Minitest Cheat Sheet – The Basics

Assert-Style Testing

Test cases are written as standard Ruby classes inheriting from **`Minitest::Test`**. Public instance methods matching the pattern **`test_*`** are treated as tests.

```
class TpsReportTest < Minitest::Test
  def test_report_must_have_cover_sheet
    report = TpsReport.new(cover_sheet: nil)
    assert report.invalid?
  end
end
```

`setup` – Executes before each test

```
def setup
  @report = TpsReport.new(cover_sheet: true)
end
```

`teardown` – Executes after each test

```
def teardown
  @report = nil
end
```

`skip` – Ends the test immediately, result is neither pass or fail

```
def test_memo_received
  skip "Hasn't been written yet"
  # possible other assertions...
end
```

`flunk` – Fail the test immediately

```
def test_tps_report_has_cover_sheet
  flunk "No cover sheet ever"
end
```

Mocking and Stubbing

`Mock#expect` – Defines a method expectation on a mock object

```
mock_report = Minitest::Mock.new
mock_report.expect(:date, Date.today)
```

`Mock#verify` – Check that all method expectations were fulfilled

```
mock_report.verify
```

`Object#stub` – Replace a method within the scope of a block

```
report.stub(:date, Date.today) do
  assert_equal Date.today, report.date   # passes
end
```

Spec-Style Testing

Specs are written using a specialized DSL similar to the one used by RSpec. Test cases and tests are defined using block helper methods, and assertions use a syntax meant to more closely mirror natural language.

```
describe TpsReport do
  it “must have a cover sheet” do
    report = TpsReport.new(cover_sheet: nil)
    expect(report).must_be :invalid?
  end
end
```

`before` – Executes before each test in a given describe scope

```
before do
  @report = TpsReport.new(cover_sheet: true)
end
```

`after` – Executes before each test in a given describe scope

```
after do
  @report = nil
end
```

`describe` – Defines a scope for a collection of related tests

```
describe TpsReport do
  describe “Validations” do
    # …
  end
end
```

`it` – Defines a test method body

```
it “includes all TpsReports for the week” do
  @inbox.must_include @peters_report
end
```

`let` – Creates a named lazy initializer using the requested block

```
let(:report) { TpsReport.first }
```

`subject` – Creates a lazy initializer named subject

```
subject { TpsReport.first }
```

`expect` – Wrap a value before making expectations on it
Aliased as: _, **`value`**

```
expect(@tps_report.date).must_equal Date.today
```

Minitest Cheat Sheet – Assertions

Assertion / Refutation	Description	Examples
`assert/refute`	Returns a “truthy” / “falsy” value	`assert @report.has_cover_sheet?, “No cover sheet”`
`assert_empty/refute_empty`	Responds to `#empty?` and returns `true` / `false`	`assert_empty @inbox`
`assert_equal/refute_equal`	Expected and actual values are equal (==)	`assert_equal :paper_jam, @printer.current_status`
`assert_in_delta` `refute_in_delta`	Values' absolute difference falls in specified range	`assert_in_delta Math:: PI, (22.0 / 7.0), 0.01`
`assert_in_epsilon` `refute_in_epsilon`	Values' relative difference falls in specified range	`assert_in_epsilon 22.55, @item.price / 2.0, 0.01`

`assert_includes`
`refute includes`
	Collection includes the requested object	`assert_includes @calendar.days_of_week, "Monday"`

`assert_instance_of` `refute_instance_of`	Requested object is an instance of the given class	`assert_instance_of String, "Initech"`
`assert_kind_of` `refute_kind_of`	Requested object inherits from the class/module	`assert_kind_of Numeric, @report.page_count`
`assert_match/refute_match`	RegExp argument matches the actual String	`assert_match /synergies/, @report.text`
`assert_mock`	All mock expectations have been satisfied	`assert_mock @report_service_mock`
`assert_nil/refute_nil`	Tested object is `nil`	`assert_nil @report.cover_sheet`
`assert_operator` `refute_operator`	Binary expression prepared using the given arguments evaluates to `true`	`assert_operator @report.page_count, :>=, 20`

`assert_output`
`assert_silent`
	Block produces the expected output on `$stdout` or `$stderr` or remains silent on both	`# exact match on $stdout`
		`assert_output("OK") { system("echo OK") }`
		`# partial match with Regexp on #stderr`
		`assert_output("", /borked/i) { system("status") }`
		`assert_silent { system "find . -name *~ -delete" }`

`assert_predicate` `refute_predicate`	Message composed using the parameters returns true when sent to the tested object	`assert_predicate @report, :submitted?`
`assert_raises`	Block raises an error like the one specified	`assert_raises(ReportFormatError) { @report.submit }`
`assert_respond_to` `refute_respond_to`	Tested object `#responds_to?` requested message	`assert_respond_to "Bill", :length`
`assert_same/refute_same`	Both parameters refer to the same object	`assert_same @new_report, @report_from_db`
`assert_send`	Message sent to the receiver with the requested arguments returns `true`	`assert_send [ @calendar, :no_meetings?, :saturday]`
`assert_throws`	Block throws the expected symbol	`assert_throws(:found) { @haystack.find(“needle”) }`

