Black Box Testing with RSpec and Capybara

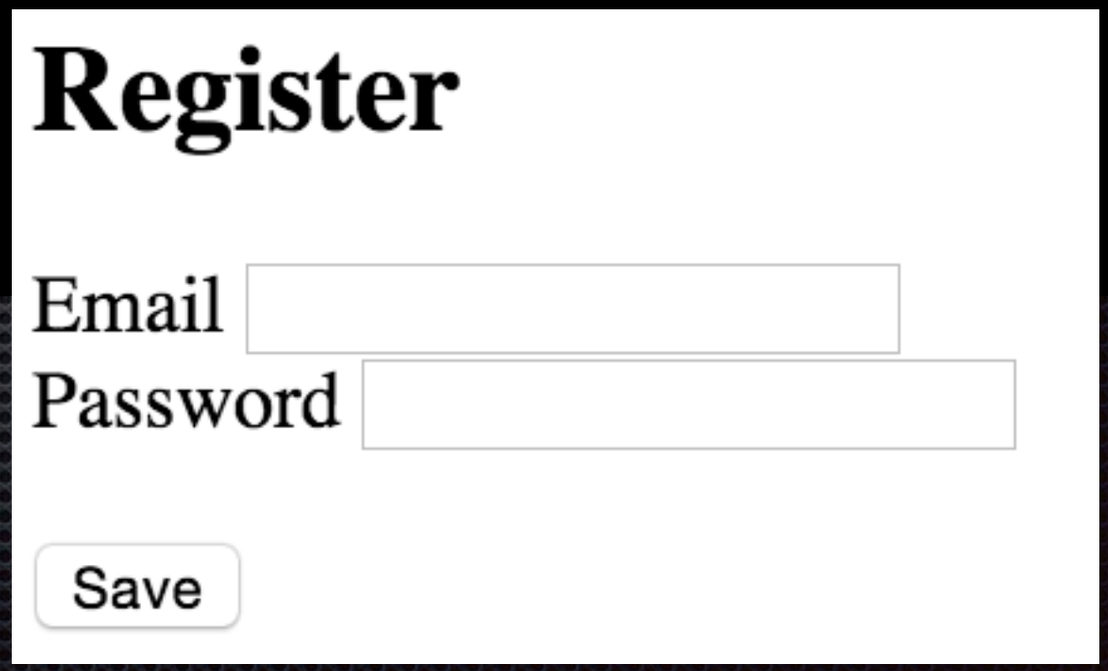
Jillian Rosile, Comverge

Capybara

```
fill_in 'Email', :with => 'user@example.com'

click_button 'Sign In'

find('#css_selector').click

expect(page).to have_content('Welcome')
```

Registration form

```
<html>
  <body>
    <h2>Register</h2>
    <form action="#">
       <label for='email'>Email</label>
       <input type='text' id='email'><br />
       <label for='password'>Password</label>
       <input type='password' id='password'><br /><br />
       <input type='button' id='save_button' name='Save' value='Save'>
    </form>
    </detail>
  </body>
</html>
```



Registration form spec

```
scenario 'registering a user' do
  visit '/register'
  fill_in 'Email', with: 'example@example.com'
  fill_in 'Password', with: 'password'
  click_button 'Save'

  expect(page).to have_no_css 'div.flash.error'
end
```

Adding another spec

```
scenario 'editing a user profile' do
  visit '/register'
  fill_in 'Email', with: 'example@example.com'
  fill_in 'Password', with: 'password'
  click_button 'Save'

  visit '/profile'
  click_button 'Edit'
  expect(page).to have_content 'Editing Profile'
end
```

And more specs!

```
scenario 'registering a user' do
  visit '/register'
  fill_in 'Email', with:
  fill_in 'Password', wi
  click_button 'Save'
```

```
scenario 'editing a user profile' do
  visit '/register'
  fill_in 'Email', with: 'example@example.com'
  fill_in 'Password', with: 'password'
```

```
scenario 'accessing the "order history" page as a registered user' do
  visit '/register'
  fill_in 'Email', with: 'example@example.com'
  fill_in 'Password', with: 'password'
  click

  visit
  expec
end
```

```
scenario 'saving an item to a wish list' do
  visit '/register'
  fill_in 'Email', with: 'example@example.com'
  fill_in 'Password', with: 'password'
  click_button 'Save'

  visit '/games/123'
  click_button 'Save to Wish List'
  expect(page).to have_content 'Successfully Saved To Wish List'
end
```

