

Pagination

will_paginate gem will help you with pagination

gem 'will_paginate', '~> 3.3'

Basic will_paginate use

```
## perform a paginated query:  
@posts = Post.paginate(page: params[:page])  
  
# or, use an explicit "per page" limit:  
Post.paginate(page: params[:page], per_page: 30)  
  
## render page links in the view:  
<%= will_paginate @posts %>
```

Estilos

http://mislav.github.io/will_paginate/

152. Add login form

usaremos sesiones para simular que un usuario esta logueado..

Crearemos un controlador nuevo para manejar esto.

```
resources :users, except: [:new]  
  get 'login', to: 'sessions#new'  
  post 'login', to: 'session#create'  
  delete 'logout', to: 'session#destroy'  
end  
  
iv class="col-10">  
<%= form_with(scope: :session, class: "shadow p-3 mb-3 bg-info rounded", local: true)>  
  <div class="form-group row">
```

154. Create and destroy user sessions

Mensajes flash

Flas sin now cuando haremos un redirect

```
def update
  if @article.update(article_params)
    flash[:notice] = "Article was updated successfully."
    redirect_to @article
  else
```

Flash now cuando no redirigimos

```
  render 'new'
  flash.now[:alert] = "There was something wrong with your login details"
```

Cuando no usamos now, flash persiste en la siguiente petición http más.

Si no lo hubiéramos usado el new.

Sessions

Las sesiones permiten tener un estado del usuario. En este caso seria si el usuario fue autenticado correctamente.

```
if user && user.authenticate(params[:ses
  session[:user_id] = user.id
end
```

Esto hara que se recuerde el id cada vez que el usuario hace una petición.

Lo que hacen las sesiones es almacenar un id para cada sesión.

5 Session

Your application has a session for each user in which you can store small amounts of data that will be persisted between requests. The session is only available in the controller and the view and can use one of a number of different storage mechanisms:

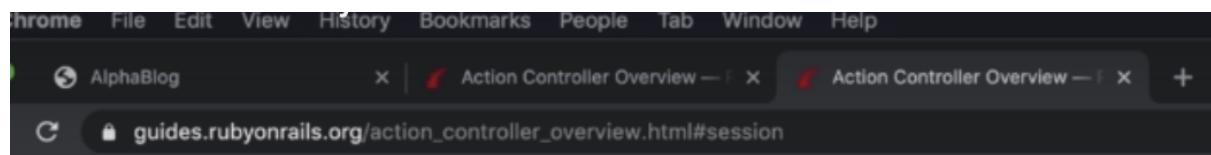
- ActionDispatch::Session::CookieStore - Stores everything on the client.
- ActionDispatch::Session::CacheStore - Stores the data in the Rails cache.
- ActionDispatch::Session::ActiveRecordStore - Stores the data in a database using Active Record. (require activerecord-session_store gem).
- ActionDispatch::Session::MemCacheStore - Stores the data in a memcached cluster (this is a legacy implementation; consider using CacheStore instead).

All session stores use a cookie to store a unique ID for each session (you must use a cookie, Rails will not allow you to pass the session ID in the URL as this is less secure).

Nosotros usaremos la primera opción que almacena toda la info en la misma cookie.

Esta sesión está encriptada en el lado del cliente y firmada para que nadie pueda modificarlo

Mas info en:



* ActionDispatch::Session::CookieStore - Stores everything on the client.

Para quitar la sesion simplemente

```
ef destroy
  session[:user_id] = nil
```

More info

<https://www.justinweiss.com/articles/how-rails-sessions-work/>

156. Authentication helper methods

Memorización

Vamos a usar un método para obtener el user con el id de la session.

```
def current_user
  User.find(session[:user_id]) if session[:user_id]
end
```

El problema es que esto es una llamada a la bd a cada rato. Así que podemos usar algo llamado **memorización**

Eso es que si ya hemos hecho una referencia a ese usuario en el pasado, pues queremos regresar ese usuario en lugar de hacer otra llamada.