Minitest Cheat Sheet – Expectations

Expectations	Description	Examples

`must_be/wont_be`	Statement prepared using the requested arguments evaluates to a “truthy” / “falsy” value	`expect(@report).must_be :complete?, “Incomplete”`
		`expect(@report.date).must_be :>, Date.today`

`must_be_empty` **`wont_be_empty`**	Responds to **`#empty?`** and returns **`true`** / **`false`**	`expect(@inbox).must_be_empty`
`must_equal` **`wont_equal`**	Expected and actual values are equal (==)	`expect(@printer.status).must_equal :paper_jam`
`must_be_within_delta` **`wont_be_within_delta`**	Values' absolute difference falls in specified range Alias: **`must_be_close_to/wont_be_close_to`**	`expect(22.0/7.0).must_be_within_delta Math::PI, 0.01`
`must_be_within_epsilon` **`wont_be_within_epsilon`**	Values' relative difference falls in specified range	`expect(@price/2.0).must_be_within_epsilon 22.55, 0.01`
`must_include` **`wont_include`**	Collection includes the requested object	`expect(@calendar.days_of_week).must_include "Monday"`
`must_be_instance_of` **`wont_be_instance_of`**	Requested object is an instance of the given class	`expect("Initech").must_be_instance_of String`
`must_be_kind_of` **`wont_be_kind_of`**	Requested object inherits from the class/module	`expect(@report.page_count).must_be_kind_of Numeric`
`must_match/wont_match`	RegExp argument matches the actual String	`expect(@report.text).must_match /synergies/`
`must_be_nil/wont_be_nil`	Tested object is **`nil`**	`expect(@report.cover_sheet).must_be_nil`

`must_output`	Output captured from **`$stdout`** / **`$stderr`** matches the expected output	`# exact match on $stdout`
		`callable = -> { system("echo OK") }`
		`expect(callable).must_output("OK")`
		`# partial match with Regexp on #stderr`
		`callable = -> { system("service-status") }`
		`expect(callable).must_output("", /borked/i)`

`must_raise`	Callable raises an error like the one specified?	`callable = proc { "TPS Report".submitted? }`
		`expect(callable).must_raise(NoMethodError)`

`must_respond_to` **`wont_respond_to`**	Tested object **`#responds_to?`** requested message	`expect("Bill").must_respond_to :length`
`must_be_same_as` **`wont_be_same_as`**	Both parameters refer to the same object	`expect(@report_from_db).must_be_same_as @new_report`

`must_be_silent`	Block produces no output on **`$stdio`** or **`$stderr`**	`callable = proc { system "find . -name *~ -delete" }`
		`expect(callable).must_be_silent`

`must_throw`	Block throws the expected symbol	`callable = -> { @haystack.search("needle") }`
		`expect(callable).must_throw(:found))`

Minitest Cheat Sheet – Rails Helpers

Declarative Helpers

Rails provides block-style helpers for defining common test methods.

`setup` – Executes before each test

```
setup do
  @report = TpsReport.new(cover_sheet: true)
end
```

`teardown` – Executes after each test

```
teardown do
  @report = nil
end
```

`test` – Defines a test with the given name and body

```
test “includes all TpsReports for the week” do
  @inbox.must_include @peters_report
end
```

Time Helpers

`travel` – Stub a frozen relative time in the future or past

```
travel 1.day
travel -1.week
```

`travel_to` – Stub a frozen absolute date and time

```
travel_to Time.new(2001, 1, 1, 12, 0, 0)
```

`travel_back` – Return to the current time by removing the stub

```
travel 1.day  # time frozen in the future
travel_back   # time returns to normal
```

File Fixtures

`file_fixture` – Get the path for a file in **`test/fixtures/files`**

```
file_fixture "foo.rb"  # = "test/fixtures/files/foo.rb"
```

Request Helpers for Controller Testing

Rails includes helper methods for simulating RESTful HTTP requests.

`get` – GET request (**`:index`**, **`:show`**, **`:new`**, **`:edit`** actions)
`post` – POST request (**`:create`** action)
`put` – PUT request (**`:update`** action)
`patch` – PATCH request (**`:update`** action)
`delete` – DELETE request (**`:destroy`** action)
`head` – HEAD request

Each of these accepts a path String and a standard set of Hash parameters using Ruby 2 keyword arguments:

- **`:params`** – request parameters
- **`:session`** – session variables
- **`:flash`** – Rails flash variables

```
get :index
get :show, params: { id: post.id }
get :new
get :edit, params: { id: post.id }
options = {
  title: "New Post",
  author_name: "CK"
}
post :create, params: options, session: { user_id: id }
put :update, params: { id: post.id, title: “Old Post” }
delete :destroy, params: { id: post.id }
```

Minitest Cheat Sheet – Rails Assertions and Expectations

The table below lists some of the most used Rails-specific assertions along with related variants such as e.g. negative assertions. Expectations and assertions in ***gray italics*** are available when using minitest-rails.

