accenture technology

Learning and Knowledge Management

Module 4: Selenium Python API

Module Objectives

At the end of this module, you will be able to:

- Explain working with Selenium WebDriver API
- Identify web elements using Selenium locator methods
- Implement Selenium commands such as WebDriver, WebElement, Wait, Select
- Discuss commands for mouse and keyboard actions

Topic List

- WebDriver API
- Selenium Locators
- WebDriver Commands
- WebElement Commands
- Wait Commands
- Select Commands
- Keyboard and Mouse Actions

Topic List

- **WebDriver API**
- Selenium Locators
- WebDriver Commands
- WebElement Commands
- Wait Commands
- Select Commands
- Keyboard and Mouse Actions

WebDriver API (1)

About WebDriver API?

- Provides simple, user-friendly and a crisp programming interface
- Supports automation testing for dynamic web pages in web applications

WebDriver API Import: The keyword and an example snippet to import Selenium Python binding is shown below:

Keyword	Example
from Source import	*from selenium import webdriver* *from selenium.webdriver.common.by import By* *from selenium.webdriver.support.ui import WebDriverWait*

Note: WebDriver API overcomes some limitations of Selenium-RC API:

- It makes direct calls to the browser using browser's native support
- WebDriver functions as normal library, is self-contained, and does not require starting any additional processes or runing installers or proxy server like in case of Selenium-RC.

WebDriver API (2)

Conventions

WebDriver API conventions	
Convention	**Example**
WebDriver methods delimits with round brackets	driver.close() This method closes a WebDriver instance
WebDriver properties do not end with round brackets	driver.current_url Fetches the current URL in a WebDriver instance

Topic List

- WebDriver API
- **Selenium Locators**
- WebDriver Commands
- WebElement Commands
- Wait Commands
- Select Commands
- Keyboard and Mouse Actions

Selenium Locators (1)

Locating Elements

- Selenium WebDriver API provides 8 strategies to identify/locate web elements in a web page as mentioned below:
  - Id
  - Name
  - Class Name
  - Link Text
  - Partial Link Text
  - Tag Name
  - CSS locators
  - XPath

Selenium Locators (2)

Selenium Web Elements and Locator Methods

- Web elements are visual entities (also called controls) that are used in a web page
  - Example: Textboxes, buttons, tables, forms, lists, images etc.,
- Selenium locator methods locate/fetch a single or set of web elements in a web page, thus enables to automate browser and simulate user actions

Fetch a single web element	Fetch list of elements
• *find_element_by_id* • *find_element_by_name* • *find_element_by_xpath* • *find_element_by_link_text* • *find_element_by_partial_link_text* • *find_element_by_tag_name* • *find_element_by_class_name* • *find_element_by_css_selector*	• *find_elements_by_name* • *find_elements_by_xpath* • *find_elements_by_link_text* • *find_elements_by_partial_link_text* • *find_elements_by_tag_name* • *find_elements_by_class_name* • *find_elements_by_css_selector*

Selenium Locators (3)

Examples

Locator Parameter	Sample Code	Selenium Locator Method
• Locating by id The first element with the given id attribute value will be returned.	<html> <body> <form id="UserForm"> <input name="username" type="text" /> <input name="password" type="password" /> </form> </body> <html>	The form element could be located: form_user = driver.find_element_by_id('UserForm')
• Locating by Name The first element with the given name attribute value will be returned.		The username element could be located: user_id = driver.find_element_by_name('username')

Selenium Locators (4)

Examples

Locator Parameter	Sample Code	Selenium Locator Method
Locating by Xpath • Used when a suitable id or name attribute for the element not available to locate. • Used to locate the element in absolute terms starting from the root or relative to an element that contains an id or name attribute.	<html> <body> <form id="UserForm"> <input name="username" type="text" /> <input name="password" type="password" /> </form> </body> <html>	The username element could be located: • *To fetch the First **form** element, with child element **input,** with attribute **name** and value **username*** *User_id = driver.find_element_by_xpath("//form[input/@name='username']")* • *To fetch First **input** child element of the **form** element with attribute named **id** and the value **UserForm*** *User_id = driver.find_element_by_xpath("//form[@id='UserForm']/input[1]")* • *To fetch First input element with attribute name and the value username* *User_id = driver.find_element_by_xpath("//input[@name='username']")*

