

AUTOMATION PAGE OBJECT PATTERN WITH OPEN SOURCE FRAMEWORK RUBY - STUDY PAPER

Devanshu Bhatt*

Abstract

Automation testing is extremely crucial method to discover defects and issues in systems, and validate systems on consistent basis possibly with machine learning approach without manual intervention. There are numerous frameworks and tools available for test automation via several different validation and verification methods. This paper describes a Page Object Pattern technique of automation testing. I have developed and executed this framework on windows system with help of Ruby Open source object oriented language and cucumber which is famous for accepted test driven or behavior driven development methodologies, and Selenium web driver which is de facto standard to drive web GUI automation within web browsers in open source community, This identical framework pattern can be configured on Linux and Mac Operating systems. Page Object pattern designed is highly acceptable industry wide in information technology domain among Test

Keywords:

Test Automation Framework; Page Object Pattern; Selenium; Ruby; Cucumber; Watir; Web Automation.

*** Consultant, IT App Development at Nationwide General Insurance. Columbus, OH, U.S.A.**

automation developers, System Architects, DevOps Engineers and Developers because of its high capabilities of abstraction, logical distribution of pages for system under tests, better control over scripting maintenance, configurability and extensibility as well better control over scenarios and features structure design.

1. Introduction

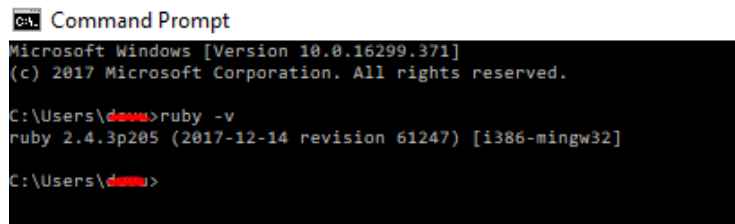
Test Automation tests are a strategy or technique which may be generate with commercially or internally designed software program, services or frameworks to assistance with the validation process, which includes functional and load/stress assessment. Automated scripts deliver constant outcomes and information factors. The advantages are simplicity of maintenance, the capability to effectively use resources, and the ability to generate reports in accordance with the performed tests. Automation testing is incredibly essential technique to uncover flaws and concerns in programs and software systems. This technique validates programs and software systems on constant schedule potentially with machine learning approach without manual intervention. There are several frameworks and tools accessible for test automation via a number of various validation and verification methods.

This paper explains a Page Object Pattern method of automation testing. I have developed and executed this framework on windows operating system with combination of Ruby Open source object oriented language. Cucumber which is well-known for acknowledged to assistance test driven or behavior driven development methodologies, and Selenium web driver which is de facto standard to drive web GUI automation within web browsers in open source community. This identical framework pattern can be configured on Linux and Mac Operating systems. Page Object pattern is extremely adequate industry wide in information technology domain amongst Test automation developers, System Architects, DevOps Engineers and Developers due to the fact of its excessive capabilities of abstraction, logical grouping of pages for system under tests, much better control around scripting maintenance, configurable and extensible as well greater command over scenarios and features design.

2. Research Method - Overview of tools and technology

2.1 Install Open Source - Ruby Binary

Ruby is a vibrant, open source computer programming language with a emphasis on ease-of-use and efficiency. It has an sophisticated syntax that is pure to understand and uncomplicated to write. Download ruby for windows from <https://rubyinstaller.org/>, I have used ruby 2.4.3p205 version in this paper. Once downloaded then install its binary by double clicking on downloaded exe file. Accept all default settings and check box to add environment variable path automatically when prompted during installation. Once it is installed successfully then open your command prompt and write command. **ruby -v**. It will display installed ruby version as per Fig. 1.



```
Command Prompt
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\demu>ruby -v
ruby 2.4.3p205 (2017-12-14 revision 61247) [i386-mingw32]

C:\Users\demu>
```

Figure 1. Installed Ruby version.