Assertions / Expectations	Description	Examples

assert_difference
assert_no_difference
refute_difference
must_change
wont_change
	Executing the block changes the value of the evaluated String by the given number (default: 1)	`assert_difference "TpsReport.count", 1 do`
		`  post :create, params: { author: "Peter Gibbons" }`
		`end`

assert_emails
assert_no_emails
	Number of emails sent so far or in the passed block equals the expected number	`assert_emails 1`
		`assert_no_emails`

assert_enqueued_emails
assert_no_enqueued_emails
	Number of emails queued for delivery so far or by the passed block equals the expected number	`assert_no_enqueued_emails`
		`assert_emails 1 do`
		`  TpsReportMailer.reminder(user).deliver_later`
		`end`

assert_enqueued_jobs
assert_no_enqueued_jobs
must_enqueue_jobs
wont_enqueue_jobs
	Number of jobs enqueued overall or by the passed block equals the expected number (respecting optional type limitations)	`assert_enqueued_jobs 1`
		`assert_no_enqueued_jobs`
		`must_enqueue_jobs 1, only: DataExtractJob do`
		`  # ...`
		`end`

assert_enqueued_with
must_enqueue_with
	Block must enqueue a job matching expected params (**:type**, :**args**, and/or **:queue**)	`assert_enqueued_with(job: TpsReportJob,`
		`                     queue: "high") do`
		`  TpsReportGenerator.schedule`
		`end`

assert_performed_job
assert_no_performed_jobs
must_perform_jobs
wont_perform_jobs
	Number of jobs performed overall or by the passed block equals the expected number (respecting optional type limitations). Jobs within the block param will be performed, other cases, use **perform_enqueued_jobs**.	`assert_performed_jobs 1`
		`assert_no_performed_jobs`
		`must_perform_jobs 1, only: ProcessCommandJob do`
		`  # ...`
		`end`

assert_performed_with
must_perform_with
	Block must perform a job matching expected params (**:type**, :**args**, and/or **:queue**)	`assert_performed_with(job: TpsReportJob,`
		`                      args: ["2015-01-04"]) do`
		`  TpsReportRunner.run`
		`end`

assert_response
must_respond_with
	Controller action must respond with expected HTTP status or status class (symbol)	`assert_response :success`
		`must_respond_with 302`

assert_redirected_to
must_redirect_to
	Controller must respond with a redirect to the expected URL or path	`assert_redirected_to root_path`
		`must_redirect_to "http://lmgtfy.com/"`

Minitest Cheat Sheet – Capybara

Navigation

`visit` – Navigate directly to a page

```
 visit root_path
```

`current_path` – The current location of the virtual browser

```
 assert_equal root_path, current_path
```

Clicking

Click methods generally work for any applicable identifying characteristic of the clickable item:

```
 click_link "#post_123"     # DOM ID
 click_link "Visit Post"    # link text
 click_button "Save"        # button label
 click "Next Page"          # clicks link or button
```

Form Elements

Fill in and work with all types of form elements:

```
 fill_in "Name", with: "Chris"      # text, textarea
 choose "Yes"                       # radio button
 check "Banana"                     # checkbox
 uncheck "Watermelon"               # checkbox
 attach_file "Report",               # file upload
             file_fixture("a.txt")
 select "Batman", from: "Character" # select
```

Searching and Querying

Capybara can query the DOM using selectors, text, or other identifiers.

```
 find_field "Email"                # field by label
 find_link "#refresh"              # link by DOM ID
 find_button "Generate Report"     # button by text
 find "#post_123"                  # anything, by DOM ID
 all "li"                          # all <li> elements
```

Scoping

Scope queries and other operations to part of the DOM tree.

```
 within("form") do
   fill_in "Username", with: "lumbergh"
   fill_in "Password", with: "MYPRSHE"
 end
```

minitest-rails-capybara

This gem combines various libraries in a single package with everything needed for implementing acceptance tests in Rails with Minitest.

```
 feature “Browsing” do
   scenario “visit site and look around”
     visit root_path
     click_link “My Profile”
     # ...
   end

   scenario “go to Ajax shopping cart”, js: true do
     visit root_path
     click_link “My Cart”
     # ...
   end
 end
```

It supports a range of assertions and expectations that follow a predictable naming scheme:

```
 assert_*                      must_have_*
 assert_no_*                   wont_have_*
 refute_*
```

For asserting and refuting the presence of content:

```
 page.must_have_content “Your cart is empty.”
 assert_text “Your message was sent.”
 page.wont_have_css “.total_price”, “$19.99”
 assert_no_selector “li:first”, text: “William”
```

For asserting and refuting the presence of DOM elements:

```
 assert_button “Save”
 assert_checked_field “Yes”
 assert_field “#”
 assert_link “Next page”
 assert_select “#payment_methods”
 assert_table “#tasks”
 assert_unchecked_field “No”
```