



# Ruby **A**cceptance **T**esting for **W**eb Applications





## Create a New Repository

Create a new empty repository into which you can push your local git repo.

**NOTE:** If you intend to push a copy of a repository that is already hosted on GitHub, then you should fork it instead.

Project Name

Description

Homepage URL

Who has access to this repository? (You can change this later)

- Anyone** (learn more about **public** repos)
- Upgrade your plan to create more private repositories!

Create Repository

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    get_via_redirect repositories_path
    assert_response :success
    assert_select "a[href=?]", new_repository_path, "Create a new repo"

    get_via_redirect new_repository_path
    assert_response :success

    assert_select "form[action=?][method=post]", repositories_path do
      assert_select "input[name=?][type=text]", "repository[name]"
      assert_select "input[name=?][type=text]", "repository[description]"
      assert_select "input[name=?][type=radio]", "repository[public]"
    end

    post_via_redirect repositories_path, :repository => {
      :name => "rack-test", :public => "true"
    }
    assert_response :success
    assert_select "Repository created."
  end
end
```

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    get_via_redirect repositories_path
    assert_response :success
    assert_select "a[href=?]", new_repository_path, "Create a new repo"

    get_via_redirect new_repository_path
    assert_response :success

    assert_select "form[action=?][method=post]", repositories_path do
      assert_select "input[name=?][type=text]", "repository[name]"
      assert_select "input[name=?][type=text]", "repository[description]"
      assert_select "input[name=?][type=radio]", "repository[public]"
    end

    post_via_redirect repositories_path, :repository => {
      :name => "rack-test", :public => "true"
    }
    assert_response :success
    assert_select "Repository created."
  end
end
```

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    get_via_redirect repositories_path
    assert_response :success
    assert_select "a[href=?]", new_repository_path, "Create a new repo"

    get_via_redirect new_repository_path
    assert_response :success

    assert_select "form[action=?][method=post]", repositories_path do
      assert_select "input[name=?][type=text]", "repository[name]"
      assert_select "input[name=?][type=text]", "repository[description]"
      assert_select "input[name=?][type=radio]", "repository[public]"
    end

    post_via_redirect repositories_path, :repository => {
      :name => "rack-test", :public => "true"
    }
    assert_response :success
    assert_select "Repository created."
  end
end
```

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    get_via_redirect repositories_path
    assert_response :success
    assert_select "a[href=?]", new_repository_path, "Create a new repo"

    get_via_redirect new_repository_path
    assert_response :success

    assert_select "form[action=?][method=post]", repositories_path do
      assert_select "input[name=?][type=text]", "repository[name]"
      assert_select "input[name=?][type=text]", "repository[description]"
      assert_select "input[name=?][type=radio]", "repository[public]"
    end

    post_via_redirect repositories_path, :repository => {
      :name => "rack-test", :public => "true"
    }
    assert_response :success
    assert_select "Repository created."
  end
end
```

# Webrat

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    visit repositories_path
    click_link "Create a new repo"
    fill_in "Name", :with => "rack-test"
    click_button "Create"
    assert_contain "Repository created."
  end
end
```

# Webrat

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    visit repositories_path
    click_link "Create a new repo"
    fill_in "Name", :with => "rack-test"
    click_button "Create"
    assert_contain "Repository created."
  end
end
```

# Webrat

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    visit repositories_path
    click_link "Create a new repo"
    fill_in "Name", :with => "rack-test"
    click_button "Create"
    assert_contain "Repository created."
  end
end
```

# Webrat

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    visit repositories_path
    click_link "Create a new repo"
    fill_in "Name", :with => "rack-test"
    click_button "Create"
    assert_contain "Repository created."
  end
end
```

# Webrat

```
class RepositoryTest < ActionController::IntegrationTest
  test "create a new repository" do
    visit repositories_path
    click_link "Create a new repo"
    fill_in "Name", :with => "rack-test"
    click_button "Create"
    assert_contain "Repository created."
  end
end
```

# Getting started

```
# config/environments/test.rb  
config.gem "webrat", :version => ">=0.4.4"
```

```
# test/test_helper.rb  
Webrat.configure do |config|  
  config.mode = :rails  
end
```

```
# sudo rake gems:install RAILS_ENV=test
```

