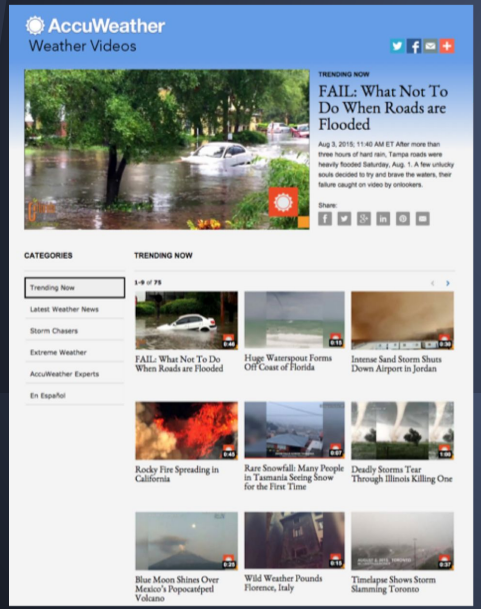
Automation Framework

Ruby - Capybara - Rspec - Site_Prism



Automation Framework Goals

1. Having the ability to quickly and easily create simple automated tests
   - Automated tests should be simple to learn, write, and understand by other human

2. Having the ability to easily maintain tests when the application changes
   - Application will need to change, automation framework should support and facilitate change
   - It is always a better choice to put the complexity into the framework instead of into the tests

2. Having the ability to switch test environments and test targets
   - Tests can run against different environments/browsers: local, stage, qa, or production
   - Tests can be executed from different platforms: build machine, or dev Mac, Linux, Windows
   - Tests can run against headless browser, remote devices, remote browsers

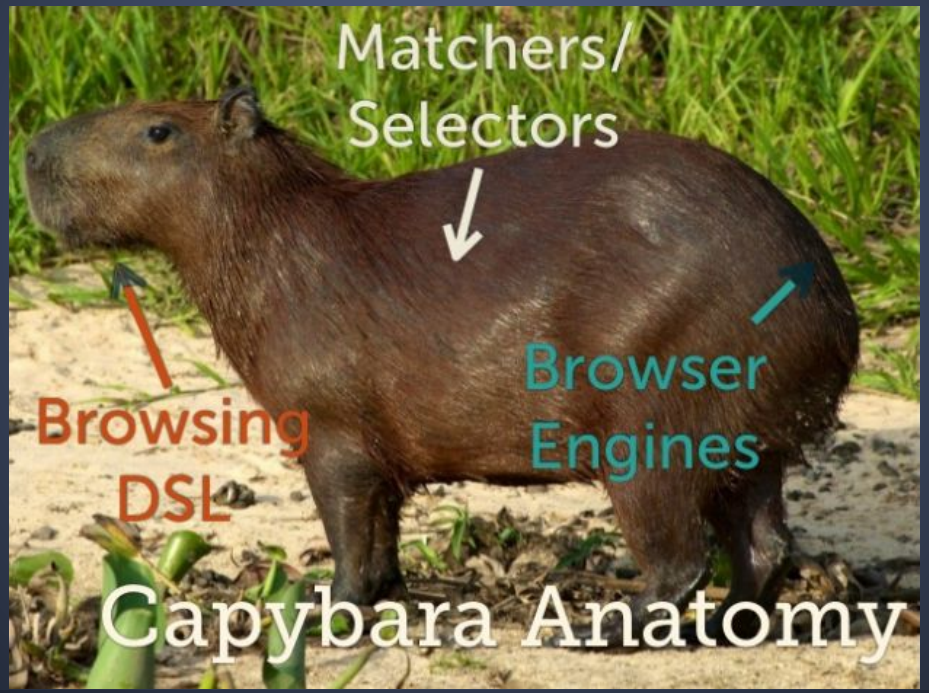
Introducing Capybara

Friendly yet exotic pet!