Page object pattern

```ruby
class RegistrationPage
  include Capybara::DSL

  URL = '/register'

  def visit
    super(URL) # This goes to Capybara's "visit" method
  end

  def register(email: 'example@example.com', password: 'password')
    visit
    fill_in 'Email', with: email
    fill_in 'Password', with: password
    click_button 'Save'
  end

  def has_no_flash_error?
    has_no_css?('div.flash.error')
  end
end
```

Revised spec

```
scenario 'registering a user' do
  reg_page = RegistrationPage.new
  reg_page.register(
    email: 'example@example.com',
    password: 'password'
  )

  expect(reg_page).to have_no_flash_error
end
```

Simplify other specs

```
scenario 'editing a user profile' do
  RegistrationPage.new.register

  visit '/profile'
  cli
  exp
end
```

```
scenario 'saving an item to a wish list' do
  RegistrationPage.new.register

  visit '/games/123'
  click_button 'Save to Wish List'
  expect(page).to have_content 'Successfully Saved To Wish List'
end
```

```
scena
  Reg

  visit '/orders'
  expect(page).to have_content 'Your Order History'
end
```

Making more page objects

```ruby
class RegistrationPage
  include Capybara::DSL

  URL = '/register'

  def visit
    super(URL) # This goes to Capybara's "visit" method
  end

  def register(email: 'example@example.com', password: 'password')
    visit
    fill_in 'Email', with: email
    fill_in 'Password', with: password
    click_button 'Save'
  end

  def has_no_flash_error?
    has_no_css?('div.flash.error')
  end
end
```

Page base object

```
class Page
  include Capybara::DSL
  attr_reader :url

  def initialize(url)
    @url = url
  end

  def visit
    super(url)
  end
end
```

Other shared methods

```
class Page

  # methods from last slide

  def flash_error_css
    'div.flash.error'
  end

  def flash_error
    find(flash_error_css)
  end

  def has_flash_error?(error)
    has_css?(flash_error_css, :text => error)
    # has_css? waits and returns true as soon as the css is present
  end

  def has_no_flash_error?(error)
    has_no_css?(flash_error_css, :text => error)
    # has_no_css? waits and returns true as soon as the css is not present
  end
end
```

New RegistrationPage

```
class RegistrationPage < Page
  def initialize
    super('/register')
  end

  def register(email:, password:)
    visit
    fill_in 'Email', with: email
    fill_in 'Password', with: password
    click_button 'Save'
  end
end
```

Page sections

```
<div class="edit-dialog">
  <form action="#">
     <label for='first_name'>First Name</label>
     <input type='text' id='first_name'><br />
     <label for='last_name'>Last Name</label>
     <input type='text' id='password'><br />
     <input type='button' id='save_button' name='Save' value='Save'>
  </form>
</div>
```

First Name
Last Name
Save

Page sections

```
class EditDialog < DelegateClass(Capybara::Node::Element)
  def initialize
    super(Capybara.page.find('div.edit-dialog'))
  end

  def edit_name(first_name, last_name)
    find_field('First Name').set(first_name)
    find_field('Last Name').set(last_name)
    find_button('Save').click
  end
end
```

http://ruby-doc.org/stdlib-1.9.3/libdoc/delegate/rdoc/Object.html

Similar sections

```
<div class="edit-dialog">
  <form action="#">
     <label for='first_name'>First Name</label>
     <input type='text' id='first_name'><br />
     <label for='last_name'>Last Name</label>
     <input type='text' id='password'><br />
     <input type='button' id='save_button' name='Save' value='Save'>
  </form>
</div>
```

First Name
Last Name
Save

```
<html>
  <body>
    <h2>Register</h2>
    <form action="#">
       <label for='email'>Email</label>
       <input type='text' id='email'><br />
       <label for='password'>Password</label>
       <input type='password' id='password'><br /><br />
       <input type='button' id='save_button' name='Save' value='Save'>
    </form>
    </detail>
  </body>
</html>
```