gira alrededor de @current_user, no se explica mas a fondo en el video

Yes, `current_user` uses `session`. You can do something similar in your application controller if you want to roll your own authentication:

```
) def current_user
  return unless session[:user_id]
  @current_user ||= User.find(session[:user_id])
end
```

```
def current_user
  user_id = session[:user_id]
  if user_id
    @current_user ||= User.find(user_id)
  end
end
```

Checar si esta logueado

```
def logged_in?
  return !!@current_user
end
```

Checamos esa variable @current_user
El !! convierte lo de la derecha a booleano

Double bang operator

In Ruby (and many other languages) there are many values that evaluate to `true` in a boolean context, and a handful that will evaluate to false. In Ruby, [the only two things that evaluate to `false` are `false` \(itself\) and `nil`.](#)

If you negate something, that forces a boolean context. Of course, it also negates it. If you double-negate it, it forces the boolean context, but returns the proper boolean value.

For example:

```
"hello"  #-> this is a string; it is not in a boolean context
!"hello" #-> this is a string that is forced into a boolean
          #   context (true), and then negated (false)
!!"hello" #-> this is a string that is forced into a boolean
           #   context (true), and then negated (false), and then
           #   negated again (true)
!!nil     #-> this is a false-y value that is forced into a boolean
           #   context (false), and then negated (true), and then
           #   negated again (false)
```

158. Controller methods as helper methods

Learn how to use the methods of the controllers

NO podemos usar metodos del helper en el controlador

Podemos mover el método de get_user hacia la clase Application controller para que todos los controladores lo tengan. El problema es que no funcionara en nuestras vistas.

Para solucionarlo hay que decirle que también sea un método helper.

AHORA PODEMOS OBTENER EL USER CON EL MISMO CÓDIGO EN EL CONTROLADOR Y VISTA

Ahora queremos que como facebook y otras apps, que si el usuario está logueado, se redireccione a otra página al tratar de ir a la home page.

HEMOS APRENDIDO A USAR MÉTODOS QUE FUNCIONEN COMO HELPER METHODS EN NUESTRAS VISTAS Y EN NUESTROS CONTROLADORES

- ▶ If it's "something all controllers use", I would place it in a base class used by all controllers. The "application_controller" comes to mind.
↓
- ▶ Share Improve this answer Follow answered Jul 24, 2009 at 15:09
Pete 9,940 ● 7 ● 52 ● 58
- ▶
- ...

160. Restrict actions from UI

Pusimos restricciones en cuanto a botones, pero aun así el usuario puede saltarse estas restricciones e ir a la página a través de la url.

Summary of changes:

- Restricted the new article creation option in the navigation menu to logged in users only.
- Added a dropdown with users profile action links.

164. Restrict actions at controller level - articles

Bloquear acceso a una página

Pones un método para verificar si el user esta logueado en la clase en que todos los controles heredan y poner el sig before_Action

```
> controllers > articles_controller.rb
  class ArticlesController < ApplicationController

    before_action :set_article, only: [:show, :edit, :update, :destroy]
    before_action :require_user, except: [:show, :index]

  def require_user
    if !logged_in?
      flash[:alert] = "You must be logged in to perform that action"
      redirect_to login_path
    end
  end
```

Tambien en otras

```
1  class ArticlesController < ApplicationController
2
3    before_action :set_article, only: [:show, :edit, :update, :destroy]
4    before_action :require_user, except: [:show, :index]
5    before_action :require_same_user, only: [:edit, :update, :destroy]
6
```

El require_same_user definirlo despues que require user

168. Delete user

Eliminar artículos al eliminar usuario

```
class User < ApplicationRecord
  before_save { self.email = email.downcase }
  has_many :articles, dependent: :destroy
  validates :username, presence: true,
    uniqueness: { case_sensitive: false },
    length: { minimum: 3, maximum: 25 }
  VALID_EMAIL_REGEX = /\A[\w+\-\.]+@[a-z\d\-.]+\.[a-z]+\z/i
  validates :email, presence: true,
    uniqueness: { case_sensitive: false },
    length: { maximum: 105 },
    format: { with: VALID_EMAIL_REGEX }
  has_secure_password
end
```

Agregar en el modelo el dependent: destroy

170. Add admin user functionality - intro

Permissions functionality

Admin users, moderators, regular users etc.

