LOST IN MINITEST

the missing guide for Minitest tourists



Rémi Mercier ➜ remimercier.com

ruby.social/@remi

CONFESSION TIME –

CONFESSION TIME –

I *LOVE* RSPEC.

What I like about RSpec –

↳ Strong opinions and conventions about its DSL.
↳ Well documented, widely adopted.
↳ A clear onboarding.
↳ Allows developers to pick a standard and move on with what matters: testing.

Minitest, on the other hand –



👋 Hey, I'm Rémi Mercier.

↳ Freelance software engineer, writer and former stained-glass master (yup).
↳ I write about building software and products at remimercier.com.
↳ I hang out on ruby.social/@remi.
↳ Need help with you project? Let's talk!

SO, BACK TO MINITEST –

SO, BACK TO MINITEST –

EVERYTHING,

SO, BACK TO MINITEST –

EVERYWHERE,

SO, BACK TO MINITEST –

ALL AT ONCE.

Let's say I need to test this method –

user_test.rb — Ruby

```ruby
class User
  def initialize(first_name:, last_name:)
    @first_name = first_name
    @last_name = last_name
  end

  def full_name
    "#{@first_name.capitalize} #{@last_name.capitalize}"
  end
end
```

I can test it like this –

user_test.rb — Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def test_returns_the_full_name
    user = User.new(first_name: "buffy", last_name: "summers")

    assert_equal "Buffy Summers", user.full_name
  end
end
```

Or like this –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  it "returns the capitalized full name" do
    _(user.full_name).must_equal "Buffy Summers"
  end
end
```

Even like this –

user_test.rb — Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  test "returns the full name" do
    user = User.new(first_name: "buffy", last_name: "summers")

    assert_equal "Buffy Summers", @user.full_name
  end
end
```

All of these are valid, conventional syntaxes –

↳ At it core, Minitest is “just” plain Ruby.

↳ Methods, blocks, for all to see.

Feeling reckless?

You can even write –

user_test.rb — Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Spec
  it "allows me to write stupid tests if I want to" do
    user = User.new(first_name: "buffy", last_name: "summers")

    assert expect(user.full_name).must_equal "Buffy Summers"
  end
end
```

Can I write it? Yes –
Does it works? Sure –
Should I do it? Probably not –

↳ Minitest is unapologetically Ruby.
↳ It comes with sharp knives and no one to tell you not to poke yourself with them.
↳ Hackable / extendable / customizable.
↳ BUT Minitest lacks a proper onboarding and clear documentation.

Minitest comes in many flavors –

Let’s focus on Minitest in Ruby –

Minitest comes in many flavors –

Vanilla Minitest

Vanilla Minitest –

↳ Minitest default syntax with classic assertions.
↳ Created almost 20 years ago by Ryan Davis.
↳ The first implementation was just under 100 lines!
↳ Embedded in Ruby as the default testing framework.

Vanilla Minitest

user.rb Ruby

```ruby
class User
  def initialize(first_name:, last_name:)
    @first_name = first_name
    @last_name = last_name
  end

  def full_name
    "#{@first_name.capitalize} #{@last_name.capitalize}"
  end
end
```

Vanilla Minitest

user_test.rb — Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def setup
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  def test_returns_the_full_name
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Vanilla Minitest

terminal Bash

```
$ ruby user_test.rb
Run options: --seed 64661

# Running:

.

Finished in 0.001239s, 807.1025 runs/s, 807.1025 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

Vanilla Minitest

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def setup
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  def test_returns_the_full_name
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Vanilla Minitest

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def setup
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  def test_returns_the_full_name
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Vanilla Minitest

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def setup
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  def test_returns_the_full_name
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Vanilla Minitest

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def setup
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  def test_returns_the_full_name
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Vanilla Minitest

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

class UserTest < Minitest::Test
  def setup
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  def test_returns_the_full_name
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest comes in many flavors –

Minitest::Spec

Minitest::Spec –

↳ Minitest's version of spec-style assertions (aka expectations).

↳ For those who want to leverage Minitest parallelization and speed, while keeping RSpec-like syntax.

↳ Available in Minitest by default.

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Minitest::Spec –

user_test.rb Ruby

