Rails in a Day

Doug Alcorn

Chris Nelson

Rob Biedenharn

gaslightsoftware

Key concepts

- Model-View-Controller (MVC)
- Convention over configuration

Models

- Represent the Things in your system
- Live in app/models
- Map to database tables
- Columns magically become attributes

Controllers

- Respond to web requests
- Coordinate between models and views
- Live in app/controllers
- Inherit from ApplicationController

gaslightsoftware

Views

- Provide the UI for your web app
- Consist of HTML with embedded Ruby (erb)
- Live in app/views
- Normally one for each controller method
- Layouts (in app/views/layouts) hold the shared HTML code that gives a common “look”

Environments

- Development
- Test
- Production

gaslightsoftware

Rails in a Day

- Getting to the starting gate
  - Change to the project directory

```
cd Suggestion-Box
git checkout -b private dor_start
```

  - If you already played around a bit and need to reset

```
git checkout private
git reset --hard dor_start
```

Rails in a Day

Create the development (and test) databases

```
$ rake db:create
```

Run all the unapplied migrations

```
$ rake db:migrate
```

Start a local server to run the application

```
$ rails server
c:\Suggestion-Box>ruby script\rails server
```

Open a browser to visit the application:

http://localhost:3000/

gaslightsoftware

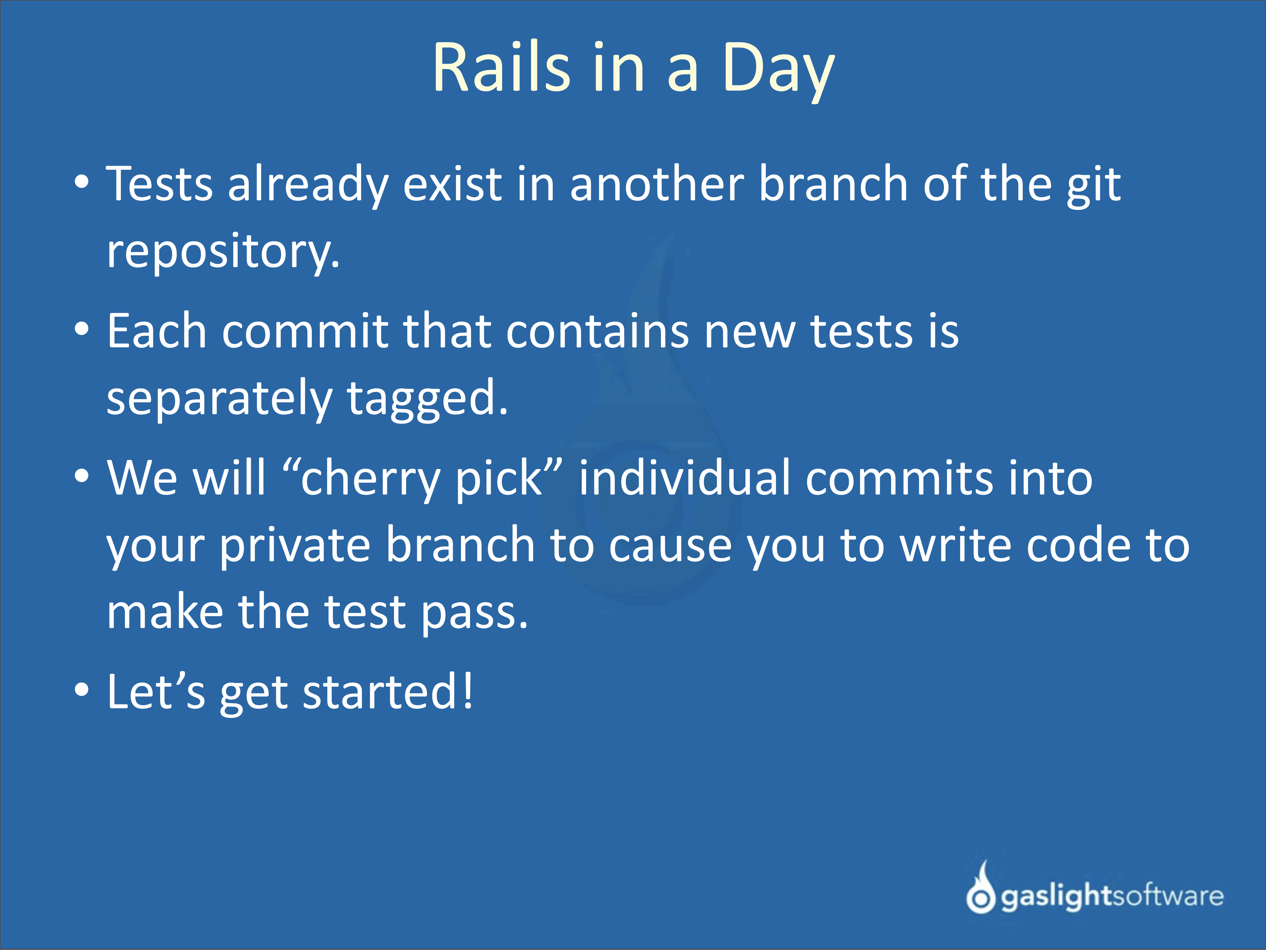
There should only be a single migration to create the topics table.
Don’t worry about what’s in a migration as we’ll be creating a new one shortly.

Rails in a Day

- Test Driven Development
  - –Start with a failing test
    - You got these handed to you all at once in the Ruby Koans
  - –Write (or edit) code to make the test pass
    - but only *just enough* code
  - –Contemplate whether the code could be improved
    - changing the implementation details without changing the behavior is called *refactoring*

- All together you get: ***Red***, ***Green***, ***Refactor***

gaslightsoftware



Rails in a Day

- Tests already exist in another branch of the git repository.
- Each commit that contains new tests is separately tagged.
- We will “cherry pick” individual commits into your private branch to cause you to write code to make the test pass.
- Let’s get started!

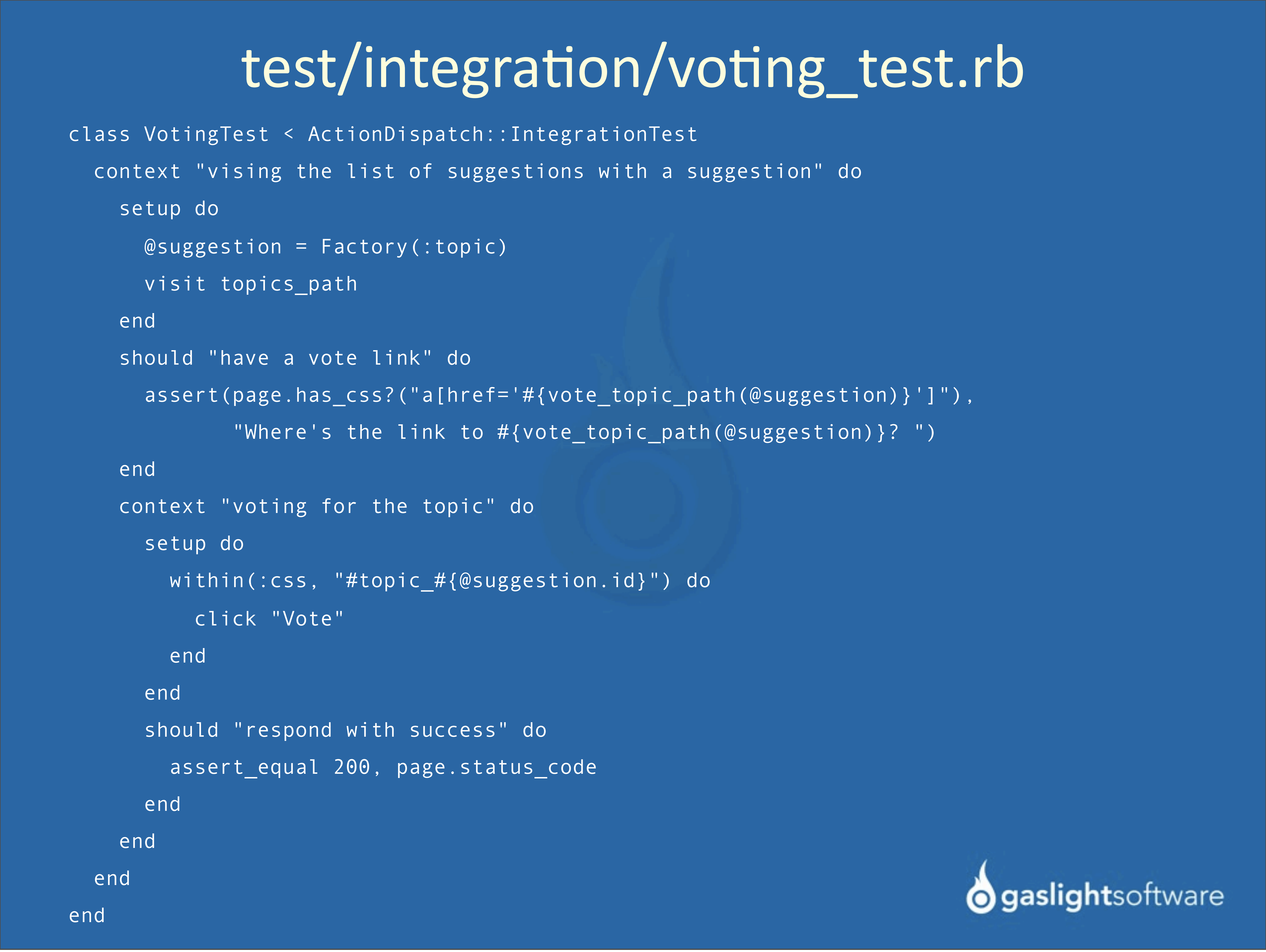
Note that before we can cherry-pick a commit, the working area typically has to be “clean” so we’ll save our work each time before we grab a new test.