CRM apps - permissions table

Permissions field (String) - with
values such as admin, moderator

Admin column (boolean) -
simplest solution

Varias maneras de manejar los permisos

```
db > migrate > 20220816003657_add_admin_field_on_users.rb
1   class AddAdminFieldOnUsers < ActiveRecord::Migration[6.1]
2     def change
3       add_column :users, :admin, :boolean, default: false
4     end
5   end
6
```

Cambiar usuario hacia admin

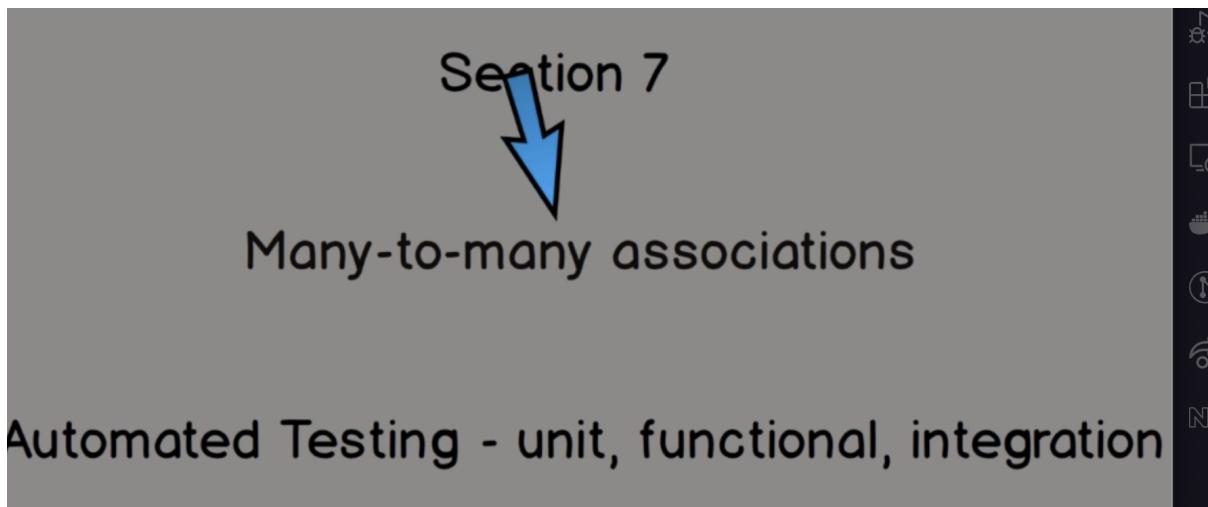
```
development      , created_at: "2020-04-03 09:23:02" , updated_at: "2020-04-03 09:23:02" , password_digest: [FILTERED], admin: false>
schema.rb
seeds.rb
2.6.3 :005 > user.toggle!(:admin)
t.datetime "updated_at", precision: 6
```

```
b > migrate > 20220816003657_add_admin_field_on_users.rb
1   class AddAdminFieldOnUsers < ActiveRecord::Migration[6.1]
2     def change
3       add_column :users, :admin, :boolean, default: false
4     end
5   end
6
```

176. Production deploy and wrap up section 6

```
> Warning: heroku update
$ heroku run rails console
```

177. Introduction to Section 7



Unit, functional and integration test

Unit Tests: Models, individual units of the application (like a validation) are working

Functional Tests: Controllers, a function is working, for example is before_action stopping a non-logged in user from performing an action

Integration Tests: Full features, start to finish of a business process, example: a user signs up for the app

178. Category model and testing

active record assertions

All News Images Shopping Videos More Settings Tools

About 11.000.000 results (0,44 seconds)

Did you mean: **activerecord** assertions

guides.rubyonrails.org › testing ▾

[Testing Rails Applications — Ruby on Rails Guides](#)