Capybara

"Makes it easy to simulate how a user interacts with your application"

A ruby gem with Intuitive API and methods - mimics the language an actual user would use



Capybara

https://github.com/jnicklas/capybara

Capybara is agnostic about the driver running your tests

- Backend switching - run tests against fast headless or full feature browsers

  Drivers: Poltergeist, Selenium Webdriver, ChromeDriver, BrowserStack
  Browsers: PhantomJS, Firefox, Chrome, BrowserStack supported browsers

- Powerful synchronization - built-in wait for asynchronous processes to complete

SitePrism

A Page Object Model DSL for Capybara

+ Capybara DSL
+ Flexible driver selection
+ Works with logical elements
+ Speed

+ Less dependencies
+ Works with html-elements
- Doesn't work with headless browsers
- Speed

Site_Prism

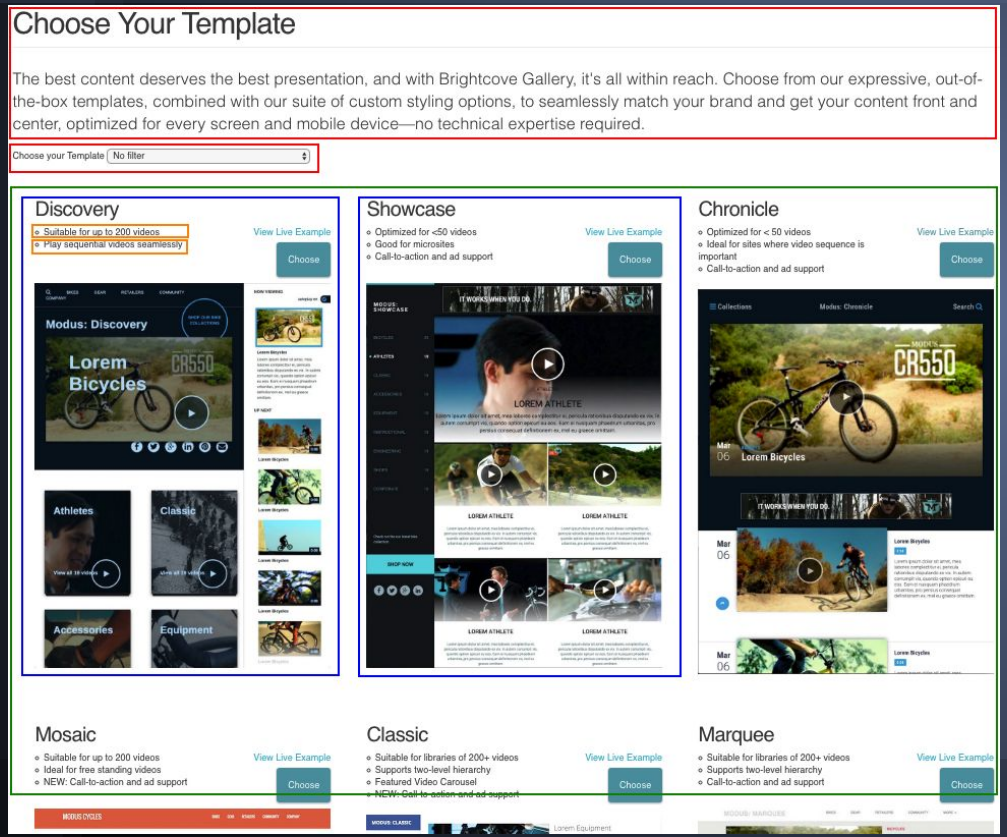
https://github.com/natritmeyer/site_prism

A Page Object Model DSL for Capybara

- Simple, clean and semantic for describing your site using the Page Object Model pattern
- Allow grouping of sections, elements
- Provide built-in helper methods
  - @page.has_<element>? (use Capybara built-in wait, return true/false value)
  - expect(@page).to have_<element> (use Capybara wait, return true/false value)
  - @page.wait_for_<element> (wait until element exist or timeout after Capybara max wait time)
  - ...

Site_Prism

Example page with dynamic number of objects (options, templates)




```ruby
class Header < SitePrism::Section
  element :heading, "h2"
  element :desc, "div.clearfix p"
end

class Filter < SitePrism::Section
  element :dropdown, "select#filterTemplateFeatureDropdown"
  elements :options, "select#filterTemplateFeatureDropdown option"
end

class Thumbnail < SitePrism::Section
  element :link, "a"
  element :img, "img.template-thumbnail"
  element :thumbnail_text, "span.thumbnail_text-content span"
end

class Templates < SitePrism::Section
  element :name, "div.col-sm-12.site-meta h3"
  elements :links, "a"
  elements :desc_bullets, "ul.template-description-bullets li"
  element :choose, "a.btn.btn-secondary.choose-main"
  section :thumbnail, Thumbnail, "div.col-sm-12.thumbnail"
end

class SitesTemplatesPage < SitePrism::Page
  set_url "/#sites/create"
  section :header, Header, "div#top"
  section :filter, Filter, "div.templateFilter"
  sections :templates, Templates, "div.row.template-container"
end
```