Rails in a Day

“As an interested person, I’d like to have a link to express my vote in favor of a topic.”

```
$ git cherry-pick dor_test_1
```

Well, let’s get a test for that!



test/integration/voting_test.rb

```
class VotingTest < ActionDispatch::IntegrationTest
  context "vising the list of suggestions with a suggestion" do
    setup do
      @suggestion = Factory(:topic)
      visit topics_path
    end
    should "have a vote link" do
      assert(page.has_css?("a[href='#{vote_topic_path(@suggestion)}']"),
             "Where's the link to #{vote_topic_path(@suggestion)}? ")
    end
    context "voting for the topic" do
      setup do
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      should "respond with success" do
        assert_equal 200, page.status_code
      end
    end
  end
end
```

test/integration/voting_test.rb

test/integration/voting_test.rb

```ruby
class VotingTest < ActionDispatch::IntegrationTest
  context "vising the list of suggestions with a suggestion" do
    setup do
      @suggestion = Factory(:topic)
      visit topics_path
    end
    should "have a vote link" do
      assert(page.has_css?("a[href='#{vote_topic_path(@suggestion)}']"),
             "Where's the link to #{vote_topic_path(@suggestion)}? ")
    end
    context "voting for the topic" do
      setup do
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      should "respond with success" do
        assert_equal 200, page.status_code
      end
    end
  end
end
```

test/integration/voting_test.rb

```ruby
class VotingTest < ActionDispatch::IntegrationTest
  context "vising the list of suggestions with a suggestion" do
    setup do
      @suggestion = Factory(:topic)
      visit topics_path
    end
    should "have a vote link" do
      assert(page.has_css?("a[href='#{vote_topic_path(@suggestion)}']"),
             "Where's the link to #{vote_topic_path(@suggestion)}? ")
    end
    context "voting for the topic" do
      setup do
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      should "respond with success" do
        assert_equal 200, page.status_code
      end
    end
  end
end
```

test/integration/voting_test.rb

```ruby
class VotingTest < ActionDispatch::IntegrationTest
  context "vising the list of suggestions with a suggestion" do
    setup do
      @suggestion = Factory(:topic)
      visit topics_path
    end
    should "have a vote link" do
      assert(page.has_css?("a[href='#{vote_topic_path(@suggestion)}']"),
             "Where's the link to #{vote_topic_path(@suggestion)}? ")
    end
    context "voting for the topic" do
      setup do
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      should "respond with success" do
        assert_equal 200, page.status_code
      end
    end
  end
end
```

gaslightsoftware

test/integration/voting_test.rb

```ruby
class VotingTest < ActionDispatch::IntegrationTest
  context "vising the list of suggestions with a suggestion" do
    setup do
      @suggestion = Factory(:topic)
      visit topics_path
    end
    should "have a vote link" do
      assert(page.has_css?("a[href='#{vote_topic_path(@suggestion)}']"),
             "Where's the link to #{vote_topic_path(@suggestion)}? ")
    end
    context "voting for the topic" do
      setup do
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      should "respond with success" do
        assert_equal 200, page.status_code
      end
    end
  end
end
```

gaslightsoftware

rake test:integration

```
Started
....EE
Finished in 1.958562 seconds.

  1) Error:
test: vising the list of suggestions with a
suggestion should have a vote link.
(VotingTest):
NoMethodError: undefined method
`vote_topic_path' for #<VotingTest:0x2792668>
```

Need to create a route such that ‘vote_topic_path’ will be valid

Routing in Rails

- Happens in config/routes.rb
- Tells how to map URLs to controller methods

REST

- A way to represent things the live on the web
- Makes use of the 4 HTTP verbs
  - GET
  - POST
  - PUT
  - DELETE

gaslightsoftware

REST in Rails

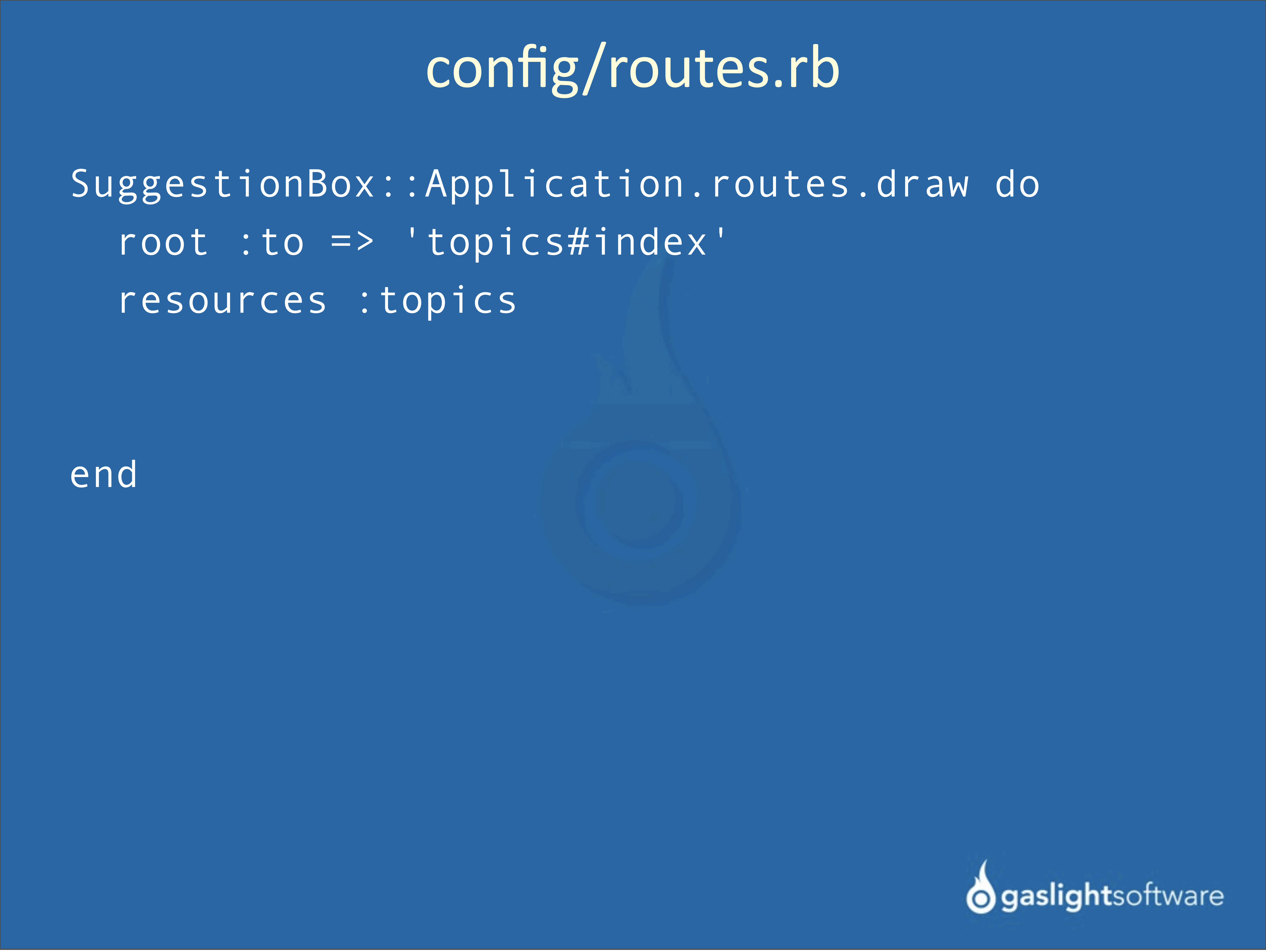
GET /topics	topics_controller#index
GET /topics/new	topics_controller#new
GET /topics/1	topics_controller#show
GET /topics/1/edit	topics_controller#edit
POST /topics	topics_controller#create
PUT /topics/1	topics_controller#update
DELETE /topics/1	topics_controller#destroy

gaslightsoftware

Try it!

```
$ rake routes
```

gaslightsoftware



config/routes.rb

```
SuggestionBox::Application.routes.draw do
  root :to => 'topics#index'
  resources :topics



end
```