2.2 Install IDE - RubyMine

RubyMine IDE consists of a extensive Ruby code editor conscious of vibrant language particulars. It offers intelligent coding guidance, sensible code refactoring, and deep code evaluation abilities. With simple project configuration, automatic Ruby Gems administration, Rake assistance, and built-in consoles, it has almost everything a Ruby developer needs in a development environment.

Download from <https://www.jetbrains.com/ruby/> and install it with default settings, once it is installed

successfully then open it and create new project and add directories and files as per mentioned in below

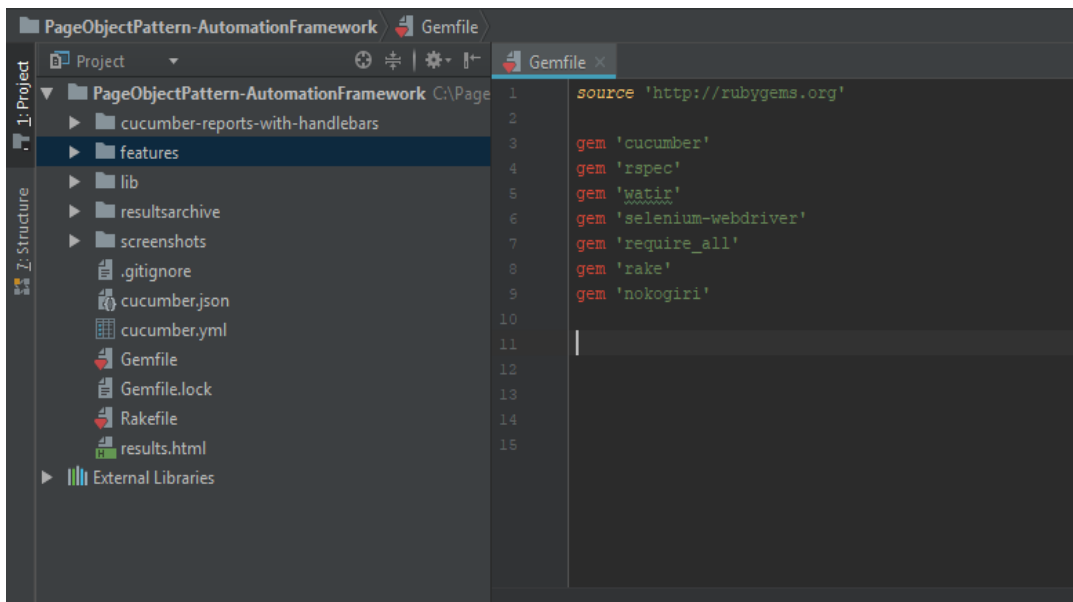
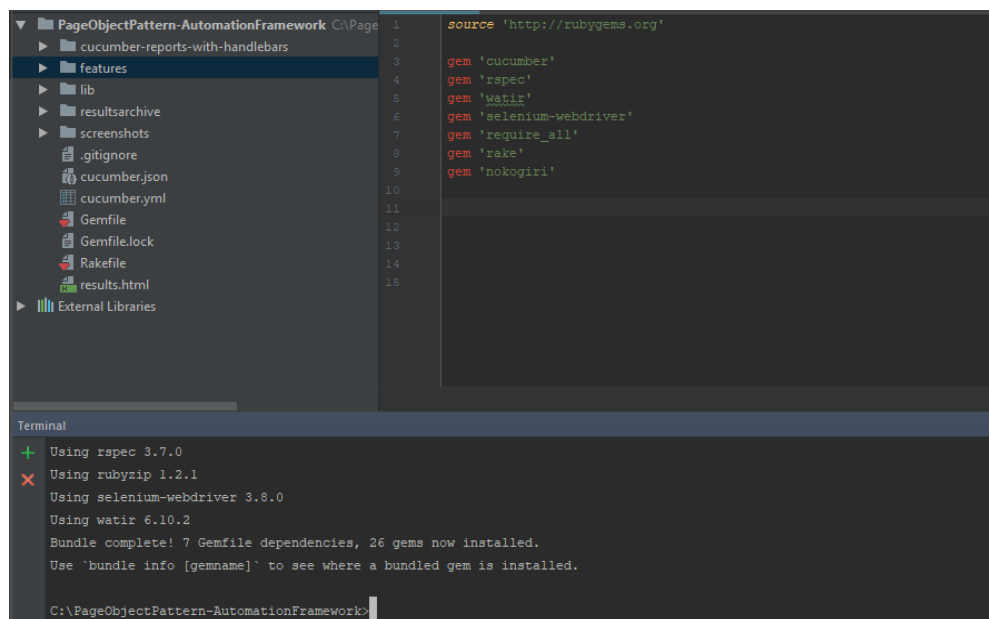