Site_Prism

Example spec utilizing page objects

Navigate to Sites page

```
describe "Sites - Templates validation", :type => :feature, :js => javascript_flag do
  before :each do
    @login_page = LoginPage.new
    @sites_page = SitesPage.new
    @templates_page = SitesTemplatesPage.new
    @sites_page.load
  end
```

Login and brings up Templates page

```
it "validate templates" do
  @login_page.login("gallery@brightcove.com")
  @sites_page.create_site.click
  expect(@templates_page.header.heading.text).to have_text 'Choose Your Template'
```

Site_Prism

Use dynamic array of objects

After bringing up Templates page

- go through each filter options
- go through all Templates in each filter
- validate image
- validate example link

```
26      @templates_page.wait_until_filter_visible
27      puts "\n*** Checking all template filter options and associated sites ***"
28      for f in 0..@templates_page.filter.options.size-1
29        puts "\nSelected template filter: " + @templates_page.filter.options[f].text
30        @templates_page.filter.options[f].click
31        puts "Number of templates for this filter: #{@templates_page.templates.size}"
32
33        for i in 0..@templates_page.templates.size-1
34          puts "Checking " + @templates_page.templates[i].name.text + " template..."
35          if f == 0 # only validate template thumbnails once from the No filter
36            puts "  validating thumbnail image"
37            img_url = @templates_page.templates[i].thumbnail.img['src']
38            if HTTParty.get(img_url).code == 200
39              valid_urls.push(img_url)
40            else
41              invalid_urls.push(img_url)
42            end
43            puts "  validating Example link"
44            if @templates_page.templates[i].links[0].text.include? "Example"
45              example_link = @templates_page.templates[i].links[0]['href']
46              if example_link.include? "http"
47                if HTTParty.get(example_link).code == 200
48                  valid_urls.push(example_link)
49                else
50                  invalid_urls.push(example_link)
51                end
52              end
53            else
54              puts "    this template does not have example link"
55            end
56          end
57        end
58          puts "  validating template chooser"
59          @templates_page.templates[i].has_choose?
60      end
```

BrowserStack integration

```ruby
# BrowserStack driver
url = "http://#{ENV['BS_USERNAME']}:#{ENV['BS_AUTHKEY']}@hub.browserstack.com/wd/hub"
Capybara.register_driver :browserstack do |app|
  capabilities = Selenium::WebDriver::Remote::Capabilities.new

  if ENV['OS']
    capabilities['os'] = ENV['OS']
    capabilities['os_version'] = ENV['OS_VERSION']
  else
    capabilities['platform'] = ENV['PLATFORM'] || 'ANY'
  end
  capabilities['device'] = ENV['DEVICE'] if ENV['DEVICE']
  capabilities['browserName'] = ENV['BROWSERNAME'] if ENV['BROWSERNAME']

  capabilities['browser'] = ENV['BROWSER'] || 'chrome'
  capabilities['browser_version'] = ENV['BROWSER_VERSION'] if ENV['BROWSER_VERSION']

  capabilities['browserstack.debug'] = ENV['DEBUG'] ? ENV['DEBUG'] : 'false'
  capabilities['project'] = ENV['PROJECT'] if ENV['PROJECT']
  capabilities['build'] = ENV['BUILD'] if ENV['BUILD']

  Capybara::Selenium::Driver.new(app,
    :browser => :remote, :url => url,
    :desired_capabilities => capabilities)
end
```

Env switching

config/capybara.yml

```
new-prod-phantom:
  default_driver: :firefox
  javascript_driver: :poltergeist
  javascript: true
  timeout: 15
  app_host:
https://studio.brightcove.com/products/videocloud/gallery

new-qa-firefox:
  default_driver: :firefox
  javascript_driver: :poltergeist
  javascript: false
  timeout: 15
  app_host:
https://studio.brightcove.com/products/bctestgallery/gallery-qa

...
```

select env specific values

```
# read config values from capybara.yml
config = YAML.load_file("config/capybara.yml")
env = ENV['TEST_ENV'] ? ENV['TEST_ENV'] : 'qa-firefox'
javascript_flag = config[env]['javascript']

describe "Sites - Templates validation", :type =>
:feature, :js => javascript_flag do
```