gaslightsoftware

You’ll see a bunch of comments in your copy of the config/routes.rb file

Explain: root
resources

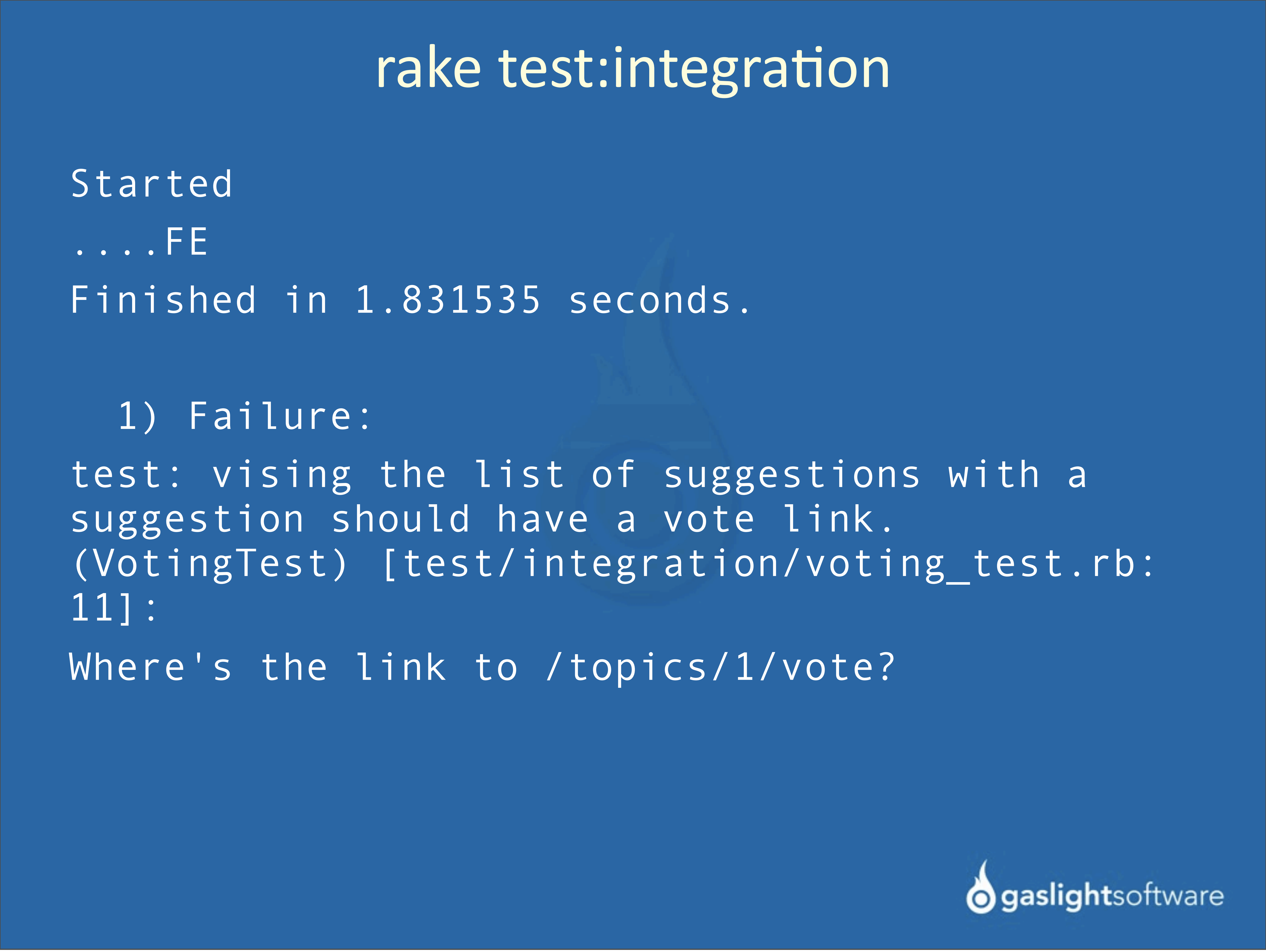
config/routes.rb

```
SuggestionBox::Application.routes.draw do
  root :to => 'topics#index'
  resources :topics do
    member { get :vote }
  end
end
```

What we just did

GET /topics/1/vote	topics_controller#vote

```
$ rake routes
```

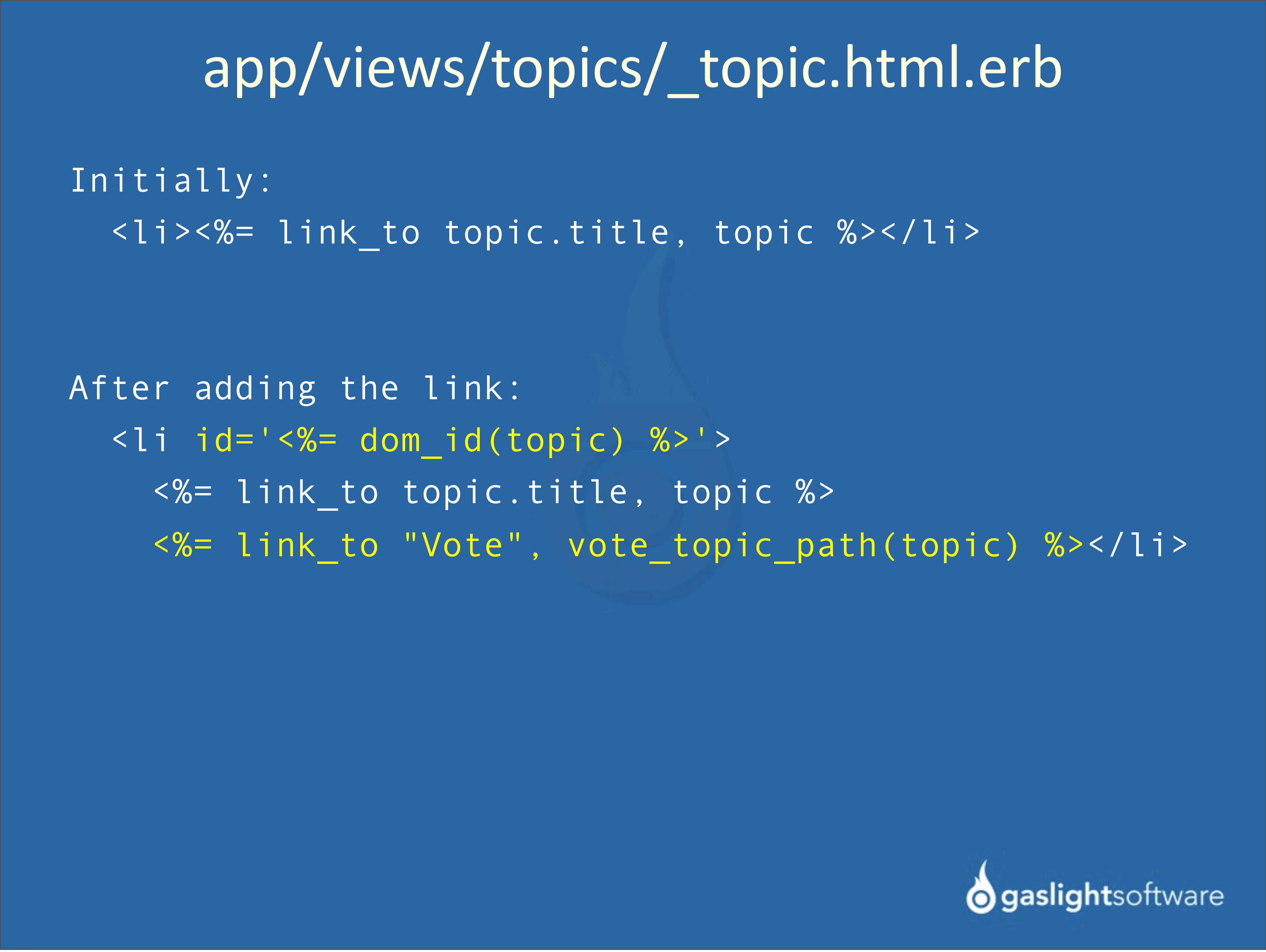


rake test:integration

```
Started
....FE
Finished in 1.831535 seconds.

  1) Failure:
test: vising the list of suggestions with a
suggestion should have a vote link.
(VotingTest) [test/integration/voting_test.rb:
11]:
Where's the link to /topics/1/vote?
```

YES! Now the vote_topic_path is defined, but we have to add the link to the listing, too.