Register

Email
Password
Save

Shared form object

```
form = Form.new(:first_name => 'First Name', :last_name => 'Last Name')
form.methods
=> first_name_field, last_name_field, fill_in_form
# etc
```

```
form = Form.new(:email => 'Email', :password => 'Password')
form.methods
=> email_field, password_field, fill_in_form
# etc
```

Sharing similar sections

```ruby
class RegistrationPage < Page
  include Forwardable

  attr_reader :form

  def initialize
    super('/register')

    form_fields = {email: 'Email', password: 'Password'}
    form = Form.new(form_fields)
    def_delegators :form, *form.field_method_names
  end

  def register(email: , password: )
    visit
    form.fill_in_form(email: email, password: password)
  end
end
```

Form implementation

```
class Form
  include Capybara::DSL

  attr_accessor :fields

  def initialize(fields)
    @fields = fields
    define_field_methods
  end

  def fill_in_form(fields)
    fields.each do |field, value|
      f = send("#{field}_field")
      f.set(value) || Array(value).each { |val| f.select(val) }
    end
  end
```

(continued next slide)

Form implementation

```
  private

  def define_field_methods
    fields.each do |field, field_string|
      field_method_name = define_field_name(field)
      define_field_method(field_method_name, field_string)
    end
  end

  def define_field_method(field_method_name, field_string)
    field_string = field_string.chomp(':')
    self.class.send(:define_method, field_method_name) do
      find_field(field_string)
    end
  end
end
```

Real world example

```
scenario 'installers can move inventory from warehouses' do
  login_page.authenticate(as: 'installer1')

  inventory_page.visit
  inventory_page.open_inventory_tab

  inventory_page.search_for_device('0000001')
  inventory_page.add_device
  inventory_page.submit

  expect(inventory_page).to have_successfully_updated_inventory

  inventory_page.open_detail_tab
  expect(inventory_page).to have_device_count(1)
end
```

Real world example

```ruby
scenario 'installers can move inventory from warehouses' do
  class WrongNumberOfDevices < StandardError; end

  if Capybara.current_session.has_link?('Sign Out')?
    visit '/dashboard'
    find_link('Sign Out').click
  end

  visit '/inventories'
  find_link('Add To My Inventory').click
  find_field('Serial # / HAN ID').set('0000001')
  find_button('Add').click
  find_button('Submit').click
  has_flash_message?('Successfully updated selected inventory.')
  expect(page).to have_css?('div.flash.notice', :text => 'Successfully updated selected inventory.')

  find_link('Detail').click

  has_one_device = begin
    page.document.synchronize(Capybara.default_wait_time, :errors => [WrongNumberOfDevices, Selenium::WebDriver::Error::StaleElementReferenceError]) do
      raise WrongNumberOfDevices unless all('table.striped tr + tr').count == 1
    end
    true
  rescue WrongNumberOfDevices
    devices.count == 1
  end

  expect(has_one_device).to be_true

  find('table.striped tr + tr', :text => '0000001').find('input[type="checkbox"]').set(true)
  find('div#assignment_fields').find_field('Location').select('Qatest Warehouse')
  find('div#assignment_fields').click_button('Submit')

  expect(has_css?('table.striped tr + tr')).to be_false
end
```

Tests are code

Tests are **not** exempt from the rules of code

Badly written tests often cause **more** maintenance problems than badly written application code