Selenium Locators (5)

Examples

Locator Parameter	Sample Code	Selenium Locator Method-Explanation
• Locating by Link Text The first element with the link text value used within an anchor tag will be returned.	*<html> <body> Do you wish to continue? Continue Cancel </body> <html>*	The NextPage.html link element could be located: • *Next_Page = driver.find_element_by_link_text('Continue')* • *Next_Page = driver.find_element_by_partial_link_text('Cont')*
• Locating by Tag Name The first element with the given tag name will be returned	*<html> <body> Welcome to Test School Testing content follows. </body> <html>*	The h1 Heading tag element could be located: • Heading_Tag = driver.find_element_by_tag_name('h1')

Selenium Locators (6)

Examples

Locator Parameter	Sample Code	Selenium Locator Method-Explanation
• Locating by Class Name The first element with the given Class attribute name will be returned	*<html> <body> Testing Content. </body> <html>*	The Class element could be located: • *Details_Class = driver.find_element_by_class_name('TestDetails')*
• Locating by CSS Selectors The first element with the matching CSS selector will be returned	*<html> <body> Testing Content. </body> <html>*	The Class element could be located: • *Details_CSS = driver.find_element_by_css_selector ('p.TestDetails')*

Selenium Locators (7)

Locating Element By CSS Selectors

- **By using ID selector**

  CSS Syntax : css=tag#id

  Python Syntax : *driver.find_element_by_css_selector("input#policy_no")*

- **By using Class selector**

  CSS Syntax : *css=tag.class*

  Python Syntax : *driver.find_element_by_css_selector("input.policy_no")*

- **By using Attributes selector**

  CSS Syntax : *css=tag[attribute=value][attribute=value]*

  Python Syntax : *driver.find_element_by_css_selector("input[value='oneway']")*

Selenium Locators (8)

Locating Element By CSS Selectors Contd..

By performing partial match on attribute values

Syntax	Example	Description
^=	input[id^='ctrl']	**Starting with:** For example, if the ID of an element is ctrl_12, this will locate and return elements with ctrl at the beginning of the ID.
$=	input[id$='_userName']	**Ending with:** For example, if the ID for an element is a_1_userName, this will locate and return elements with _userName at the end of the ID.
*=	input[id*='userName']	**Containing:** For example, if the ID of an element is panel_login_userName_textfield, this will use the userName part in the middle to match and locate the element.

Selenium Locators (9)

Locating Element By CSS Selectors Contd..

- **Location by DOM structure using Absolute Path**

  CSS absolute path refer to the very specific location of the element considering its complete hierarchy in the DOM.

  Example: *css=html body div div form input*

  Parent to child relationships with **> or space** separator

  Example: *css=html>body>div>div>form>input*

  Child to child relationships with **+** separator

  Example: *css=html>body>div>div>form>input+input+input*

- **Locating by DOM structure using Relative Path**

  With relative path we can locate an element directly, irrespective of its location in the DOM.

  Example: *css=form[id='f1']>div>input*

- **Finding the nth child element**

  Example: *css=div#_eEe *:nth-child(3)* – this example is for identifying "Telugu" link in Google Home page.

Selenium Locators (10)

Locating Element By XPath

- XPath, the XML path language is a query language for selecting nodes from an XML document. All the major browsers support XPath HTML pages are represented as XHTML documents in DOM.
- Selenium supports XPath for locating elements using XPath expressions or queries.
- XPath provides lot of flexibility in locating elements but with slow performance. This drawback makes using Xpath the least preferable locator strategy

Note: Important difference between XPath and CSS

- With Xpath, elements can be searched both backward or forward in the DOM hierarchy, which makes it possible to locate a parent element using a child element
- CSS works only in a forward direction

Selenium Locators (11)

Locating Element By XPath Contd..

- **Finding elements with absolute path**

  Example: xpath=/html/body/form

- **Finding elements using index**

  Example: xpath=//form/input[1]

- **Finding elements using attributes values**

  Example: xpath= //input[@name='username']

  With multiple attributes:

  Example: xpath=//input[@name='username'][@class='required']

  Example: xpath=//input[@name='username' and @class='required']

  Example: xpath=//input[@name='username' or @class='required']

Selenium Locators (12)

Locating Element By XPath Contd..