The default rails library for testing is mini-test, tienen assertions

HAREMOS TESTING DE MODELOS

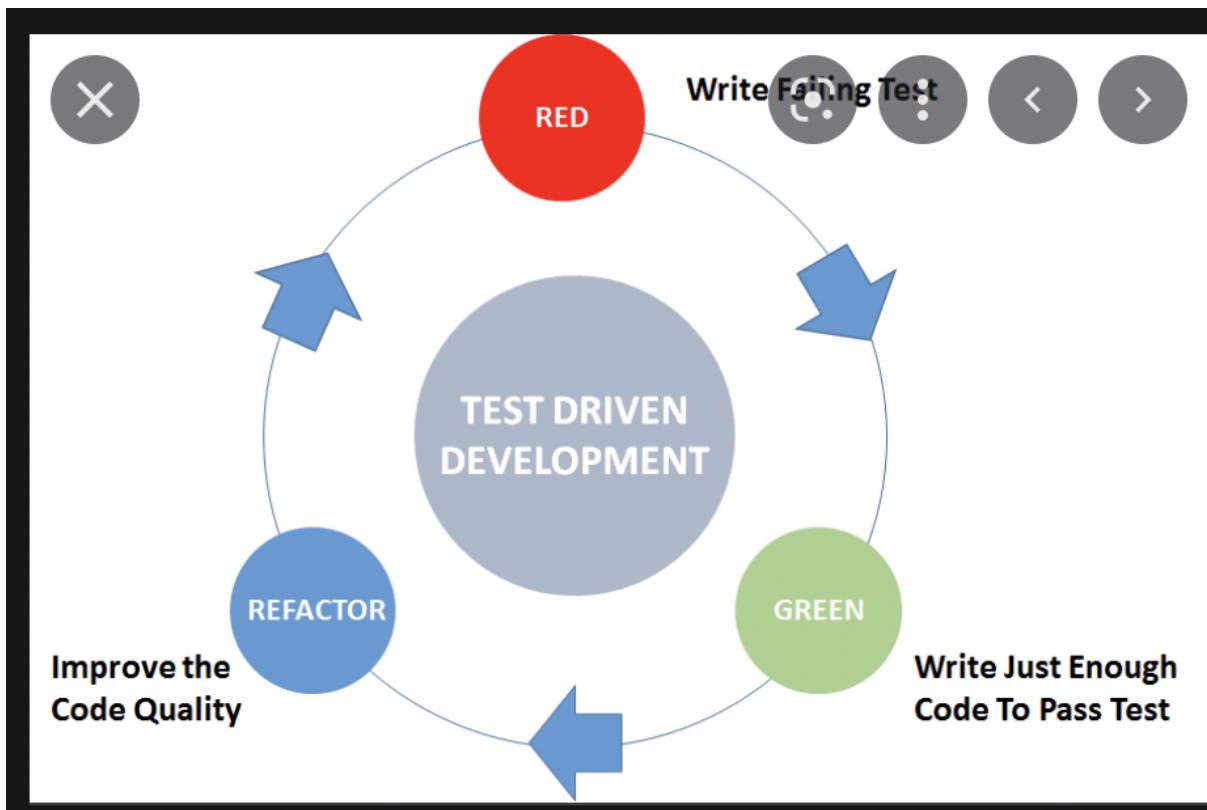
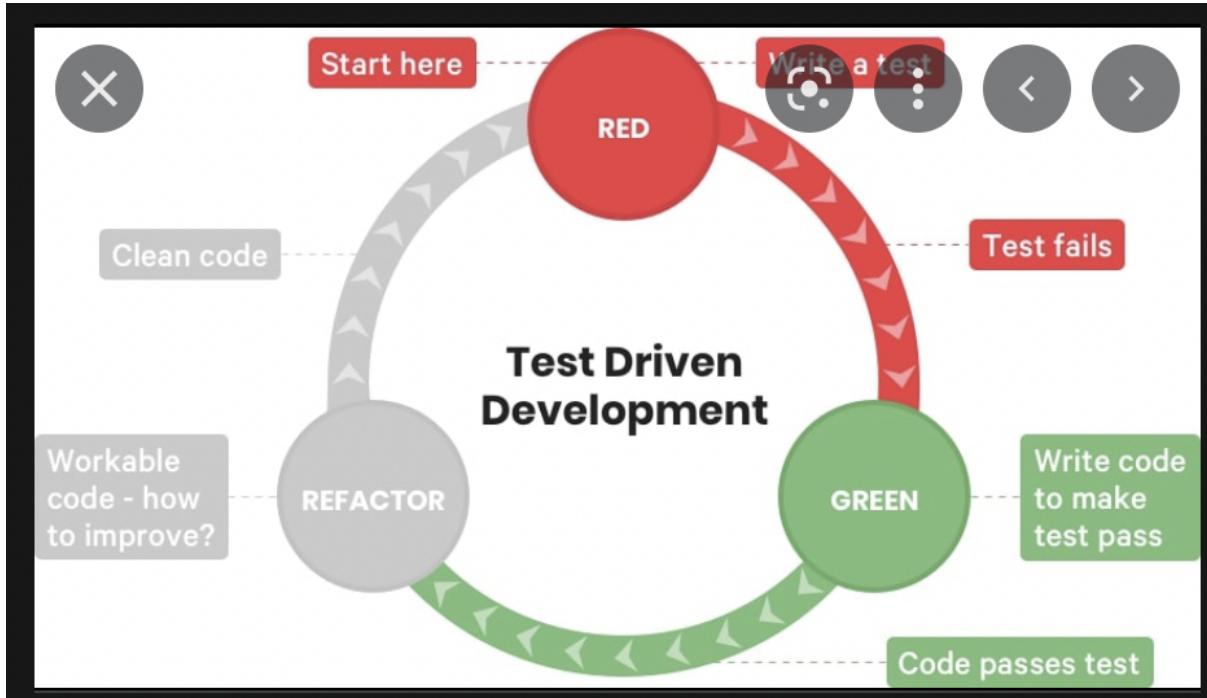
Como queremos hacer un test de modelo de la categoria creamos category_test.rb

```
st > models > category_test.rb
1   require 'test_helper'
2
3   class CategoryTest < ActiveSupport::TestCase
4   end
5
```