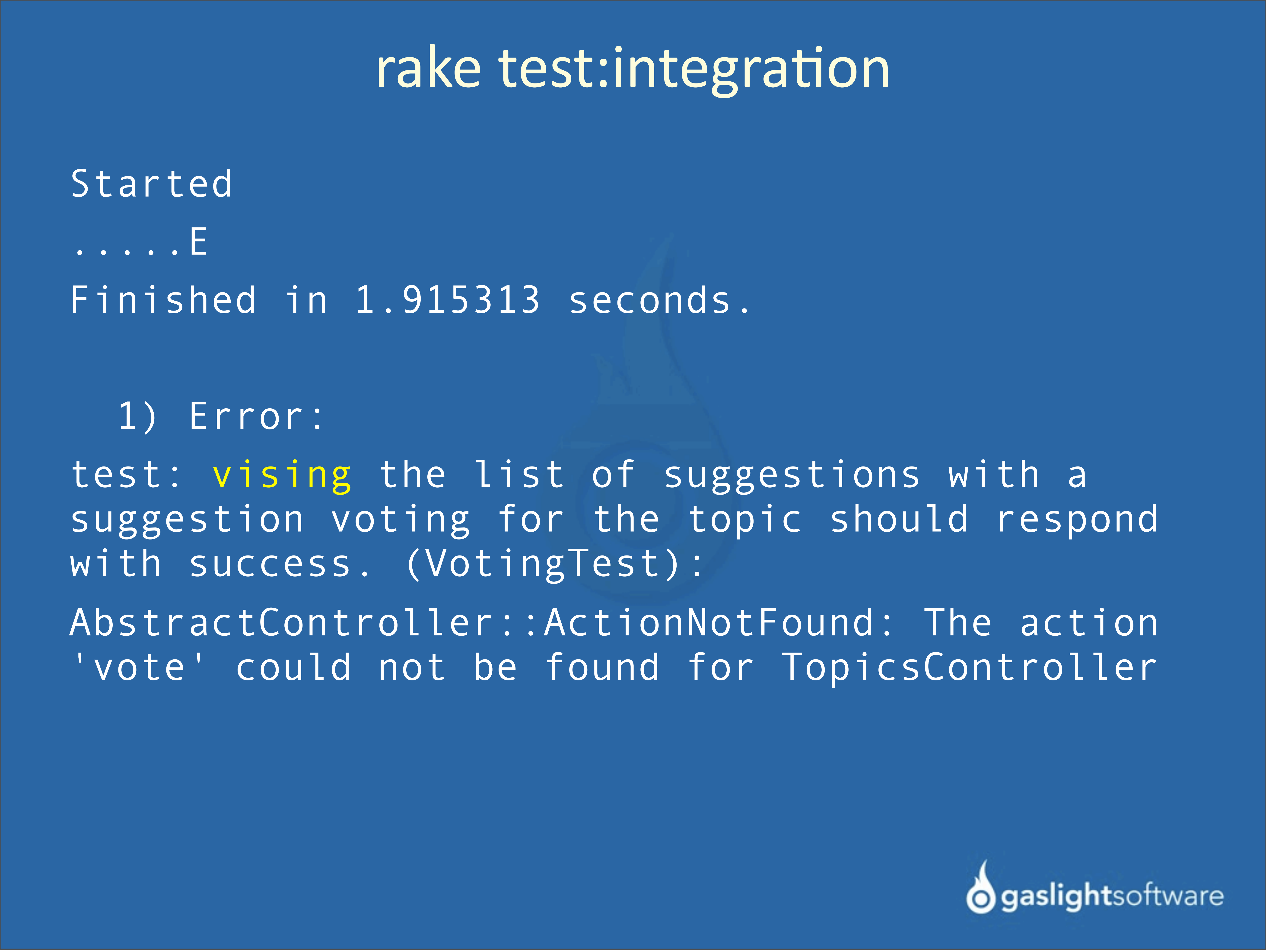


app/views/topics/_topic.html.erb

```
Initially:
  <li><%= link_to topic.title, topic %></li>


After adding the link:
  <li id='<%= dom_id(topic) %>'>
    <%= link_to topic.title, topic %>
    <%= link_to "Vote", vote_topic_path(topic) %></li>
```

Refresh your browser to see the new link!



rake test:integration

```
Started
.....E
Finished in 1.915313 seconds.

  1) Error:
test: vising the list of suggestions with a
suggestion voting for the topic should respond
with success. (VotingTest):
AbstractController::ActionNotFound: The action
'vote' could not be found for TopicsController
```

Argh! So close. Now there needs to be an action to be the target of the vote link.

app/controllers/topics_controller.rb

```
class TopicsController < ApplicationController
  # ... other actions ...

  def vote
    render :nothing => true
  end
end
```

gaslightsoftware

Sure, it doesn’t really *do* anything, but will the test pass?

rake test:integration

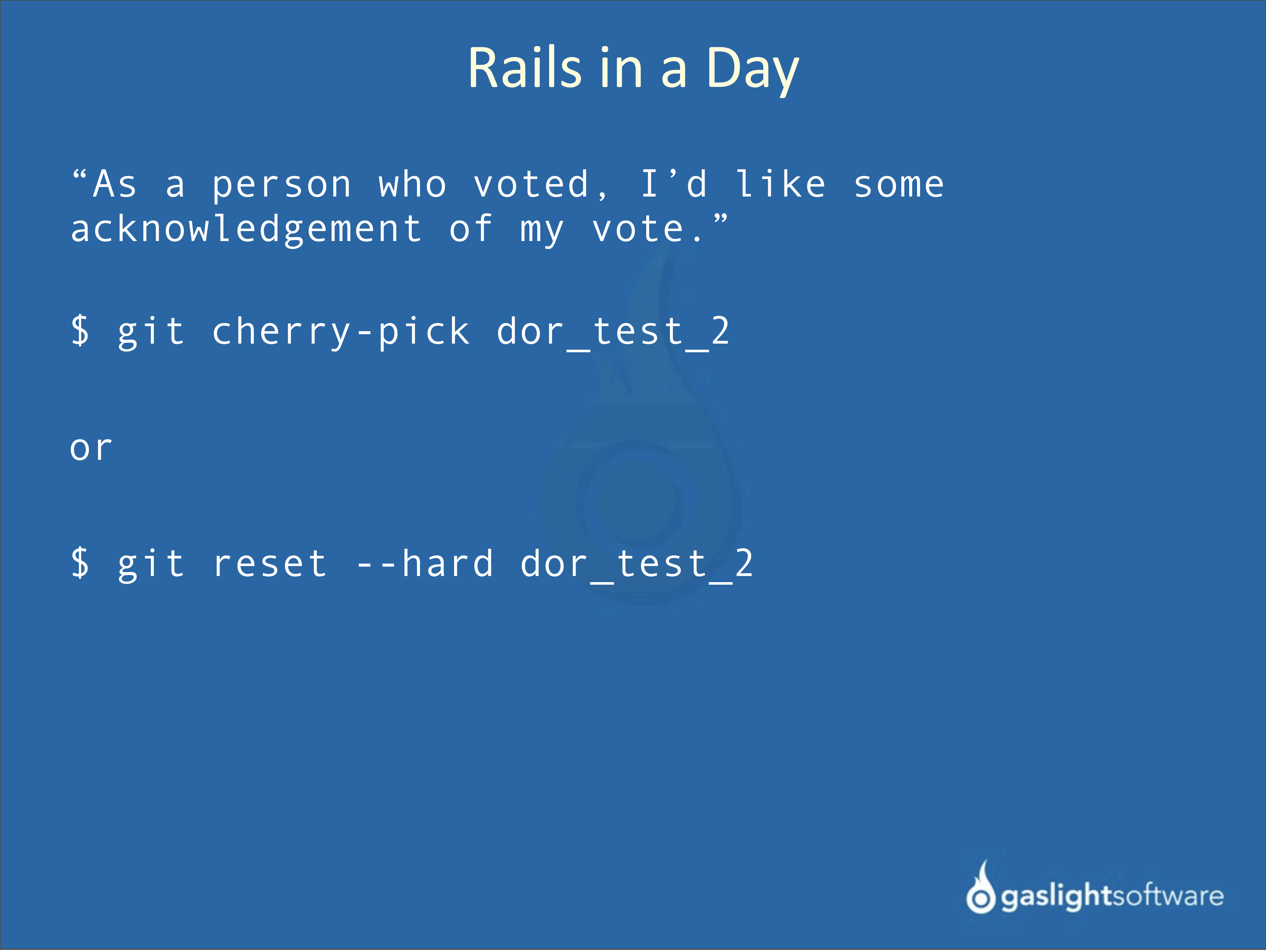
```
Started
......
Finished in 1.892785 seconds.

6 tests, 8 assertions, 0 failures, 0 errors, 0 skips
```

And there was much rejoicing!