```ruby
require "minitest/autorun"
require_relative "user"

describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Now that we covered Ruby, let's talk about Rails –

Minitest comes in many flavors –

Minitest in Rails

Minitest in Rails –

↳ Minitest is the default testing framework for Ruby on Rails.
↳ Both vanilla and spec syntaxes can be used with Rails.
↳ BUT Rails does NOT port Minitest::Spec by default.
↳ Rails adds custom assertions: mutable data, database changes, time manipulation, mailers, etc...

Minitest in Rails –

user_test.rb Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  setup do
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  test "returns the full name" do
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest in Rails –

user_test.rb Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  setup do
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  test "returns the full name" do
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest in Rails –

user_test.rb Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  setup do
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  test "returns the full name" do
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest in Rails –

user_test.rb Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  setup do
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  test "returns the full name" do
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest in Rails –

user_test.rb Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  setup do
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  test "returns the full name" do
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest in Rails –

user_test.rb Ruby

```ruby
require "test_helper"

class UserTest < ActiveSupport::TestCase
  setup do
    @user = User.new(first_name: "buffy", last_name: "summers")
  end

  test "returns the full name" do
    assert_equal "Buffy Summers", @user.full_name
  end
end
```

Minitest in Rails: beware of inheritance –

↳ Based on the type of test, your test class inherits from a different Rails test class.

Minitest in Rails: beware of inheritance –

Type of test	inherits from →	Rails base test class
Model tests		ActiveSupport::TestCase
Controller tests		ActionDispatch::IntegrationTest
Helper tests		ActionView::TestCase
Integration tests		ActionDispatch::IntegrationTest
Mailer tests		ActionMailer::TestCase
Job tests		ActiveJob::TestCase

Minitest in Rails: custom assertions –

Rails custom assertions –

user_test.rb Ruby

```ruby
class UsersControllerTest < ActionDispatch::IntegrationTest
  test "creates a new user" do
    params = { user: { first_name: "buffy", last_name: "summers" } }

    assert_difference "User.count", 1 { post users_path, params: params }

    user = User.last

    assert_redirected_to user_path(user)
    follow_redirect!
    assert_response :success
    assert_enqueued_email_with UserMailer, :welcome, args: [user]
  end
end
```

Rails custom assertions –

↳ Some of Rails' assertions are documented in the Rails guide.

↳ But a lot of them aren't.

↳ You have to dig into each base class to find the assertions you need.

What about using Minitest::Spec in Rails –

Minitest::Spec in Rails –

Not as easy as that

Minitest::Spec in Rails –

↳ Rails does **not** integrate the Minitest::Spec syntax by default.

↳ Yet, Rails pulls all of Minitest as a dependency.

↳ You need some extra setup: poorly documented and if not done right, it can catch you off guard.

Let's start with the "not done right" –

The wrong way (will trip you up) –

test_helper.rb Ruby

```ruby
require "minitest/spec"

module ActiveSupport
  class TestCase
    extend Minitest::Spec::DSL
  end
end
```

The wrong way (will trip you up) –

↳ Gives you access to `let`, `it`, inner `describe`

↳ But you don't get test class inference, custom test types declaration...

↳ And you'll need to port your custom configuration into Minitest::Spec too.

↳ A lot of moving parts and error-prone.

What is the “right way”, then –

The right way –

Gemfile — Ruby

```ruby
gem "minitest-rails", "~> 8.1.0"
```

↳ Follows the version of your Rails app.
↳ Has an install command that will add necessary configuration.
↳ Infer the ad hoc Rails base class from your test class.

And voilà –

user_test.rb — Ruby

```ruby
describe User do
  let(:user) { User.new(first_name: "buffy", last_name: "summers") }

  describe "#full_name" do
    subject { user.full_name }

    it "returns the capitalized full name" do
      expect(subject).must_equal "Buffy Summers"
    end
  end
end
```

Now, go and test all of the things.

Thank you.