6. Page Objects

This chapter is a tutorial introduction to page objects design pattern. A page object represents an area in the web application user interface that your test is interacting.

Benefits of using page object pattern:

* Creating reusable code that can be shared across multiple test cases
* Reducing the amount of duplicated code
* If the user interface changes, the fix needs changes in only one place

6.1. Test case

Here is a test case which searches for a word in python.org website and ensure some results are found.

**import** **unittest**

**from** **selenium** **import** webdriver

**import** **page**

**class** **PythonOrgSearch**(unittest.TestCase):

*"""A sample test class to show how page object works"""*

**def** setUp(self):

self.driver = webdriver.Firefox()

self.driver.get("http://www.python.org")

**def** test\_search\_in\_python\_org(self):

*"""*

*Tests python.org search feature. Searches for the word "pycon" then verified that some results show up.*

*Note that it does not look for any particular text in search results page. This test verifies that*

*the results were not empty.*

*"""*

*#Load the main page. In this case the home page of Python.org.*

main\_page = page.MainPage(self.driver)

*#Checks if the word "Python" is in title*

**assert** main\_page.is\_title\_matches(), "python.org title doesn't match."

*#Sets the text of search textbox to "pycon"*

main\_page.search\_text\_element = "pycon"

main\_page.click\_go\_button()

search\_results\_page = page.SearchResultsPage(self.driver)

*#Verifies that the results page is not empty*

**assert** search\_results\_page.is\_results\_found(), "No results found."

**def** tearDown(self):

self.driver.close()

**if** \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

6.2. Page object classes

The page object pattern intends creating an object for each web page. By following this technique a layer of separation between the test code and technical implementation is created.

The page.py will look like this:

**from** **element** **import** BasePageElement

**from** **locators** **import** MainPageLocators

**class** **SearchTextElement**(BasePageElement):

*"""This class gets the search text from the specified locator"""*

*#The locator for search box where search string is entered*

locator = 'q'

**class** **BasePage**(object):

*"""Base class to initialize the base page that will be called from all pages"""*

**def** \_\_init\_\_(self, driver):

self.driver = driver

**class** **MainPage**(BasePage):

*"""Home page action methods come here. I.e. Python.org"""*

*#Declares a variable that will contain the retrieved text*

search\_text\_element = SearchTextElement()

**def** is\_title\_matches(self):

*"""Verifies that the hardcoded text "Python" appears in page title"""*

**return** "Python" **in** self.driver.title

**def** click\_go\_button(self):

*"""Triggers the search"""*

element = self.driver.find\_element(\*MainPageLocators.GO\_BUTTON)

element.click()

**class** **SearchResultsPage**(BasePage):

*"""Search results page action methods come here"""*

**def** is\_results\_found(self):

*# Probably should search for this text in the specific page*

*# element, but as for now it works fine*

**return** "No results found." **not** **in** self.driver.page\_source

6.3. Page elements

The element.py will look like this:

**from** **selenium.webdriver.support.ui** **import** WebDriverWait

**class** **BasePageElement**(object):

*"""Base page class that is initialized on every page object class."""*

**def** \_\_set\_\_(self, obj, value):

*"""Sets the text to the value supplied"""*

driver = obj.driver

WebDriverWait(driver, 100).until(

**lambda** driver: driver.find\_element\_by\_name(self.locator))

driver.find\_element\_by\_name(self.locator).clear()

driver.find\_element\_by\_name(self.locator).send\_keys(value)

**def** \_\_get\_\_(self, obj, owner):

*"""Gets the text of the specified object"""*

driver = obj.driver

WebDriverWait(driver, 100).until(

**lambda** driver: driver.find\_element\_by\_name(self.locator))

element = driver.find\_element\_by\_name(self.locator)

**return** element.get\_attribute("value")

6.4. Locators

One of the practices is to separate the locator strings from the place where they are being used. In this example, locators of the same page belong to same class.

The locators.py will look like this:

**from** **selenium.webdriver.common.by** **import** By

**class** **MainPageLocators**(object):

*"""A class for main page locators. All main page locators should come here"""*

GO\_BUTTON = (By.ID, 'submit')

**class** **SearchResultsPageLocators**(object):

*"""A class for search results locators. All search results locators should come here"""*

**pass**