- **Performing partial match on attribute values**

Syntax	Example	Description
starts-with()	xpath=//input[starts-with(@id,'ctrl')]	**Starting with:** For example, if the ID of an element is ctrl_12, this will locate and return elements with ctrl at the beginning of the ID.
ends-with()	xpath=//input[ends-with(@id,'_userName')]	**Ending with:** For example, if the ID of an element is a_1_userName, this will locate and return elements with _userName at the end of the ID
contains()	xpath=//input[contains(@id,'userName')]	**Containing:** For example, if the ID for an element is panel_login_userName_textfield, this will use the userName part in the middle to match and locate the element.

Selenium Locators (13)

Locating Element By XPath Contd..

- **Matching any attribute or any element using a value**

  Example: xpath=//input[@*='username']

  Example: xpath=//*[@name='username']

- **Using Xpath text function**

  Example: xpath= //span[contains(text(),'some text')]

Exercise 4.1: Identifying Elements

Locate the web elements in 'TestMeApp' through the Tomcat Manager app and identify elements

To Do

1. Open TestMe app http://localhost:8083/TestMeApp/ through Tomcat Manager app
2. Inspect the WebPage and note down the various WebElements
3. Write down Python module Selenium WebDriver scripts to
    - Open TestMe app
    - Login to the site using valid credentials
    - Print the username of the Logged In user on console
    - Logout
    - Close browser
4. Run the Python module

Refer Exercise 4.1 in Selenium with Python_wbk.doc for the detailed steps.

Topic List

- WebDriver API
- Selenium Locators
- **WebDriver Commands**
- WebElement Commands
- Wait Commands
- Select Commands
- Keyboard and Mouse Actions

WebDriver Commands

Instructions for Actions on Drivers

- **WebDriver** commands instructions actions to be performed on specified or Stored drivers

WebDriver Command_Module 5.txt

Note: Please refer to http://selenium-python.readthedocs.io/ for complete list of WebDriver commands used with Python

- ***driver.name***

Will return the name of the browser instance used

- ***driver.get(url)***

will open the given url in the WebDriver instance

- ***driver.implicitly_wait(sec)***

Will wait for the predefined number of secs before proceeding with next step

- ***driver.maximize_window()***

Will maximize the window of the current instance

- ***driver.back()***

Will move back to the previous webpage in the instance

- ***driver.title***

Will Return the title of the webPage

- ***driver.get_screenshot_as_file('path/file')***

Will Take screenshot of the current webpage and save it to a file

- ***driver.forward()***

Will move forward to the next webpage in the instance

- ***driver.switch_to.alert.dismiss()***

Will shift focus to the alert message and perform cancel action

- ***driver.refresh()***

Will refresh the current webpage

- ***driver.close()***

Will close the current WebDriver instance

Topic List

- WebDriver API
- Selenium Locators
- WebDriver Commands
- **WebElement Commands**
- Wait Commands
- Select Commands
- Keyboard and Mouse Actions

WebElement Commands (1)

Instructions for Actions on Web Elements

WebElement commands instructions actions to be performed on specified or Stored web elementsConsider:

Consider the line below:

Demo_WebElement = driver.find_element_by_name("DemoElement")

Following actions can be performed for the element, Demo_WebElement

- **Demo_WebElement.sendKeys(“Demo_text”)**

If the Demo_WebElement is a Text Box, then this command will enter the value Demo_text inside the Text Box

- **Demo_WebElement.clear()**

If the Demo_WebElement is a Text Box which is filled with some value, then this command will clear the values inside the Text Box

- **Demo_WebElement.click()**

This command will perform the click action on the Demo_WebElement (Like in case of Command button, radio buttons,Checkboxes, etc.)

- **Demo_WebElement.isEnabled()**

This command will check whether the Demo_WebElement is currently Enabled and accordingly will return a boolean

WebElement Commands (2)

Instructions for Actions on Web Elements

More actions can be performed for the web element 'Demo_WebElement'

- ***Demo_WebElement.isDisplayed()***

This command will check whether the Demo_WebElement is currently Displayed and accordingly will return a boolen

- ***Demo_WebElement.isSelected()***

This command will check whether the Demo_WebElement is currently Selected (Like in case of radio buttons,Checkboxes, etc.) and accordingly will return a boolen

- ***Demo_WebElement.submit()***

This command will perform the Submit action on the Demo_WebElement (Like in case of Command buttons, forms etc.)