Modularize shared functions

- utilizing shared functions to minimize repeatable codes
- try to make shared functions backward compatible and
- try to harden tests from timing error with try / catch / retry

```
describe "End2End - Chronicle template", :type => :feature, :js => javascript_flag do
  before :each do
    @login_page = LoginPage.new
    @sites_page = SitesPage.new
    @edit_page = SitesEditPage.new
    @templates_page = SitesTemplatesPage.new
    @sites_page.load
  end

  it "created and validated Chronicle template" do
    template = "Chronicle"
    @login_page.login("gallery@brightcove.com")
    # product_message('close')
    select_template(template)
    # dismiss_site_editor_tutorial
    prepopulate = ENV['PREPOP'] ? ENV['PREPOP'] : 'true'
    prepopulate_site(prepopulate)
    # enable_social_sharing
    add_site_description(template)
    # add_collection
    enable_related_link(template)
    # enable_autoplayNext(template)
    enable_video_download(template)
    publish_validate_site(template)
  end
end
```

Example Shared functions

```
def prepopulate_site(prepopulate)
  if (@edit_page.has_prepop_dialog?)
    puts "Pre-populate choice = " + prepopulate
    puts "Checking dialog: " + @edit_page.prepop_dialog.header.title.text
    expect(@edit_page.prepop_dialog.header.title.text.include? "Pre-populate").to eq true
    if Capybara.current_driver == :poltergeist
      if prepopulate == 'false'
        puts " choosing option: " + @edit_page.prepop_dialog.footer.cancel.text
        @edit_page.prepop_dialog.footer.cancel.trigger('click')
      elsif prepopulate == 'true'
        puts " choosing option: " + @edit_page.prepop_dialog.footer.accept.text
        @edit_page.prepop_dialog.footer.accept.trigger('click')
        sleep 3
      end
    else
      if prepopulate == 'false'
        puts " choosing option: " + @edit_page.prepop_dialog.footer.cancel.text
        @edit_page.prepop_dialog.footer.cancel.click
      elsif prepopulate == 'true'
        puts " choosing option: " + @edit_page.prepop_dialog.footer.accept.text
        @edit_page.prepop_dialog.footer.accept.click
        sleep 3
      end
    end
  else
    puts "No Pre-populate dialog"
  end
end
```

Parallel tests

https://github.com/grosser/parallel_tests

ParallelTests splits tests into even groups and runs each group in a single process

```
gem 'parallel_tests'
```

TEST_ENV=qa-phantom rake spec:suite:end2end
TEST_ENV=qa-phantom rake parallel:spec[end2end]
TEST_ENV=qa-phantom rake parallel:rake[spec:suite:end2end]
TEST_ENV=new-qa-phantom parallel_rspec -n 2 spec/specs/end2end

Caveats:

Jenkins job output has mixed logging between the processes

Studio has a limit of 6 concurrent sessions for a login -> limit to 2 processes per job to handle multiple jobs running against different environments or use entirely different logins.

Run Tests on Jenkins

http://ci.gallerydev.net:8080/job/gallery_qa_end2end_GA_templates (headless phantomJS)

TEST_ENV=new-qa-phantom parallel_rspec -n 2 spec/specs/end2end

http://ci.gallerydev.net:8080/job/gallery_qa_autoplayNext_BrowserStack (Firefox, Chrome, IE)

rake spec:suite:autoplay TEST_ENV=qa-browserstack OS=WINDOWS OS_VERSION=7 BROWSER=FIREFOX PROJECT=$JOB_NAME BUILD=$BUILD_NUMBER BS_USERNAME=userx BS_AUTHKEY=keyx

rake spec:suite:autoplay TEST_ENV=qa-browserstack OS=WINDOWS OS_VERSION=8 BROWSER=CHROME PROJECT=$JOB_NAME BUILD=$BUILD_NUMBER BS_USERNAME=userx BS_AUTHKEY=keyx

rake spec:suite:autoplay TEST_ENV=qa-browserstack OS=WINDOWS OS_VERSION=7 BROWSER=IE BROWSER_VERSION=10 PROJECT=$JOB_NAME BUILD=$BUILD_NUMBER BS_USERNAME=userx BS_AUTHKEY=keyx

Run tests with rake or rspec

Run test suites with rake, auto retry failed spec

- create Rakefile - add gem ‘rake’ to Gemfile and require 'rspec/core/rake_task'
- rake -T (build/list all rake tasks)
- rake spec:suite:all (run all spec suites against default TEST_ENV)
- rake spec:suite:sites (run sites spec suite against default TEST_ENV)
- TEST_ENV=qa-phantom rake spec:suite:sites (run sites spec suite against qa-phantom TEST_ENV)
- optional parameters: RETRY_COUNT (default = 3)