# Webrat's Core API

`#visit`

`#click_link`

`#fill_in`

`#check` and `#uncheck`

`#choose`

`#select`

`#attach_file`

`#click_button`

# Works with RSpec

```
describe "repository management" do
  it "should create a repository with a name" do
    visit repositories_path
    click_link "Create a new repo"
    fill_in "Name", :with => "rack-test"
    click_button "Create"
    response.should contain("Repository created.")
  end
end
```

# Works with Shoulda

```
class RepositoryTest < ActionController::IntegrationTest
  context "repository management" do
    should "create a repository" do
      visit repositories_path
      click_link "Create a new repo"
      fill_in "Name", :with => "rack-test"
      click_button "Create"
      assert_contain "Repository created."
    end
  end
end
```

# Works with Cucumber

**Feature:** Manage repositories

**Scenario:** Create repository with name

**When** I create a repository

**Then** I should see "Repository created."

# Works with Cucumber

```
When /^I create a repository$/ do
  visit repositories_path
  click_link "Create a new repo"
  fill_in "Name", :with => "rack-test"
  click_button "Create"
end
```

```
When /^I should see "([^"]*)"$/ do |text|
  response.should contain(text)
end
```



# Sinatra

## Application Frameworks



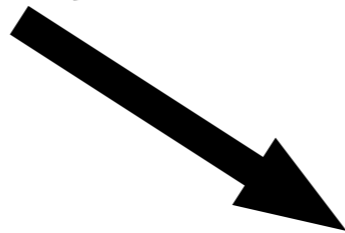
# Matches labels to fields

```
fill_in "Name", :with => "rack-test"
```

```
<label for="repository_name">Name</label>  
<input id="repository_name" name="repository[name]" />
```

# Matches labels to fields

```
fill_in "Name", :with => "rack-test"
```



```
<label for="repository_name">Name</label>  
<input id="repository_name" name="repository[name]" />
```

# Matches labels to fields

```
<label>  
  <input type="checkbox" name="tos" />  
  I accept the Terms of Service  
</label>
```

```
check "I accept the Terms of Service"
```

# Matches labels to fields

```
<label>  
  <input type="checkbox" name="tos" />  
  I accept the Terms of Service  
</label>
```



```
check "I accept the Terms of Service"
```

# Locator strategies

```
field_labeled("E-mail")
```

```
field_with_id("user_email")
```

```
field_named("user[email]")
```

# Locator strategies

```
field_labeled("Name").value.should ==  
  "Mr. Prefilled"
```

```
fill_in field_with_id("user_username"),  
  :with => "brynary"
```

Webrat verifies web app behaviour

OUT OF ORDER  
F A I L

2 HOUR MAXIMUM PARKING TIME

# Verifies HTTP status codes

1) Error:

```
test_create_a_new_repository(RepositoryTest):
```

```
Webrat::PageLoadError: Page load was not successful (Code: 500):
```

```
<snip>
```

```
webrat (0.4.4) lib/webrat/core/session.rb:110:in `request_page'
```

```
webrat (0.4.4) lib/webrat/core/session.rb:205:in `visit'
```

```
(eval):2:in `visit'
```

```
/test/integration/repository_test.rb:5:in `test_create_a_new_repository'
```

# Verifies form fields

1) Error:

```
test_create_a_new_repository(RepositoryTest):
```

```
Webrat::NotFoundError: Could not find field: "Name"
```

```
webrat (0.4.4) lib/webrat/core/locators/locator.rb:14:in `locate!'
```

```
webrat (0.4.4) lib/webrat/core/locators/field_locator.rb:21:in `field'
```

```
webrat (0.4.4) lib/webrat/core/scope.rb:327:in `locate_field'
```

```
webrat (0.4.4) lib/webrat/core/scope.rb:50:in `fill_in'
```

```
(eval):2:in `fill_in'
```

```
/test/integration/repository_test.rb:8:in `test_create_a_new_repository'
```

# Verify HTML content

```
response.should contain("Hello, world!")
```

```
response.should have_selector("li", :class => "new",  
  :count => 2)
```

```
response.should_not have_xpath(".//meta[@name = 'robots']")
```

# Verify HTML content

```
response.should have_selector("#album li:nth-child(3)") do |li|
  li.should have_selector("img", :src => photo_path(@photo))
  li.should contain("Vacation Photo")
end
```

weplay

file:///Users/bhelmkamp/p/weplay/tmp/webrat-1239757556.html

Google

Toggle Rack::Bug RailsConf proposals Campfire: Rails 3 TeamZoneSports pragpi weplay Rails API 3000 post to del.icio.us

**weplay** BETA Login

Join Weplay. It's quick, easy and FREE.