Figure 2. Page Object Framework structu

2.3 Install Cucumber and Selenium Webdriver

Open Gemfile.rb file as mentioned in Fig. 3. Install required Gems. and mentioned all required gems as described. after run command **bundle install** within RubyMine Terminal. window, it will successfully install all the required gems i.e. Cucumber and Selenium WebDriver. Now the



project is ready to implement page object pattern with open source technologies Ruby, Selenium and watir. Install drivers to run browsers by following <http://watir.com/guides/drivers/>

Figure. 3. Install required Gems

3 What is page Object pattern

3.1 Introduction context

Whenever we communicate with web pages using a internet browser and also have to have interaction through an element within the web page, we discover the element utilizing selenium's defined application interfaces and carry out an action upon it. The idea of page object is in order to contain the element initialized the moment we would like to communicate with the element, because of that reason there is no need to look into application's document object model (DOM) each time we have to communicate with element.

Architecting a page-object on synchronous applications is easier because the page-object represents the underlying html source of the web page. So for example, if the web page is fully loaded into the DOM, then our page-object [once initialized] will represent the web page and all we need to do is to call the members of the page-object and interact with it [vs. trying to locate the element before we interact with it]

3.2 Define a page object pattern

There isn't any simple solution also known as. one-size-fits-all solution in this problem. Nevertheless, I might highlight this aspect that the automation tests would operate correctly towards the magnitude about how you have build your page-objects and the interaction between the page-objects. in Nutshell there should be balance on how page object pattern should be define - page object pattern should able to capture the application behavior as much as possible. page object pattern should intuitively make sense to user. As well page object pattern should able to align with asynchronous calls like Ajax.

3.2 Establish a page object pattern

Considering that all webpages involve some resemblances, why don't we establish that inside a

Basic Foundation Page. We develop this within 'lib/pages/basic_foundation_page

```
class BasicFoundationPage

  def initialize browser, visit = false
    @browser = browser
    goto if visit
  end

  def method_missing sym, *args, &block
    @browser.send sym, *args, &block
  end

  def self.page_url url
    define_method 'goto' do
      @browser.goto url
    end
  end

  def self.element element_name
    define_method element_name.to_s do
      yield self
    end
  end
end
```

Figure. 4. Basic Foundation Page Class and methods

As per Fig. 4. there are very few methods defined within class. Now instead of declaring method for every element in a page-object, let's have a very generic way to define an element.

```
def self.element element_name
  define_method element_name.to_s do
    yield self
  end
end
```

The initialize method as you can see connects to the “visit” method. All it goes is assign the browser instance to @browser when the constructor is called.

```
def initialize browser, visit = false
  @browser = browser
  goto if visit
end
```

Page_url takes url as argument and just calls browser.goto

```
def self.page_url url
  define_method 'goto' do
    @browser.goto url
  end
end
```

To implement page object pattern, divide application pages in logical sequence which make sense, remember there is no right or wrong way, only catch is to make balance between too much abstraction and too little abstraction. this paper is using <https://www.phptravels.net/> site for demo purposes of how to define page object pattern. consider below as landing page and define required objects under lib\pages\landing_page.rb

