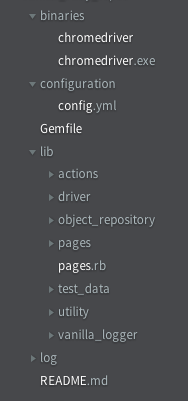
Development GUIDE

FOR

Vanilla Ruby Automation Framework

This document provides details of the Vanilla Ruby Test Automation framework developed for testing Web Application.

**Framework Structure**



Brief Explanation for the folders

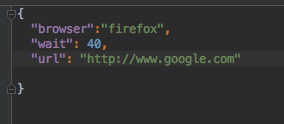
* **binaries** contains the chrome\_driver binaries for Windows and MAC
* **configuration** contains the config.yml file in which the runtime configuration is set
* **Gemfile** contains the configuration of dependent / required gems for the project
* **lib**
  + **actions** contains the web driver actions
  + **driver** contains the initialization of webdriver
  + **object\_repository** contains the element and respective locators in json or csv file
  + **pages** :: user has flexibity to use framework as pae object model. Can put the respective screens with desired logic
  + **test\_data** :: User can put the Test Data here and has to put the working logic to read test data in utility ( As the format of test data can be drive as per user knowledge )
  + **utility** contains the utility files (eg. file\_reader)
  + **vanilla\_logger** an individual logger module for Vanilla framework

**How to Setup Configuration for the vanilla project**

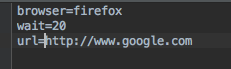
* Navigate to project directory
* Open the folder configuration
* open file config.yml **or** config.json **or** config.csv
  + config.yml



* + config.json



* + config.csv

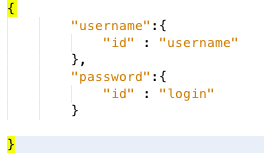


* browser : take parameter to open the browser parameters are
  + firefox
  + chrome ( chrome\_driver path should be set in environment variable )
* wait : takes the wait time till the webdriver will for the element in web page
* url : takes the web URL ( http:// & https:// strictly not required )

**How to Setup “object\_repository” for the vanilla project**

**Object repository contains the web element using specific format using**

* Navigate to project directory
* Navigate to object\_repository directory inside project directory, Either json or csv file can be used for putting the page element
* File name should be **“object\_repository.json” or “object\_repository.csv”**
  + json : file example below



* + csv : file example below



**How to use logger in vanilla ruby project**

To use logger in Vanilla one can simply define the log levels to get the Log information

**Info ::** $LOG.info “your comment”



**Debug** :: $LOG.error “your comment”



**error** :: $LOG.error “your comment”



**How to use improvize / customoize selenium action**

Make an object of action class and use the default improvize / customoize method

eg:: test = Vanilla::PageAction.new(driver)

test.forward

Improvize / customize methods

* **get\_url**

*# get\_url :: It opens the browser defined in config file and performs sending the URL and executing it on open web browser*

*# Input : URL String eg.* ***www.google.com***

*# It can be either passed using configuration file or using the default parameter*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.get\_url("http://www.google.com")*

* **forward**

*# forward performs sending forward action to opened browser button*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.forward*

* **back**

*# back performs sending backward action to opened browser button*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.back*

* close\_browser

*# close\_browser :: closes all browser windows, and quit the selenium session*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.close\_browser*

* close\_window

*# close\_window :: closes the current browser windows, selenium session is still active*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.close\_window*

* wait\_and\_find\_element

*# wait\_and\_find\_element: will wait for element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# O/P :: return true* ***if*** *element is appears in webpage* ***else*** *it will throw error*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.wait\_and\_find\_element(element\_name)*

* element\_enabled

*# element\_enabled : It verifies the element is enabled on webpage*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# O/P :: return true* ***if*** *element is enabled to perform action in webpage* ***else*** *it will throw error*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.element\_enabled(element\_name)*

* click\_on\_element

*# click\_on\_element :: It click on defined element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# O/P :: It clicks on element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.click\_on\_element(element\_name)*

* delete\_all\_cookies

*# delete\_all\_cookies :: It delete all cookies from browser*

*# O/P :: delete all cookies*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.delete\_all\_cookies*

* delete\_cookie

*# delete\_cookie :: Delete particular cookie in Selenium session browser*

*# I/P :: parameter should be the name of cookie to be deleted*

*# O/P :: It deletes particular from browser*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.delete\_cookie(cookie\_name)*