OK, let's set up your personal account. Unless otherwise noted, all fields are required.

Tell us about yourself...

First & Last name

E-mail

Gender  Male  Female

Country

ZIP code

Create your username




Username

Password

Optional

Favorite sport

**Weplay is for YOU!**

-  Gives you access to blogs and weekly Q & A's from LeBron, Peyton, Jeter, Jennie and more
-  Let's you connect to other athletes your age
-  Builds a sports profile to share your highlights
- PLAY!** Offers contests featuring superstar pros for you and your team

save\_and\_open\_page

# Webrat adapters

- Rails
- Merb
- Sinatra
- Selenium
- Mechanize

# WWW::Mechanize

```
require "webrat"
require "webrat/mechanize"

session = Webrat::MechanizeSession.new
session.visit "http://google.com/"
session.fill_in "q", :with => "GoGaRuCo"
session.click_button "Google Search"

session.dom.search("h3 a").each_with_index do |link, i|
  puts "#{i+1}) #{link["href"]}"
end
```



# Selenium

```
# test/test_helper.rb
class ActiveSupport::TestCase
  # ...
  self.use_transactional_fixtures = false

  # ...

  setup do |session|
    session.host! "localhost:3001"
  end
end

Webrat.configure do |config|
  config.mode = :selenium
end
```

```
$ rake test:integration
```

```
Started
```

```
==> Waiting for Selenium RC server on port 4444... Ready!
```

```
==> Waiting for Rails application server on port 3001... Ready!
```

```
..
```

```
Finished in 23.54011 seconds.
```

```
2 tests, 2 assertions, 0 failures, 0 errors
```

```
$ rake test:integration
```

```
Started
```

```
==> Waiting for Selenium RC server on port 4444... Ready!
```

```
==> Waiting for Rails application server on port 3001... Ready!
```

```
..
```

```
Finished in 23.54011 seconds.
```

```
2 tests, 2 assertions, 0 failures, 0 errors
```

```
$ rake test:integration
```

```
Started
```