Rails in a Day

- Save your work!
  - git status
  - git add .
  - git commit -a -m ‘done with first lesson’



Rails in a Day

“As a person who voted, I’d like some acknowledgement of my vote.”

```
$ git cherry-pick dor_test_2
```

or

```
$ git reset --hard dor_test_2
```

gaslightsoftware

Well, let’s get a test for that!

test/integration/voting_test.rb

```
class VotingTest < ActionDispatch::IntegrationTest
  # ...
    context "voting for the topic" do
      setup do
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      should "have a message in flash" do
        within(:css, ".flash") do
          assert page.has_content?("Thanks"), "Not even a thank you?"
        end
      end
      should "respond with success" do
        assert_equal 200, page.status_code
      end
    end
  # ...
end
```

gaslightsoftware

rake test:integration

```
Started
.....E.
Finished in 2.027329 seconds.

  1) Error:
test: visiting the list of suggestions with a
suggestion voting for the topic should have a
message in flash. (VotingTest):
Capybara::ElementNotFound: scope '//
*[contains(concat(' ', @class, ' '), ' flash
')]' not found on page
```

gaslightsoftware

app/views/layouts/application.html.erb

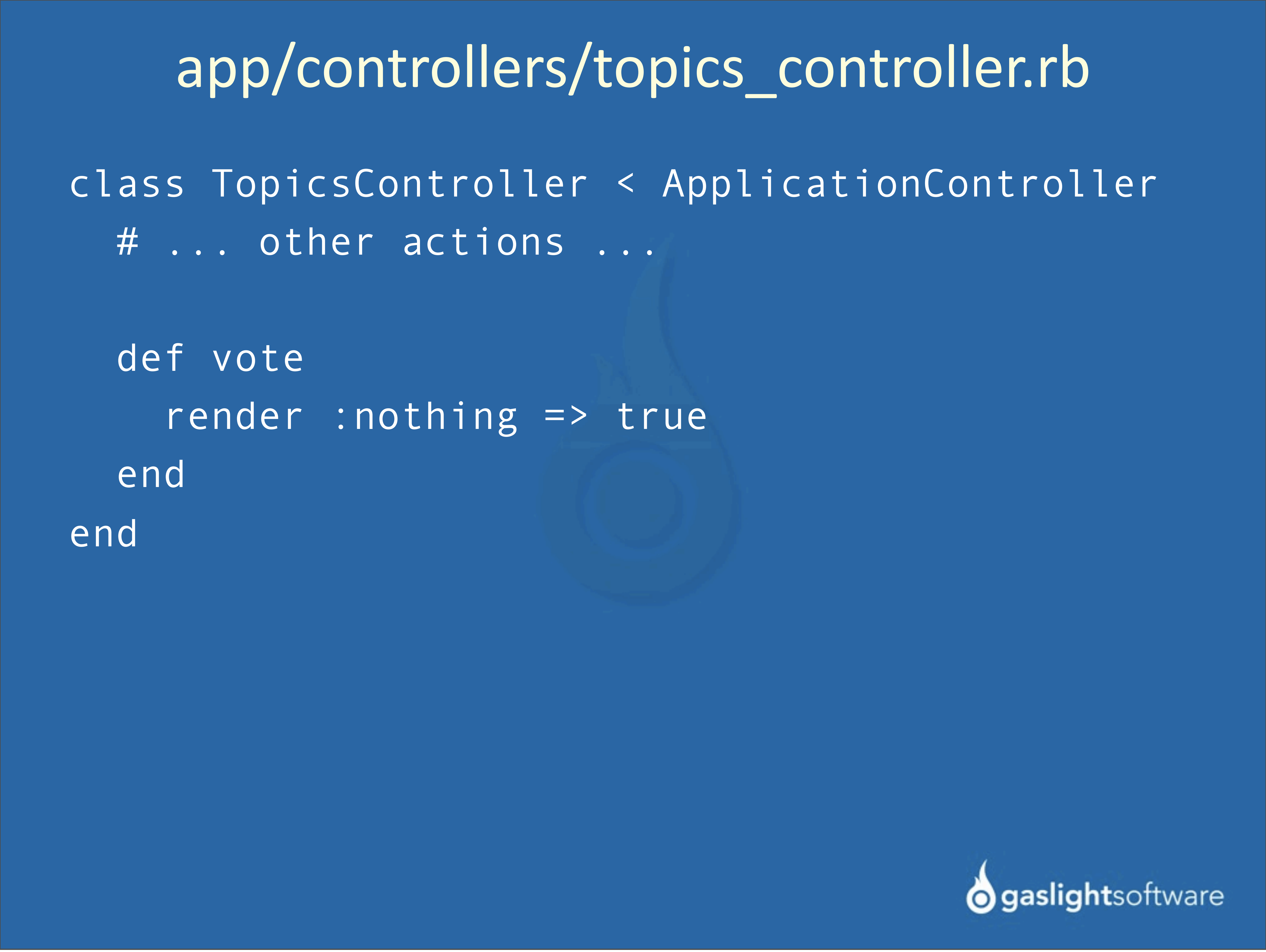
```
<!DOCTYPE html>
<html>
<head>
  <title>Cincinnati.rb Suggestion Box</title>
  <%= stylesheet_link_tag :all %>
  <%= javascript_include_tag :defaults %>
  <%= csrf_meta_tag %>
  <%= formtastic_stylesheet_link_tag %>
</head>
<body>

<div class='flash'><%= flash[:message] %></div>

<%= yield %>

</body>
</html>
```

gaslightsoftware



app/controllers/topics_controller.rb

```
class TopicsController < ApplicationController
  # ... other actions ...

  def vote
    render :nothing => true
  end
end
```

Remember this? The vote action still doesn’t *do* anything. It doesn’t even send back a page. Let’s fix that!

app/controllers/topics_controller.rb

```
class TopicsController < ApplicationController
  # ... other actions ...

  def vote
    redirect_to topics_path
  end
end
```

gaslightsoftware

rake test:integration

```
Started
.....F.
Finished in 1.903812 seconds.

  1) Failure:
test: visiting the list of suggestions with a
suggestion voting for the topic should have a
message in flash. (VotingTest) [test/
integration/voting_test.rb:23]:
Not even a thank you?
```

gaslightsoftware

app/controllers/topics_controller.rb

```
class TopicsController < ApplicationController
  # ... other actions ...

  def vote
    redirect_to topics_path
  end
end
```

app/controllers/topics_controller.rb

```
class TopicsController < ApplicationController
  # ... other actions ...

  def vote
    flash[:message] = "Thanks for voting"
    redirect_to topics_path
  end
end
```

rake test:integration

```
Started
.......
Finished in 2.281981 seconds.

7 tests, 9 assertions, 0 failures, 0 errors, 0 skips
```

gaslightsoftware

Rails in a Day

- Save your work!
  - git status
  - git add .
  - git commit -m “Thanks for a vote”

Rails in a Day

- Enter a few new topics into your own copy

- Start your application server with:

```
rails server
```

or just

```
rails s
```

gaslightsoftware

Rails console

- rails c
- interactive in environment for playing with rails
-
  ```
  Topic.create(
   :title => “My new topic”,
   :description => “is a very nice topic”)
  ```

Rails in a Day

“As a person who voted, I’d like to have my vote recorded for posterity.”

```
$ git cherry-pick dor_test_3
```

gaslightsoftware

Well, let’s get a test for that!

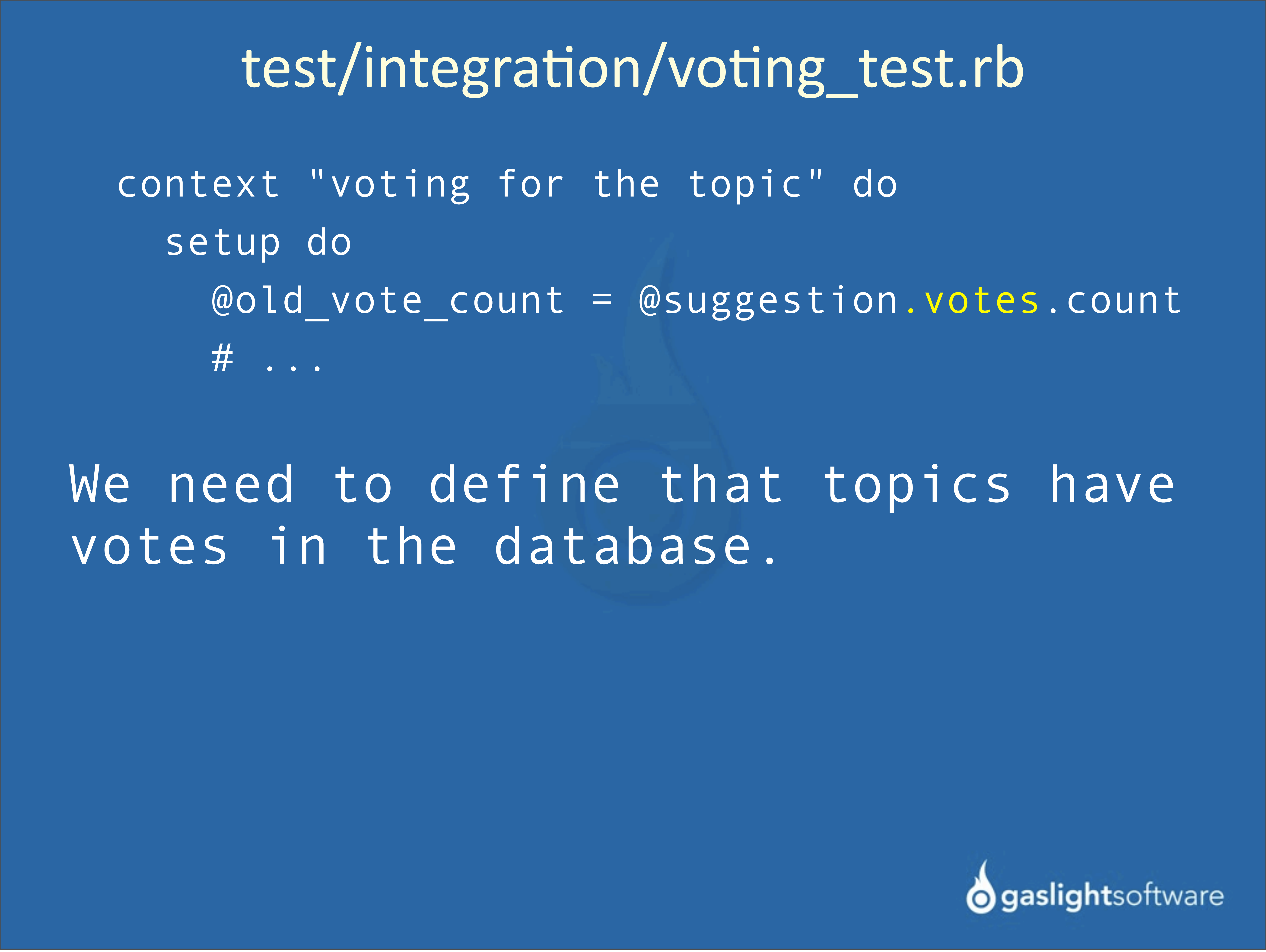
test/integration/voting_test.rb

```
class VotingTest < ActionDispatch::IntegrationTest
  # ...
    context "voting for the topic" do
      setup do
        @old_vote_count = @suggestion.votes.count
        within(:css, "#topic_#{@suggestion.id}") do
          click "Vote"
        end
      end
      # ...
      should "add a vote for this topic" do
        assert_equal @old_vote_count + 1, @suggestion.votes(true).count, "Where's the vote? "
      end
    end
  # ...
end
```