Run test suites with rspec

- TEST_ENV=qa-phantom rspec spec/specs/sites (run all specs in a directory on phantomjs browser)
- TEST_ENV=qa-firefox rspec spec/specs/sites/sites_edit_spec.rb (run a specific spec file on firefox browser)

Debug a spec

use of rspec --tag option

rspec --tag debug # to include

rspec --tag ~debug # to exclude

```
15  it "navigate Sites page", :debug => true do
16    login
17
18    binding.pry
19    @sites_page.has_sub_nav?
20    expect(@sites_page.sub_nav.sites).to have_text('My', 'Sites')
```

use of pry and pry-nav gems

binding.pry

(step, next, continue, exit, exit!)

save and open a screenshot

open_screenshot

```
$ rspec --tag debug
Run options: include {:debug=>true}

Sites

From: /Users/kpham/work/constellation/webdriver/spec/specs/sites/sites_spec.rb @ line 18 :

    13:   end
    14:
    15:   it "navigate Sites page", :debug => true do
    16:     login
    17:
 => 18:     binding.pry
    19:     @sites_page.has_sub_nav?
    20:     expect(@sites_page.sub_nav.sites).to have_text('My', 'Sites')
    21:     expect(@sites_page.sub_nav.settings).to have_text 'Settings'
    22:     expect(@sites_page.sub_nav.reports).to have_text 'Reports'
    23:

[1] pry(#<RSpec::Core::ExampleGroup::Nested_2>)> @sites_page.has_sub_nav?
=> true
```

Demo - Run tests...

```
$ TEST_ENV=prod-firefox rake spec:suite:all
Running spec:suite:login ...

Login
  login into Gallery prod-firefox

Finished in 6.29 seconds
1 example, 0 failures
Running spec:suite:sites ...

Sites
  navigate Sites page
  create a Site (FAILED - 1)

Failures:

  1) Sites  create a Site
     Failure/Error: @sites_page.create_site.click
     Capybara::ElementNotFound:
       Unable to find css "[href='#sites/create']"
     # ./spec/specs/sites/sites_spec.rb:42:in `block (2 levels) in <top (required)>'

Finished in 38.39 seconds
2 examples, 1 failure

Failed examples:

rspec ./spec/specs/sites/sites_spec.rb:39 # Sites  create a Site
[2014-10-16 17:24:23 -0400] Failed, retrying 1 failure(s) in spec:suite:sites ...
Run options: include {:full_description=>/Sites\ \ create\ a\ Site/}

Sites
  create a Site

Finished in 12.21 seconds
1 example, 0 failures
```

<- use a Rake task to run all spec suites

<- a spec in Sites spec suite failed

<- failed spec is retry

Test environment setup

Prerequisites:

- rvm (https://rvm.io/rvm/install)
- ruby-2.x (rvm install 2.1.0)
- phantomjs webkit (brew install phantomjs)
- chromedriver (brew install chromedriver)
- optional web browsers: firefox, chrome, safari

Setup:

- Clone git repo, then install all gem bundles from Gemfile (bundle install)
- Setup test env in config/capybara.yml with specific host url and settings

What to include

- Gemfile (ruby libraries)

```
gem 'selenium-webdriver'
gem 'poltergeist'
gem 'rspec', '~> 2.14.1'
gem 'capybara'
gem 'site_prism'
```

- spec_helper.rb (capybara, drivers, Rspec configurations)

```
require 'capybara/rspec'
require 'capybara/poltergeist'
require 'selenium-webdriver'
require 'site_prism'
```

- .rspec (allow all specs to include these configuration options)

```
--require spec_helper
```

References

- Ruby - https://www.ruby-lang.org
- Ruby Open-source Libraries - https://rubygems.org
- RSpec Guidelines - http://betterspecs.org
- Capybara - https://github.com/jnicklas/capybara
- SitePrism - https://github.com/natritmeyer/site_prism
- PhantomJS - http://phantomjs.org/
- Poltergeist - https://github.com/teampoltergeist/poltergeist
- BrowserStack - https://www.browserstack.com/list-of-browsers-and-platforms

Thank you

Questions, Comments, and Suggestions?

Thank you!

Khai Pham
khaipham@gmail.com