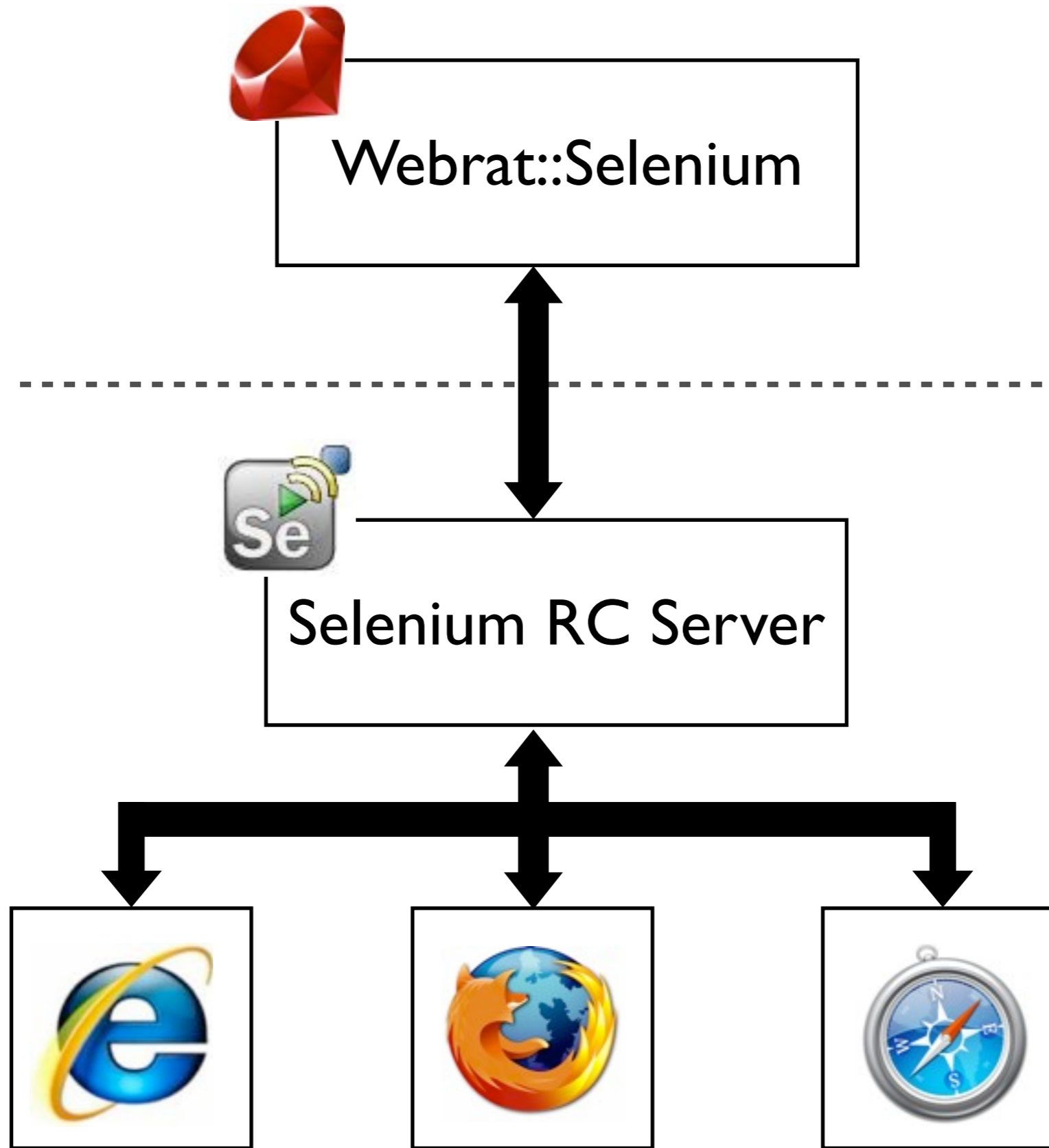
```
==> Waiting for Selenium RC server on port 4444... Ready!
```

```
==> Waiting for Rails application server on port 3001... Ready!
```

```
..
```

```
Finished in 23.54011 seconds.
```

```
2 tests, 2 assertions, 0 failures, 0 errors
```



# One method, two modes

```
def upload_photo
```

```
  webrat.simulate do  
    simulate_uploads
```

```
  end
```

```
webrat.automate do
```

```
  simulate_upload_via_javascript("avatar", "foo.jpg")
```

```
  response.should contain("Uploading image")
```

```
  response.should contain("successfully uploaded")
```

```
  click_link "Continue"
```

```
end
```

```
end
```

# One method, two modes

```
def upload_photo
  webrat.simulate do
    simulate_uploads
  end

  webrat.automate do
    simulate_upload_via_javascript("avatar", "foo.jpg")
    response.should contain("Uploading image")
    response.should contain("successfully uploaded")
    click_link "Continue"
  end
end
```