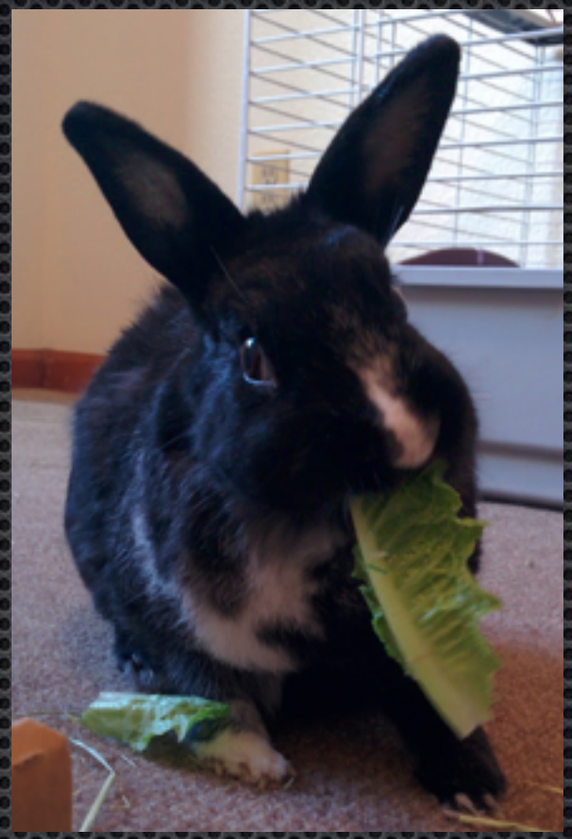
Tests must change **less often** than your application code

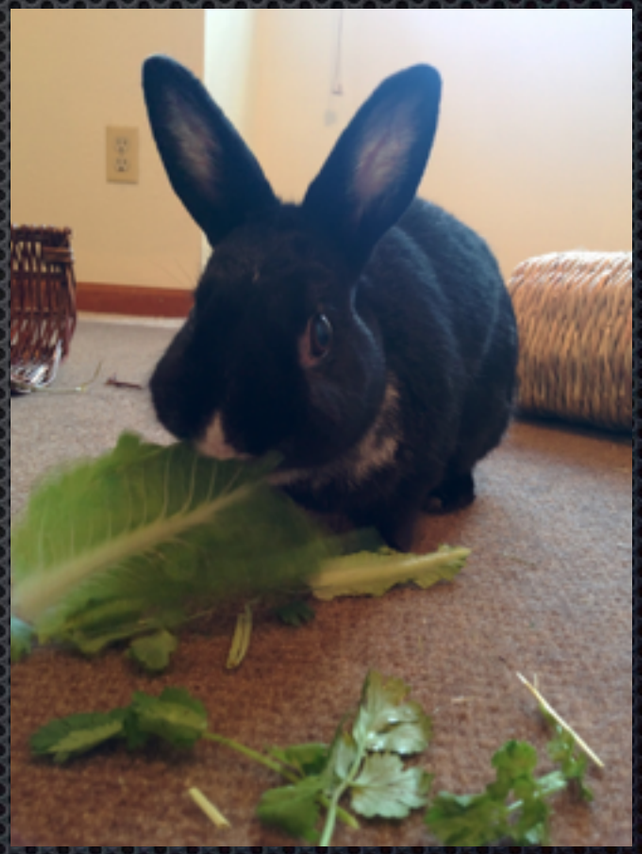
Jillian Rosile

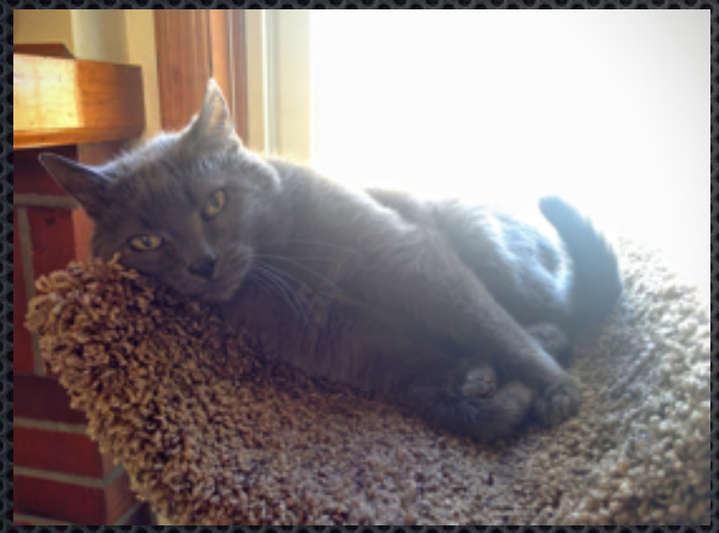
Comverge

@jillianrosile

jillianrosile.github.io







Thank you to Eric Dobbs and Glen Aultman-Bettridge

https://github.com/gamera-team/gamera