rake test:integration

```
Started
.....EEE
Finished in 1.866956 seconds.

  1) Error:
test: visiting the list of suggestions with a
suggestion voting for the topic should add a
vote for this topic. (VotingTest):
NoMethodError: undefined method `votes' for
#<Topic:0x21dfd9c>
```



test/integration/voting_test.rb

```
context "voting for the topic" do
  setup do
    @old_vote_count = @suggestion.votes.count
    # ...
```

We need to define that topics have votes in the database.

Because we’ve added this code to the setup block, even some of the test that were recently passing now have an error because the setup isn’t valid.

Rails in a Day

- A vote belongs to a topic
- A topic can have many votes
- ActiveRecord makes these associations easy!

Associations

- One to many
  - specified with belongs_to
  - convention is a related_to_id column
- Many to one
  - specified with has_many
  - assumes “foreign key” column is in the other table
- Does not use real FKs by default
- Others: has_one, has_and_belongs_to_many, etc

gaslightsoftware

Rails in a Day

- First we need a vote model and we’ll let a “generator” help create the files that we need.

```
rails generate model
```

will give a helpful summary of the options.

gaslightsoftware

rails generate model Vote …

```
$ rails generate model Vote topic:belongs_to --no-fixture
invoke  active_record
      create    db/migrate/20100905173226_create_votes.rb
      create    app/models/vote.rb
      invoke    test_unit
      create      test/unit/vote_test.rb
```

rails generate model Vote ...

```
$ rails generate model Vote topic:belongs_to --no-fixture
invoke  active_record
      create    db/migrate/20100905173226_create_votes.rb
      create    app/models/vote.rb
      invoke    test_unit
      create      test/unit/vote_test.rb
```

gaslightsoftware

Migrations

- Incremental changes to the schema of your database
- Contains code to apply “up” or forward migration changes as well as “down” or “revert”
- Are time-stamped so that they can be created and applied by different developers
- Are database agnostic! The same migration can be used against your local development and test databases as the production database.

gaslightsoftware

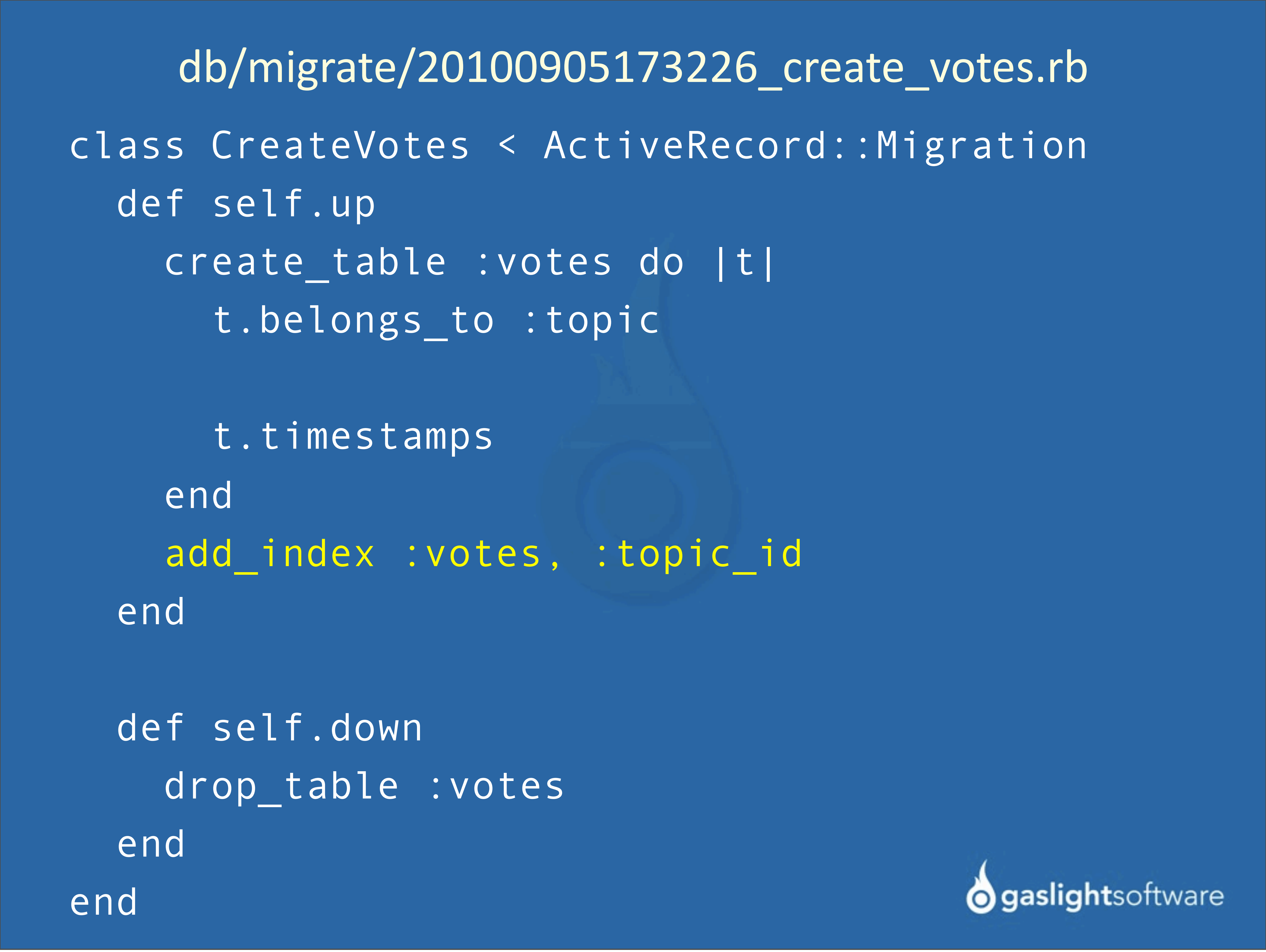
db/migrate/20100905173226_create_votes.rb

```
class CreateVotes < ActiveRecord::Migration
  def self.up
    create_table :votes do |t|
      t.belongs_to :topic

      t.timestamps
    end
  end

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

gaslightsoftware



db/migrate/20100905173226_create_votes.rb

```
class CreateVotes < ActiveRecord::Migration
  def self.up
    create_table :votes do |t|
      t.belongs_to :topic

      t.timestamps
    end
    add_index :votes, :topic_id
  end

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

gaslightsoftware

Foreign key? Discuss the index.
WARNING: Pronounce the punctuation:
“add(underscore)index (space) (colon)votes(comma) (space) (colon)topic(underscore)id”

rails generate model Vote …

```
$ rails generate model Vote topic:belongs_to --no-fixture
invoke  active_record
      create    db/migrate/20100905173226_create_votes.rb
      create    app/models/vote.rb
      invoke    test_unit
      create      test/unit/vote_test.rb
```