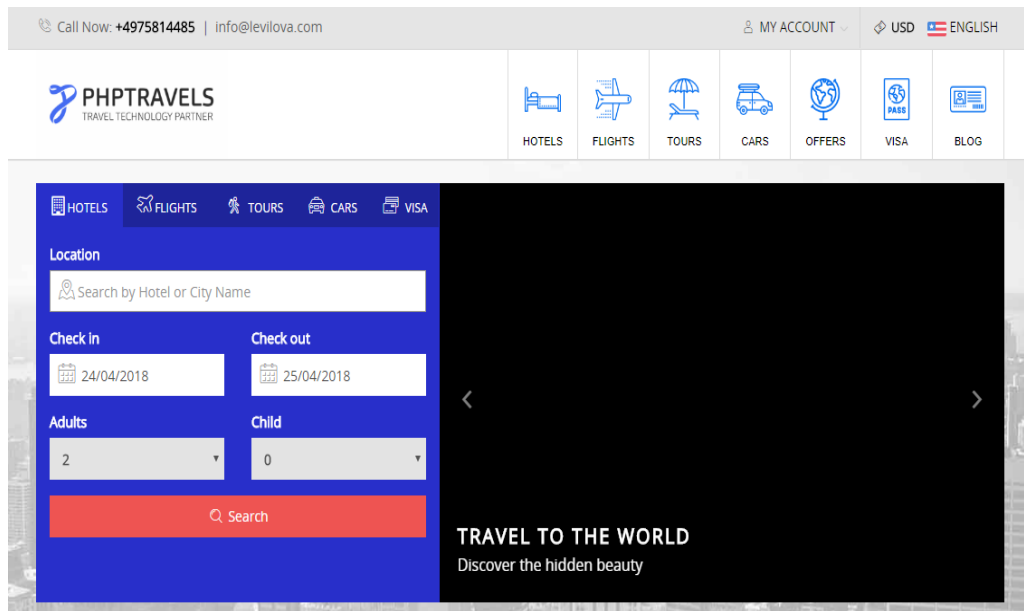


Figure. 5. Landing Page

```

class LandingPage < BasicFoundationPage
  page_url " https://www.phptravels.net "

  element(:my_account) {|b| b.link(text: "My Account")}
  element(:hotels) {|b| b.link(text: "Hotels")}
  element(:flights) {|b| b.link(text: "Flights")}

```

```

class MiddlePage < BasicFoundationPage
  element(:check_in) { |b| b.text_field(name: 'Check in')}
  element(:check_out) {|b|b.text_field(title: 'Check Out')}
  element(:adults) {|b|b.select_list(id: 'adults')}
end
end

```

Similar way define objects and elements of Login page from link <https://www.phptravels.net/login>.

```

class LoginPage < BasicFoundationPage

  element(:email) {|b| b.text_field(name: "email")}
  element(:password) {|b| b.text_field(name: "passwd")}
  element(:login_button) {|b|b.button(text: "Login")}
  deflogin(username="automationphptravels@gmail.com",pa  passwd="kevinsmith")
    email.set username
    password.set passwd
    signin_button.click
  end
end

```

Now defined objects in page object pattern can be used within step definitions easily to drive feature defined as a test scenario within feature file, that is out of scope in this paper.

4. Conclusion

Page Object pattern is industry recognize de facto standard to build sustain and maintainable automation framework and to define application logical navigation methodologies within framework. This methodology is easy to understand and maintain for longer period of time. page Object pattern very well establish with synchronous applications, where application span across multiple pages and modules.

There is usually a false impression on the Test Automation community that Page Object framework is miracle and can resolve all issues and that is the solution for all browser automation problems. Page Objects characterize the html document origin comparatively very well, nevertheless they have their very own restrictions on comprising the page navigation and DOM recent state. Asynchronous calls and Ajax calls need more attention while implementing page object pattern on web GUI.

Reference

- [1] <https://github.com/SeleniumHQ/selenium/wiki/PageObjects>
- [2] <https://github.com/SeleniumHQ/selenium/wiki/PageFactory>
- [3] https://en.wikipedia.org/wiki/Test_automation
- [4] <http://watir.com/guides/installation/>
- [5] <http://watir.com/guides/ruby/>
- [6] <http://watir.com/guides/drivers/>
- [7] <https://cucumber.io/>