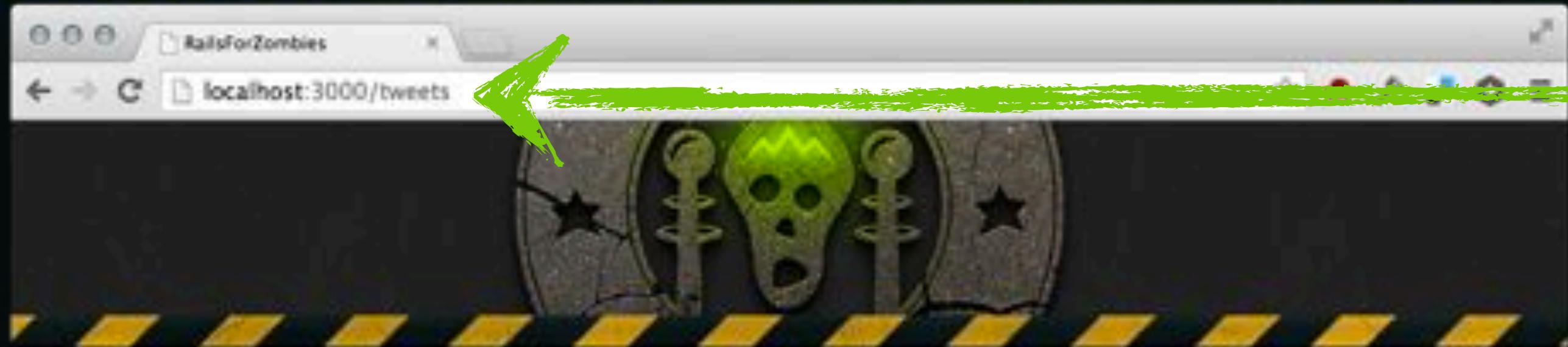
ROUTING



redirects to



ROUTING



Listing tweets

Status

Zombie

Where can I get a good bite to eat?

Ash

[Edit](#) [Destroy](#)

My left arm is missing, but I don't care. Bob

[Edit](#) [Destroy](#)

I just ate some delicious brains.

Jim

[Edit](#) [Destroy](#)



ROUTING

Redirect

`http://localhost:3000/all`

`redirect_to`

`http://localhost:3000/tweets`

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

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



ROUTING

Root Route

`http://localhost:3000/`

`root to: "tweets#index"`

Controller

Action

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

render

`http://localhost:3000/tweets`



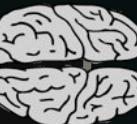
Route Parameters

/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
```

/local_tweets/32828
/local_tweets/32801

*Find all zombie tweets
in this zip code*



Route Parameters

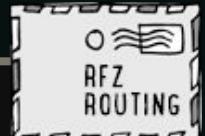
/local_tweets/32828
/local_tweets/32801

*Find all zombie
in this zip code*

```
get '/local_tweets/:zipcode' => 'tweets#index'
```

referenced by params[:zipcode] in controller

```
get '/local_tweets/:zipcode'  
=> 'tweets#index', as: 'local_tweets'
```



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



ROUTING

A screenshot of a web browser window displaying the GitHub Twitter profile at <http://twitter.com/github>. The browser interface includes a toolbar with icons for back, forward, search, and refresh, and a tab bar with the current URL.

The Twitter profile page for GitHub shows the following information:

- Profile Picture:** A GitHub logo featuring a black cat icon.
- Username:** **github**
- Follow:** Follow button.
- Lists:** Lists dropdown menu.
- Tweets:** 1,385 tweets.
- Favorites:** 1,541 favorites.
- Following:** 12 users.
- Followers:** 15,358 followers.
- Blocked:** 1,541 blocked users.

The profile bio reads: "Name GitHub
Location San Francisco, CA
Web <http://github.com>
Bio Social coding? Pretty awesome."

The timeline on the left side of the page displays the following tweets:

- @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/9GL150>
12:29 AM Aug 23rd 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
- CiTHub Meetup SF this Thursday!



ROUTING

A screenshot of a web browser window displaying the GitHub Twitter profile at <http://twitter.com/github>. The browser interface includes a toolbar with back, forward, and search buttons, and a tab bar showing the current URL.

The Twitter profile page for GitHub shows the following information:

- Profile Picture:** The GitHub logo (a black cat icon).
- Username:** github
- Follow:** Follow button.
- Lists:** Lists dropdown menu.
- Settings:** Settings gear icon.
- Followers:** 15,358 followers.
- Following:** 12 following.
- Listed:** 1,541 listed.
- Tweets:** 1,385 tweets.
- Favorites:** No favorites shown.
- Actions:** Block GitHub and Report for spam.
- You both follow:** A small user icon.
- Following:** A grid of 12 user icons.
- More like github:** Suggested users: darkhelmetlive (Daniel Huckstep), and a "Follow" link.

The timeline on the left side of the page displays the following tweets:

- @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
- CiHub Meetup SF this Thursday!



Route Parameters

/eallam /envylabs
/greggpollack /github

Show the tweets
for these zombies



```
get ':name' => 'tweets#index', as: 'zombie_tweets'
```

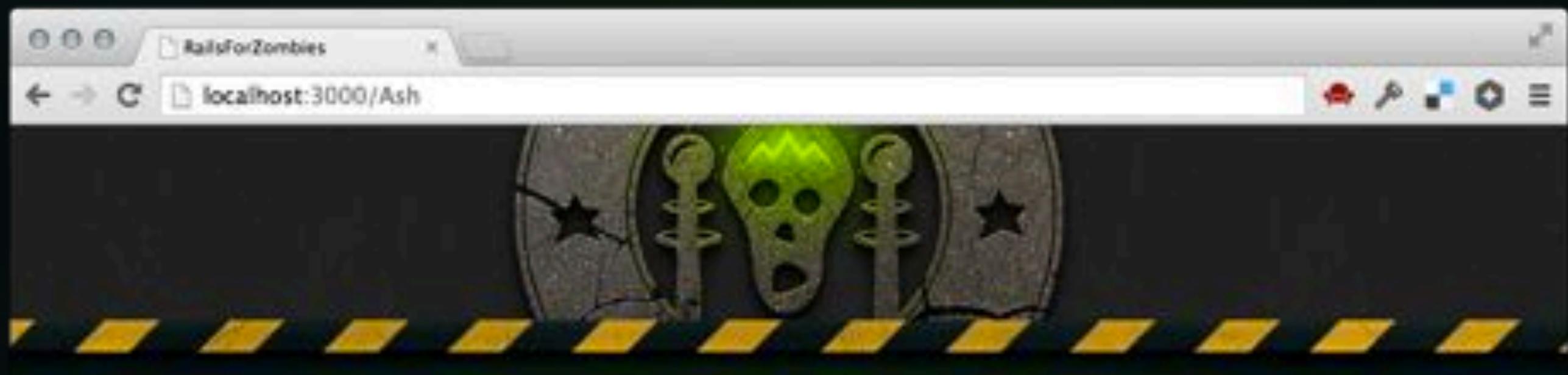
```
<%= link_to "Gregg", zombie_tweets_path('greggpollack') %>
```

/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
```

ROUTING



Listing tweets

Status

Zombie

Where can I get a good bite to eat? Ash [Edit](#) [Destroy](#)

OMG. my fingers turned green. #FML Ash [Edit](#) [Destroy](#)



ZOMBIE LAB 5