# One method, two modes

```
def upload_photo
  webrat.simulate do
    simulate_uploads
  end

  webrat.automate do
    simulate_upload_via_javascript("avatar", "foo.jpg")
    response.should contain("Uploading image")
    response.should contain("successfully uploaded")
    click_link "Continue"
  end
end
```

```
class MediaTest < ActionController::IntegrationTest
  test "drag and drop media" do
    album = create_album :title => "Vacation"
    photo1 = create_photo :album => album
    photo2 = create_photo :album => album

    visit album_path(album)
    click_link "Order media"

    selenium.dragdrop("id=#{dom_id(photo1)}", "+350, 0")
    wait_for do
      assert selenium.is_ordered(
        "id=#{dom_id(photo1)}",
        "id=#{dom_id(photo2)}")
    end
  end
end
end
```

```
class MediaTest < ActionController::IntegrationTest
  test "drag and drop media" do
    album = create_album :title => "Vacation"
    photo1 = create_photo :album => album
    photo2 = create_photo :album => album

    visit album_path(album)
    click_link "Order media"

    selenium.dragdrop("id=#{dom_id(photo1)}", "+350, 0")
    wait_for do
      assert selenium.is_ordered(
        "id=#{dom_id(photo1)}",
        "id=#{dom_id(photo2)}")
    end
  end
end
end
```

```
class MediaTest < ActionController::IntegrationTest
  test "drag and drop media" do
    album      = create_album :title => "Vacation"
    photo1     = create_photo :album => album
    photo2     = create_photo :album => album

    visit album_path(album)
    click_link "Order media"

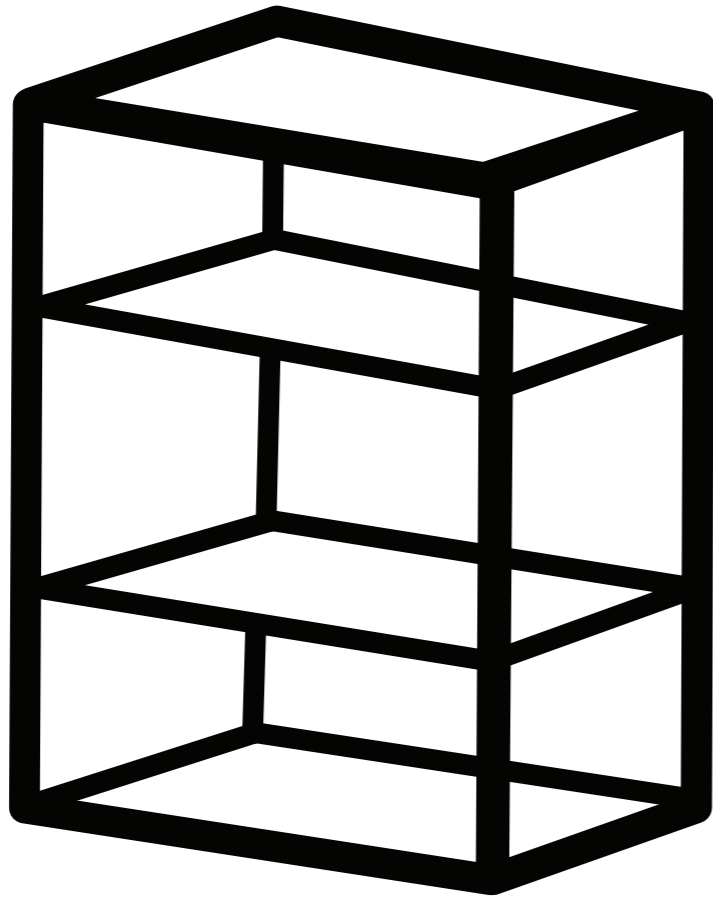
    selenium.dragdrop("id=#{dom_id(photo1)}", "+350, 0")
    wait_for do
      assert selenium.is_ordered(
        "id=#{dom_id(photo1)}",
        "id=#{dom_id(photo2)}")
    end
  end
end
```



# Quick demo

# One More Thing...





**rack**  
powers web applications

# Rack::Test

```
require "rack/test"

class HomepageTest < Test::Unit::TestCase
  include Rack::Test::Methods

  def app
    MyApp.new
  end

  def test_redirect_logged_in_users_to_dashboard
    authorize "bryan", "secret"
    get "/"
    follow_redirect!

    assert_equal "http://example.org/redirected", last_request.url
    assert last_response.ok?
  end
end
```

# Rack::Test API

```
#get(uri, params = {}, env = {})  
#post, #put, #delete, and #head
```

```
#request(uri, env = {})
```

```
#follow_redirect!
```

```
#header(name, value)
```

```
#authorize(username, password)
```

```
#last_request
```

```
#last_response
```

# Webrat adapters

- Rails
- Merb
- Sinatra
- Selenium
- Mechanize

# Webrat adapters

- **Rack::Test**
- Selenium
- Mechanize

The Pragmatic  
Programmers

# The RSpec Book

Behaviour Driven Development  
with RSpec, Cucumber,  
and Friends

*David Chelimsky*  
with *Dave Astels,*  
*Zach Dennis,*  
*Aslak Hellesøy,*  
*Bryan Helmkamp,*  
and *Dan North*

*Edited by Jacquelyn Carter*

The Facets  of Ruby Series

# Thanks

<http://www.flickr.com/photos/tambako/2908186658/>

<http://www.flickr.com/photos/pkmousie/2199520904/>

<http://www.flickr.com/photos/audreyjm529/155024495/>

<http://www.flickr.com/photos/acaben/541334636/>

<http://www.flickr.com/photos/millermz/3267766667/>