Para correrlo usamos rails test (corre todos los tests)

Test driven development

Escribiremos la cantidad de código suficiente para hacer primero el test.





Different types of testing

Sarah · Clase 177 · hace 4 años

0 ⬆ :

My company does TDD and other testing so I've been spending a lot of time studying testing and working with senior engineers on my team on it, and I think this definition of functional tests is confusing. The Rails documentation says this about functional tests: "When writing functional tests, you are testing how your actions handle the requests and the expected result or response, in some cases an HTML view."

1 respuesta

Seguir las respuestas

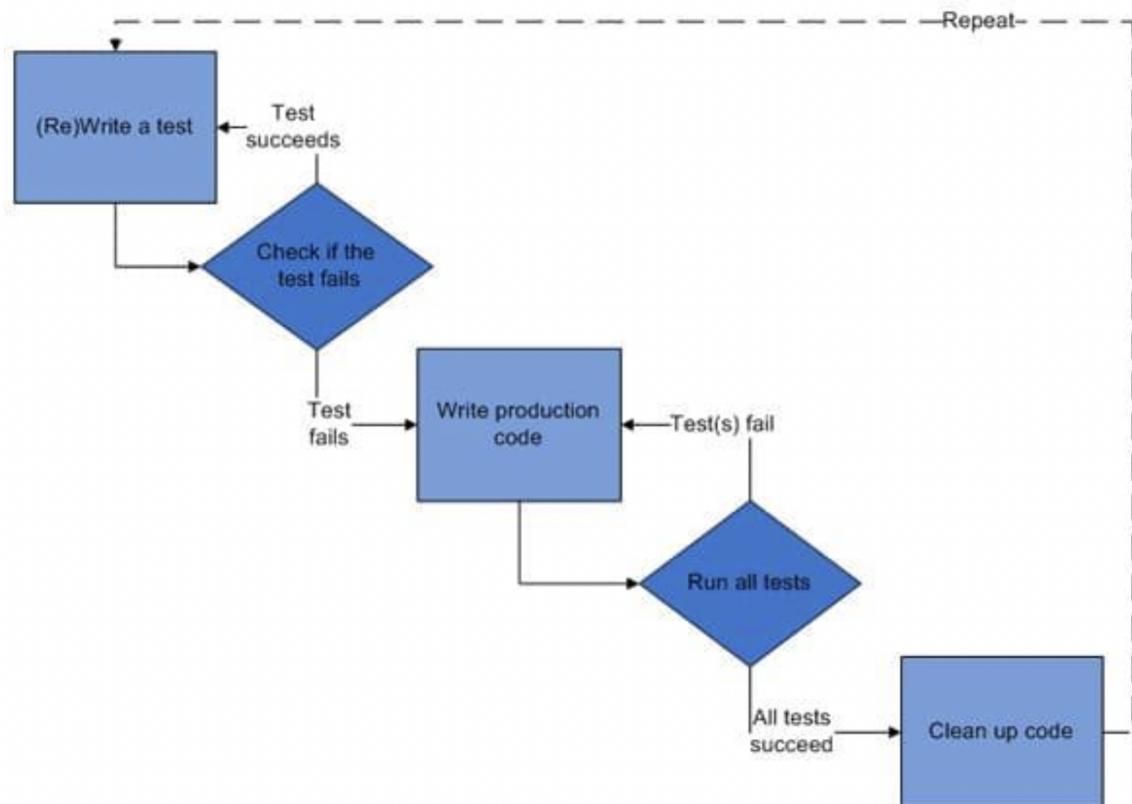


Sasikala — Profesor asistente

hace 4 años

2 ⬆ :

Hi Sarah, What Mashrur says is pretty much same as what is in Rails documentation. It means when you say "[you are testing how your actions handle the requests](#)", it is nothing but controller actions of the app which handles the requests and "[the expected result or response, in some cases an HTML view](#)" means that the controller renders the view as response in case of index or show actions. Also, Mashrur was mentioning that in case of using before_action callback, we can even check whether controller action has been using the callback or not. Hope this helps you to make the connection between the two definitions.



Queremos probar que la categoría sea válida

1. Escribimos el test
2. El test falla porque no existe la categoría
3. Escribimos la cantidad mínima para hacer que esa prueba funcione (creamos el modelo categoría)
4. Probamos otra vez
 - a. Error, nos falta la tabla categoria, así que la agregamos
 - b. Esto se repite hasta el éxito
5. Probamos otra vez y da éxito

Los tests corren en una base de datos de testeo. Cada vez que corres un test la bd es limpiada.

Esto es un test automatizado

```
test > models > category_test.rb
1   require 'test_helper'
2
3   class CategoryTest < ActiveSupport::TestCase
4
5       test "category should be valid" do