gaslightsoftware

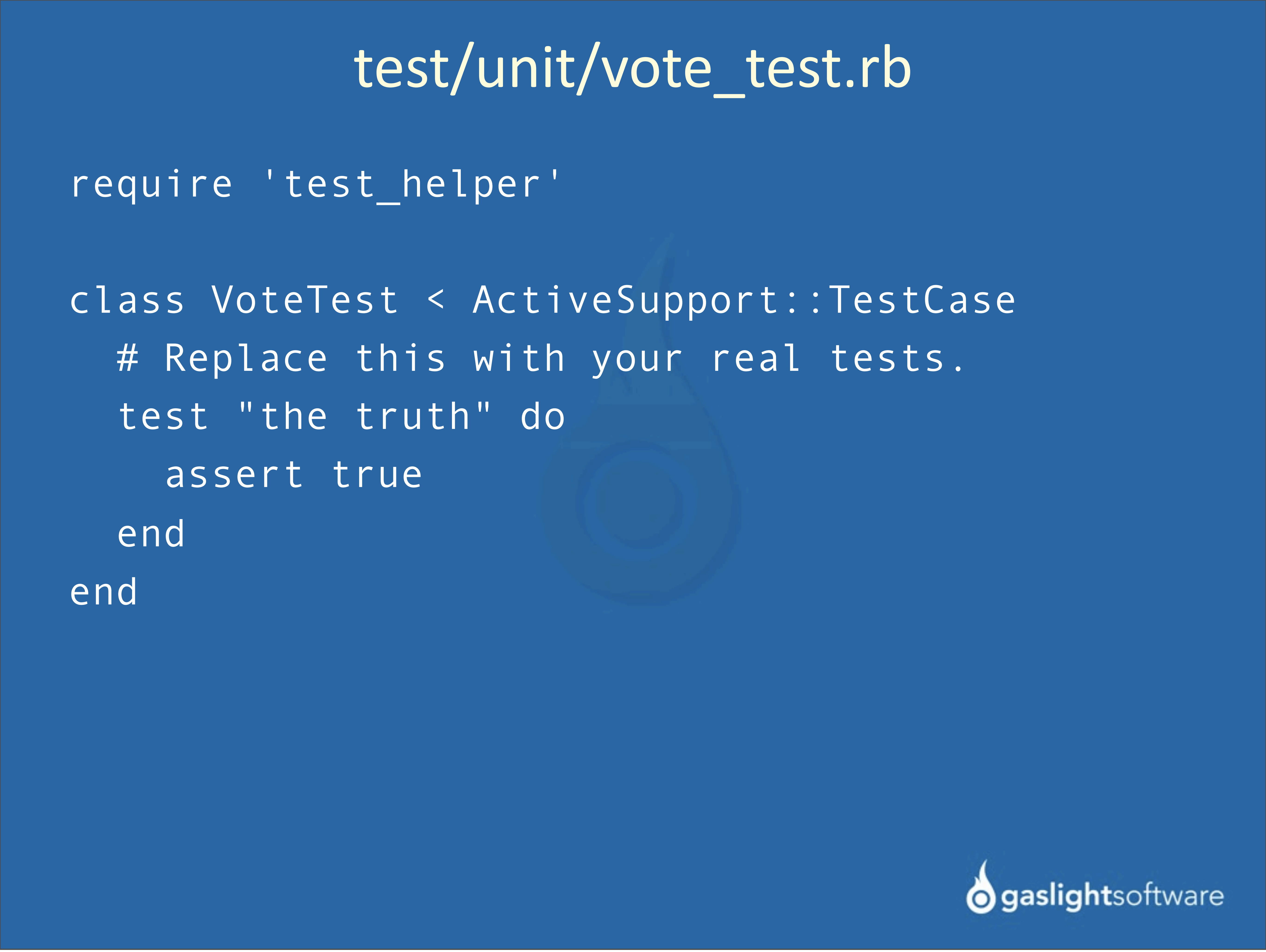
app/models/vote.rb

```
class Vote < ActiveRecord::Base
  belongs_to :topic
end
```

gaslightsoftware

rails generate model Vote ...

```
$ rails generate model Vote topic:belongs_to --no-fixture
invoke  active_record
      create    db/migrate/20100905173226_create_votes.rb
      create    app/models/vote.rb
      invoke    test_unit
      create      test/unit/vote_test.rb
```



test/unit/vote_test.rb

```
require 'test_helper'

class VoteTest < ActiveSupport::TestCase
  # Replace this with your real tests.
  test "the truth" do
    assert true
  end
end
```

Of course, this doesn’t really test anything, but our Vote model doesn’t really *do* anything worth testing yet.

rake test

```
$ rake test
(in /Users/rab/projects/Suggestion-Box)
You have 1 pending migrations:
  20100905173226 CreateVotes
Run "rake db:migrate" to update your database
then try again.
```

rake db:migrate

```
==  CreateVotes: migrating ====================
-- create_table(:votes)
   -> 0.0035s
-- add_index(:votes, :topic_id)
   -> 0.0015s
==  CreateVotes: migrated (0.0055s) ===========
```

rake test

```
Started
.....EEE
Finished in 1.951733 seconds.

  1) Error:
test: visiting the list of suggestions with a
suggestion voting for the topic should add a
vote for this topic. (VotingTest):
NoMethodError: undefined method `votes' for
#<Topic:0x1c49478>
```

gaslightsoftware

Wait! Didn’t we fix this?
Nope. A vote knows that it belongs to a topic, but a topic doesn’t yet know that it has votes.

Rails in a Day

- Save your work!
  - git status
  - git add .
  - git commit -m “Made a vote model”

gaslightsoftware

Rails in a Day

“As a rails developer, I want to capture my model associations between Topic and Vote.”

```
$ git cherry-pick dor_test_3a
```

gaslightsoftware

Well, let’s get a test for that!

Associations Explained

gaslightsoftware

test/unit/topic_test.rb

```
require 'test_helper'

class TopicTest < ActiveSupport::TestCase
  should have_many(:votes).dependent(:delete_all)
end
```

test/unit/vote_test.rb

```
require 'test_helper'

class VoteTest < ActiveSupport::TestCase
  # Replace this with your real tests.
  test "the truth" do
    assert true
  end
end
```

gaslightsoftware

test/unit/vote_test.rb

```ruby
require 'test_helper'

class VoteTest < ActiveSupport::TestCase
  should belong_to(:topic)
end
```

gaslightsoftware

rake test:units

```
Started
F.
Finished in 0.091453 seconds.

  1) Failure:
test: Topic should have many votes dependent => delete_all. (TopicTest) [/Users/rab/.rvm/gems/ruby-1.9.2-p0@suggestion_box/gems/shoulda-2.11.3/lib/shoulda/context.rb:324]:
Expected Topic to have a has_many association called votes (no association called votes).
Expected block to return true value.
```

app/models/topic.rb

```
class Topic < ActiveRecord::Base
end
```

gaslightsoftware

app/models/topic.rb

```
class Topic < ActiveRecord::Base
  has_many :votes, :dependent => :delete_all
end
```

rake test:units

```
Started
....
Finished in 0.357875 seconds.

4 tests, 4 assertions, 0 failures, 0 errors, 0 skips
```

gaslightsoftware

rake test:integration

```
Started
.....F..
Finished in 2.038847 seconds.

  1) Failure:
test: visiting the list of suggestions with a
suggestion voting for the topic should add a
vote for this topic. (VotingTest) [test/
integration/voting_test.rb:31]:
Where's the vote? .
<1> expected but was
<0>.
```

app/controllers/topics_controller.rb

```
class TopicsController < ApplicationController
  # ...
  def vote
    if (@topic=Topic.find_by_id(params[:id])) &&
      @topic.votes.create
      flash[:message] = "Thanks for voting"
    else
      flash[:message] ="Sorry, couldn't record your vote"
    end
    redirect_to topics_path
  end
end
```

gaslightsoftware

rake test:integration

```
Started
........
Finished in 1.154301 seconds.

8 tests, 10 assertions, 0 failures, 0 errors, 0 skips
```

Rails in a Day

- Save your work!
  - git status
  - git add .
  - git commit -m “create vote in database”

Rails in a Day

- Vote a few times for each of the topics into your own copy of the application

- If it isn't running, start your application server with:

```
rails server
```

Rails in a Day

“As a curious person who voted, I want to see the tally of votes accumulated by each topic.”

```
$ git cherry-pick dor_test_4
```

Well, let’s get a test for that!

test/unit/topic_test.rb

```
class TopicTest < ActiveSupport::TestCase
  context "a topic" do
    setup do
      @topic = Factory(:topic)
    end
    should "have a zero tally" do
      assert_equal 0, @topic.tally
    end
    # …
```

gaslightsoftware

test/unit/topic_test.rb

```ruby
    # …
    context "with a vote" do
      setup do
        @topic.votes.create
        @topic.reload
      end
      should "have a tally of 1" do
        assert_equal 1, @topic.tally
      end
    end
  end
end
```

Why the reload?

rake test:units

```
Started
.EE.
Finished in 0.226788 seconds.

  1) Error:
test: a topic should have a zero tally.
(TopicTest):
NoMethodError: undefined method `tally' for
#<Topic:0x1984df0>
```

gaslightsoftware

app/models/topic.rb

```
class Topic < ActiveRecord::Base
  has_many :votes, :dependent => :delete_all

  def tally
    votes.count
  end
end
```

rake test:units

```
Started
....
Finished in 0.229227 seconds.

4 tests, 4 assertions, 0 failures, 0 errors, 0 skips
```

gaslightsoftware

Rails in a Day

- Save your work!
  - git status
  - git add .
  - git commit -m “add Topic#tally”

gaslightsoftware

Rails in a Day

- You can run up some votes, but you might notice that you can't *see* the tally anywhere.