* add\_cookie

*# add\_cookie :: add cookie in browser session*

*# I/P :: (key,value) cookie key and value is added to brower session*

*# O/P :: adds the cookie to browser session*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.add\_cookie(key,value)*

* all\_cookies

*# all\_cookies :: get all cookies in current session*

*# O/P :: Returns the array of cookies*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.all\_cookies*

* drag\_and\_drop

*# drag\_and\_drop :: It drags the element from current location to target location*

*# I/P :: parameter(source\_element) :: element name, Defined in object repository (csv or json),*

*# parameter(target\_element) :: element name, Defined in object repository (csv or json)*

*# O/P :: drags the element name*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.drag\_and\_drop(source\_element, target\_element)*

* get\_text

*# get\_text :: ets the text value of the current element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# O/P :: gets the text from the element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.get\_text(element\_name)*

* clear\_text

*# clear\_text :: clear value of text field or text area element name and text value*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: clear value of text field or text area*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.clear\_text(element\_name)*

* send\_keys\_text

*# send\_keys\_text :: sends the text value of text field or text area element name*

*# Input :: parameter(element name) defined in JSON / CSV file*

*# parameter(text) send the text value in text field and text area*

*# Output :: type the text value in text field and text area*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.send\_keys\_text(element\_type, text)*

* element\_location

*# element\_location(element\_type) :: gets the element location*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: Gets the element location*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.element\_location(element\_name)*

* location\_scroll

*# location\_scroll(element\_type) :: scroll to element location*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: scroll to location*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.location\_scroll(element\_name)*

* element\_size

*# element\_size :: get the size of element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: Gets the size of element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.element\_size(element\_name)*

* element\_displayed

*# element\_displayed :: The element is displayed on web page*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: return element is displayed on web page*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.element\_displayed(element\_name)*

* element\_attribute

*# element\_attribute :: Get the attribute of element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: return the attribute of element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.element\_attribute(element\_name, attribute\_type)*

* value

*# value :: Get the value of element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# O/P :: Get the element value*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.value(element\_name)*

* move\_to

*# move\_to element\_name, right\_by = nil, down\_by = nil :: move the element from one location to other*

*# Input :: Element\_name => element name defined in JSON / CSV file, right\_by => value to shift right, down\_by => value to move*

*# Output :: Move the element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.move\_to(element\_name, right\_by = nil, down\_by = nil)*

* double\_click

*# double\_click :: double clicks on element*

*# I/P :: Parameter is defined "element name" in csv or json file*

*# Output :: double clicks on element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.double\_click(element\_name)*

* select

*# select:: Select the element, can be used with checkbox or radio*

*# Input :: Element\_name => element name defined in JSON / CSV file*

*# Output :: select the element*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.select(element\_name)*

* selected?

*# selected?(element\_name) :: verify elememnt is selected or not*

*# Input :: Element\_name => element name defined in JSON / CSV file*

*# Output :: true or false*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.selected?(element\_name)*

* checkbox\_checked

*# checkbox\_checked(element\_name, be\_selected=true) :: selecting and verifying check box is selected*

*# Input :: Element\_name => element name defined in JSON / CSV file*

*# Output :: true or false*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.checkbox\_checked(element\_name, be\_selected=true)*

* select\_option

*# Usage ::*

*# element\_name = company\_dropdown*

*# dropdown should be selected by "value"*

*# value is "afour"*

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.select\_option('company\_dropdown', :value, 'afour')*

* refresh

*# refresh :: it refresh current web page*

*# O/P :: refresh current web page, with current URL*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.get\_text(element\_name)*

* switch\_frame

*# switch\_frame :: it switch the frame to given frame id or index*

*# I/P :: parameter(frame\_name) : it requires the frame name or frame id*

*# O/P :: Switch to the given frame* ***else*** *return the error message*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.switch\_frame(frame\_name)*

* switch\_default

*# switch\_default :: switches back to default frame*

*# O/P :: Switch to the default frame*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.switch\_default*

* title

*# title :: It verify the title*

*# I/P :: it takes input as expected title*

*# O/P :: return true* ***if*** *page title matches with expected title* ***else*** *return false*

*# Information :: This method uses a custom made rspec matchers*

*# usage :: create the object of PageAction* ***class***

*# $browser = Vanilla::PageAction****.new****(driver)*

*# $browser.title(expected\_title)*

**How to use default selenium webdriver action**

using the global variable **$driver** default selenium browser action can be used.

eg. $driver.find\_element(css: , “css\_string”)