6           @category = Category.new(name: "Sports")
7           assert @category.valid?
8       end
9
10  end
11
```

segun esto en algunas versiones daba error el rails test porque una gema no se agregaba por default

Por si acaso nosotros la pondremos

```
gem 'rexml', require: false
```

180. Validations using unit tests

En cada test, los objetos y variables se van a limpiar, asi que @category no existira en el proximo test

```
test "category should be valid" do
    @category = Category.new(name: "Sports")
    assert @category.valid?
end

test "name should be present" do
    @category = Category.new(name: " ")
    assert_not @category.valid?
end
```

Si no queremos crear el category en cada test podemos usar setup. **Este corre algo al comienzo de cada test**

```
def setup
  @category = Category.new("Fantasy") You
end
```

182. Categories controller and tests

Functional Tests

Categories controller:

- new category
- show category details
- categories index (list)

show category details -> show all movies on that category

Generador

```
~/rails_projects_2020/rails_6_projects/alpha-blog — bash
$ rails generate test_unit:scaffold category
```

Solo testear solo controladores

```
FINISHED IN 0.4782773, 17.0115 runs,  
8 runs, 5 assertions, 0 failures, 3  
category_system_test... $ rails test test/controllers/  
end
```

o solo un controlador

```
rails test test/controllers/categories_controller_test.rb
```

Esos test de módulos y controlador están testeando cosas individuales (Test funcionales)

Queremos hacer un test de integración para ver que todo funcione en conjunto

186. Integration test: Create category business process

Test funcional vs integración

Funcional o test de controlador

Es para checar que las funciones individuales del controlador funcionen

Integración

Testea el full business process y puede incluir múltiples funciones. Checa que estas funcionen juntas

Página para saber más de testing

google.com/search?q=activerecord+assertions&oq=activerecord+assertions&aqs=chrome.0.69i59.2217j0j4&sourceid=