- If it isn't running, start your application server with:

```
rails server
```

gaslightsoftware

Rails in a Day

```
“As a curious person who voted, I want to
REALLY see the tally of votes accumulated by
each topic.”

$ git cherry-pick dor_test_5
```

Well, let’s get a test for that!

test/integration/voting_test.rb

```
# A little refactoring, too.
should "have a message in flash" do
  assert page.has_css?(".flash", :text => "Thanks"), \
    "Not even a thank you? "
end

should "show the vote tally" do
  assert page.has_css?("#topic_#{@suggestion.id} .tally", \
    :text => (@old_vote_count + 1).to_s), \
    "No tally given"
end
```

gaslightsoftware

rake test:integration

```
Started
........F
Finished in 1.323555 seconds.

  1) Failure:
test: visiting the list of suggestions with a
suggestion voting for the topic should show the
vote tally. (VotingTest) [test/integration/
voting_test.rb:32]:
No tally given
```

app/views/topics/_topic.html.erb

```
Initially:
  <li id='<%= dom_id(topic) %>'>
    <%= link_to topic.title, topic %>
    <%= link_to "Vote", vote_topic_path(topic) %></li>

After adding the link:
  <li id='<%= dom_id(topic) %>'>
    <%= link_to topic.title, topic %>
    <%= link_to "Vote", vote_topic_path(topic) %>
    <span class="tally"><%= topic.tally %></span></li>
```

gaslightsoftware

Refresh your browser to see the new link!

rake test:integration

```
Started
.........
Finished in 1.269294 seconds.

9 tests, 11 assertions, 0 failures, 0 errors, 0 skips
```

Rails in a Day

- Save your work!
  - –git status
  - –git add .
  - –git commit -m “show the tally”

gaslightsoftware

Rails in a Day

- Now that you can **see** the tally of your votes, what else might you want to do?

- If it isn't running, start your application server with:

```
rails server
```

gaslightsoftware

Rails in a Day

“The topics list looks bad as a list when the titles are of different lengths.”

Change the topics list view to use a table.

No tests need to be added.

All existing tests should continue to pass.

gaslightsoftware

app/views/topics/index.html.erb

```
<h2>Suggestion Box</h2>

<p>List of topics submitted so far:</p>

<% if @topics.blank? %>
  <p>No topics submitted so far. Do you have
any suggestions?</p>
<% else %>
  <ul>
    <%= render @topics %>
  </ul>
<% end %>
```

app/views/topics/_topic.html.erb

```
<li id='<%= dom_id(topic) %>'>
  <%= link_to topic.title, topic %>
  <%= link_to "Vote", vote_topic_path(topic) %>
  <span class="tally"><%= topic.tally %></span></li>
```

Rails in a Day

- Save your work!
  - git status
  - git add .
  - git commit -m “replace ul with table”

Rails in a Day

“As a meeting organizer, I’d like to see the list of topics sorted by descending vote tally.”

```
$ git cherry-pick dor_test_6
```

gaslightsoftware

test/integration/voting_test.rb

```
context "sorting topics" do
  setup do
    @most = Factory(:topic)
    5.times { @most.votes.create }
    @least = Factory(:topic)
    3.times { @least.votes.create }
    @none = Factory(:topic)
    visit topics_path
  end
```

gaslightsoftware

test/integration/voting_test.rb

```
should "have a sort by tally link" do
  sort_by_tally_link = \
    topic_path(:sort => :tally)
  assert(page.has_css?("a[href='#{sort_by_tally_link}']",
         :text => 'Vote Tally'),
    "Where's the link to #{sort_by_tally_link}? (with
text 'Vote Tally')")
end
```

test/integration/voting_test.rb

```
  context "by vote tally" do
    setup do
      click "Vote Tally"
    end
    should "have the most suggested topic first" do
      selector = "tbody tr:first-child td.tally"
      assert page.has_css?(selector),
        "Can't find the element to match: #{selector}"
      assert page.has_css?(selector,
                           :text => @most.tally.to_s),
        "Topic with most votes isn't listed first"
    end
  end
end
```

rake test:integration

```
Started
.....EF....
Finished in 1.293699 seconds.

  1) Error:
test: visiting the list of suggestions with a suggestion sorting
topics by vote tally should have the most suggested topic first.
(VotingTest):
Capybara::ElementNotFound: no link or button 'Vote Tally' found

  2) Failure:
test: visiting the list of suggestions with a suggestion sorting
topics should have a sort by tally link. (VotingTest) [test/
integration/voting_test.rb:47]:
Where's the link to /topics?sort=tally? (with text 'Vote Tally')
```

app/views/topics/index.html.erb

```
<thead>
  <tr>
    <th>Topic</th>
    <th></th>
    <th>
      <%= link_to "Vote Tally",
            topics_path(:sort => :tally) %>
    </th>
  </tr>
</thead>
```

rake test:integration

```
Started
.....F.....
Finished in 1.305399 seconds.

  1) Failure:
test: visiting the list of suggestions with a
suggestion sorting topics by vote tally should
have the most suggested topic first. (VotingTest)
[test/integration/voting_test.rb:57]:
Topic with most votes isn't listed first
```

app/controllers/topics_controller.rb

```
def index
  @topics = Topic.all
end
```

app/controllers/topics_controller.rb

```
def index
  @topics = Topic.all
  if params[:sort]
    @topics = @topics.sort_by { |topic|
      topic.send(params[:sort])
    }.reverse
  end
end
```

rake test:integration

```
Started
...........
Finished in 1.314155 seconds.

11 tests, 14 assertions, 0 failures, 0 errors, 0 skips
```

Rails in a Day

- Save your work!
  - –git status
  - –git add .
  - –git commit -m “sort by tally”

gaslightsoftware

Rails in a Day

- Great! Now you can vote up your favorite topics and see them sort to the top of the table!

- If it isn't running, start your application server with:

```
rails server
```

gaslightsoftware

Rails in a day (bonus round!)

“As a submitter, I want a helpful error message to show up if I forget to fill in the title.”

```
$ git cherry-pick dor_test_7
```

test/integration/creating_topics_test.rb

```
...

  context "without a title" do
    setup do
      visit new_topic_path
      click_button "Create Topic"
    end
    should "see an error message" do
      assert page.has_content?("Title can't be blank"), "The page has no errors"
    end
  end
```

rake test:integration

```
Started
....F.......
Finished in 0.995729 seconds.

  1) Failure:
test: without a title should see an error
message. (CreatingTopicsTest) [test/
integration/creating_topics_test.rb:47]:
The page has no errors

12 tests, 15 assertions, 1 failures, 0 errors,
0 skips
```

gaslightsoftware

Validation in Rails

- Specified in the model
- Bunches built in
- Easy to roll your own
- Messages are based on column name
  - can be overridden

gaslightsoftware

test/units/topic_test.rb

```
require 'test_helper'

class TopicTest < ActiveSupport::TestCase
  should have_many(:votes).dependent(:delete_all)

  should_validate_presence_of :title

...
```

rake test:units

```
Started
.F...
Finished in 0.162058 seconds.

  1) Failure:
test: Topic should require title to be set. (TopicTest) [/Users/superchris/.rvm/gems/ruby-1.9.2-p0/gems/shoulda-2.11.3/lib/shoulda/context.rb:324]:
Expected errors to include "can't be blank" when title is set to nil, got no errors.
Expected block to return true value.
```

gaslightsoftware

app/models/topic.rb

```
class Topic < ActiveRecord::Base
  has_many :votes, :dependent => :delete_all

  validates_presence_of :title

  def tally
    votes.count
  end
end
```

rake test:units

```
Started
.....
Finished in 0.159702 seconds.

5 tests, 5 assertions, 0 failures, 0 errors, 0 skips
```

app/views/topics/new.html.erb

```
<h1>New Topic</h1>

<ul>
  <% @topic.errors.full_messages.each do |error| %>
      <li><%= error %></li>
  <% end %>
</ul>
```

gaslightsoftware

app/controllers/topics_controller.rb

```ruby
def create
  # Fix that bug!
  @topic = Topic.new(params[:topic])
  if @topic.save
    redirect_to topics_path
  else
    render :new
  end
end
```

rake test:integration

```
Started
............
Finished in 1.039139 seconds.


12 tests, 15 assertions, 0 failures, 0 errors,
0 skips
```

To learn more:

- rubyonrails.org
  - http://rubyonrails.org/documentation
- railscasts.com
  - by Ryan Bates
    - rails-3.0 episodes
- Agile Web Development with Rails, 4th Edition
  - by Sam Ruby
  - Pragmatic Programmers

The Pragmatic Programmers

Agile Web Development with Rails

Fourth Edition

Sam Ruby
Dave Thomas
David Heinemeier Hansson

BETA

gaslightsoftware

To learn EVEN MORE

- JRubyConf!!
- Columbus, OH Oct 1-3
- Save $100 with `DAYOFRUBY` discount code
  - –You’re welcome :)

Any Questions?

Speakers will be hanging out at the

Brazenhead Irish Pub

immediately following the

Cincinnati Day of Ruby

gaslightsoftware