The screenshot shows a Google search results page. The search query 'activerecord assertions' is entered in the search bar. Below the search bar, there are filter options: 'All' (selected), 'Images', 'Videos', 'News', 'Shopping', 'More', 'Settings', and 'Tools'. The search results indicate 'About 663.000 results (0,31 seconds)'. The top result is from 'guides.rubyonrails.org' titled 'Testing Rails Applications — Ruby on Rails Guides'. It includes a snippet about Rails Specific Assertions and links to 'A Guide to Testing Rails ...' and 'Controller tests'.

activerecord assertions

All Images Videos News Shopping More Settings Tools

About 663.000 results (0,31 seconds)

guides.rubyonrails.org Testing

Testing Rails Applications — Ruby on Rails Guides

Jump to **Rails Specific Assertions** - 2.5 Rails Specific Assertions. Rails adds some custom assertions of its own to the minitest framework: Assertion ...

A Guide to Testing Rails ...
Understand Rails testing terminology; Write unit ...
[More results from rubyonrails.org »](#)

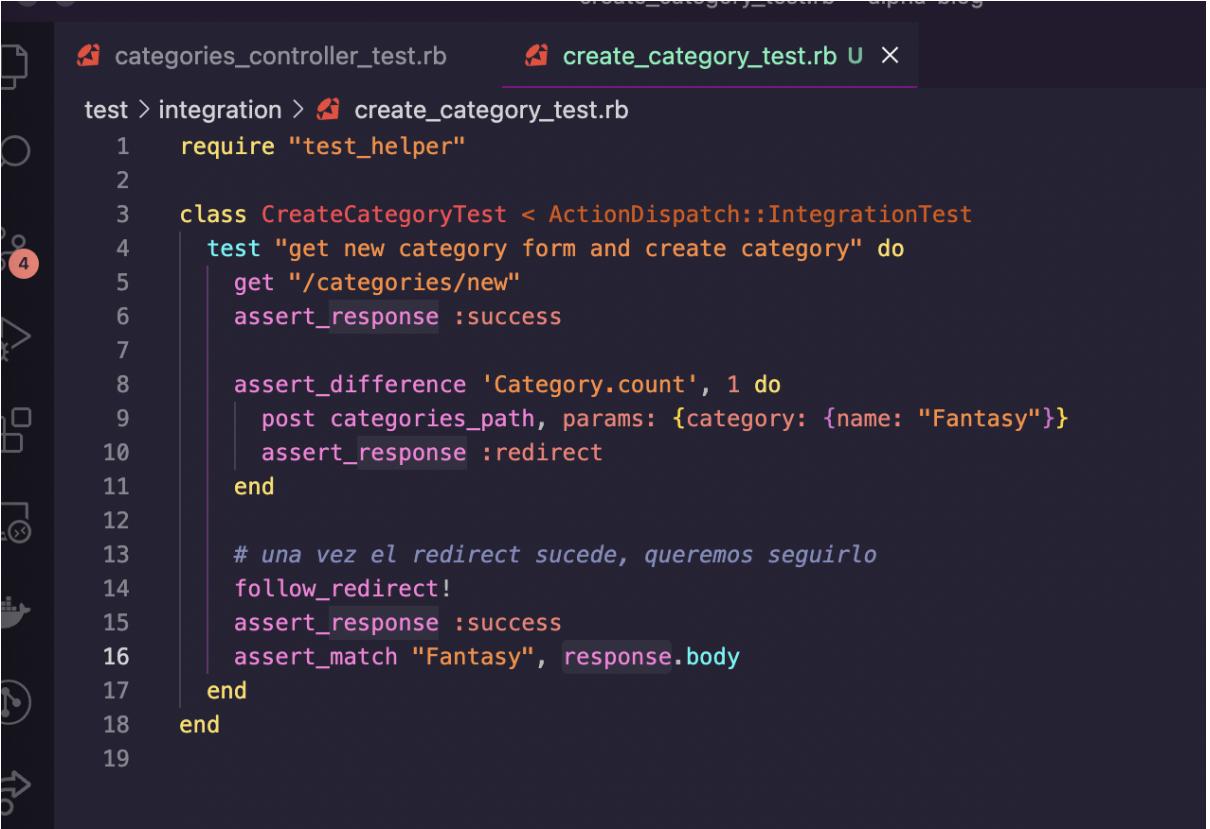
Controller tests
Rails testing terminology. How to write unit, functional, integration ...

Generar integration test

The screenshot shows a terminal window with the command '\$ rails generate integration_test create_category' being typed. The terminal also displays environment variables like 'TERM_PROGRAM=Apple_Terminal' and the path '/Users/mashrurhossain/.rvm/bin'.

```
mashrurhossain...m_bin_path=/Users/mashrurhossain/.rvm/bin TERM_PROGRAM=Apple_Terminal ~/rails_projects  
$ rails generate integration_test create_category  
The dependency 'active_record' (~> 5.0) will be unused by default
```

Haremos un test de integración de todo el proceso de crear una categoría



The screenshot shows a code editor with a dark theme. On the left is a sidebar with various icons. The main area has two tabs: "categories_controller_test.rb" and "create_category_test.rb". The "create_category_test.rb" tab is active, showing the following code:

```
test > integration > create_category_test.rb
1   require "test_helper"
2
3   class CreateCategoryTest < ActionDispatch::IntegrationTest
4     test "get new category form and create category" do
5       get "/categories/new"
6       assert_response :success
7
8       assert_difference 'Category.count', 1 do
9         post categories_path, params: {category: {name: "Fantasy"}}
10        assert_response :redirect
11      end
12
13      # una vez el redirect sucede, queremos seguirlo
14      follow_redirect!
15      assert_response :success
16      assert_match "Fantasy", response.body
17    end
18  end
19
```

Eso fue integración pues fue todo un flow, desde entrar a index, ir hacia la página de new, hacer el post, ser redirigido hasta ver la página de la categoría.

188. Integration test for invalid category

190. Integration test and feature: listing categories

The integration test now will be create two categories and check if these two categories are shown on the index page

192. Admin user requirement and test

Podemos poner el método de login en el archivo test_helper

197. Many-to-many association - back-end implementation

```
> models > category.rb
  class Category < ApplicationRecord
    validates :name, presence: true, length: {minimum: 3, maximum: 255}
    validates_uniqueness_of :name
    has_many :article_categories
    has_many :articles, through: :article_categories
  end

  class Article < ApplicationRecord
    belongs_to :user
    has_many :article_categories
    has_many :categories, through: :article_categories
    validates :title, presence: true, length: {minimum: 6, maximum: 255}
    validates :description, presence: true, length: {minimum: 10, maximum: 500}
  end

> models > article_category.rb
  class ArticleCategory < ApplicationRecord
    belongs_to :article
    belongs_to :category
  end
```

Agregar a la tabla article_category

Se puede desde article.categories << article o category.articles << category

199. Add association from UI

Create article with category

```
rails c
Loading development environment (Rails 6.0.2.1)
2.6.3 :001 > article = Article.new(title: "some title", description: "some description", user: User.last, category_ids: [1, 2])
```

Since they are multiple ids you add an s to "category_ids"

The screenshot shows a search results page with the query "rails collection_select multiple options". The results include a link to a Stack Overflow question about selecting multiple options in a collection select. Below the search bar, there are filters for "All", "Shopping", "Images", "News", "Videos", and "More", along with "Settings" and "Tools" buttons. The search bar has a microphone and search icon. At the bottom, there are links for "RSpec", "Ruby", and "Ruby on Rails". A navigation bar at the top includes "method" and "collection_select". A progress bar at the bottom indicates the search took 0.44 seconds.

201. Update article views to display categories

Render partial of each category inside article

```
<% if @article.categories.any? %>
  <%= render @article.categories %>
<% end %>
```

203. Complete category index and show views

207. Deploy to production, homework, wrap up
section 7

Components useful for building web apps

front-end, back-end etc.

- Associations, one-to-many, many-to-many
- Front-end implementation and styling
- Back-end implementation, REST
- Git, GitHub, local dev and production (Heroku)

Homework!!! (testing fun)

* Optional, completion not necessary to move forward in the course

** Solutions not provided

*** Research, code and try solutions

- Create 2 integration tests
 - a) Users sign-up process
 - b) New article creation process