- ***Demo_WebElement.getText()***

This command will obtain the innertext value of the Demo_WebElement

- ***Demo_WebElement.getTagName()***

This command will obtain the associated Tag Name of the Demo_WebElement

WebElement Commands (3)

Instructions for Actions on Web Elements

More actions can be performed for the web element ‘Demo_WebElement’

- ***Demo_WebElement.getCssValue()***

This command will Return the CSS property value of the Demo_WebElement

- ***Demo_WebElement.getSize()***

This command will Return the size in terms of Height, width of the Demo_WebElement

- ***Demo_WebElement.getLocation()***

This command will Return the location point size in terms of X,Y corodinates of the Demo_WebElement

Topic List

- WebDriver API
- Selenium Locators
- WebDriver Commands
- WebElement Commands
- **Wait Commands**
- Select Commands
- Keyboard and Mouse Actions

Wait Commands (1)

Commands to Handle Delay

ElementNotVisibleException exception: The exception that occurs when the Selenium WebDriver is not able to locate web elements in a page

- The reason for the exception is the delay with browsers loading a web page or loading web elements in a page
- Following are the 2 types of Selenium WebDriver wait commands to handle the exception:

Type of wait command	Explanation
Explicit wait	Makes WebDriver to wait for a specific condition to occur before proceeding further with next step of execution
Implicit wait	Makes WebDriver wait & constantly check the page to locate the desired element, for the specified amount of time.

Wait Commands (2)

Wait import

- Importing WebDriver wait can be done as follows

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected_conditions as EC

Example

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

from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

driver = webdriver.Chrome()

driver.get("http://localhost:8083/TestMeApp/")
driver.find_element_by_link_text("SignIn").click()

driver.implicitly_wait(5000)

driver.find_element_by_id("userName").send_keys("Lalitha")
driver.find_element_by_name("password").send_keys("Password123")
driver.find_element_by_xpath('//form/fieldset/div[4]/div/input[1]').click()

WebDriverWait(driver, 500000).until(EC.presence_of_element_located((By.CLASS_NAME, "nav")))

print driver.find_element_by_class_name("nav").text
driver.find_element_by_link_text("SignOut").click()
driver.close()
```

Topic List

- WebDriver API
- Selenium Locators
- WebDriver Commands
- WebElement Commands
- Wait Commands
- **Select Commands**
- Keyboard and Mouse Actions

Select Commands

Methods and Properties for User Interaction

The Selenium Select class provides methods and properties for drop-down or lists' user interactions

- Importing Select is required:

from selenium.webdriver.support.ui import Select

Some Select class methods

- ***select_by_index(index)***

Selects an option from the dropdown or list at the given index

- ***select_by_visible_text(text)***

Selects the options from the dropdown or list that display the text matching the text

- ***select_by_value(value)***

Selects the options from the dropdown or list which contains a value matching the argument

- ***deselect_all()***

Clears from a multiselect dropdown or list all the selected entries

Exercise 4.2: Using Select Command

Scenario: Use Select Command to select Paris city in http://newtours.demoaut.com/

To Do :

1. Write down Python module Selenium WebDriver scripts to
    - Open newtours website http://newtours.demoaut.com/
    - Login using valid credentials
    - Choose number of passengers as 3
    - Choose Departing city as Paris
    - Choose a date from next month
    - Choose Arriving city as London
    - Click on Continue
    - Logout
    - Close browser
2. Run the Python module

Refer Exercise 4.2 in Selenium with Python_wbk.doc for the detailed steps.

Topic List

- WebDriver API
- Selenium Locators
- WebDriver Commands
- WebElement Commands
- Wait Commands
- Select Commands
- **Keyboard and Mouse Actions**

Keyboard and Mouse Actions (1)

Simulate Keyboard and Mouse Events

WebDriver API enables simulation of keyboard and mouse actions using the **ActionChains** class

- Importing ActionChains is required:

from selenium.webdriver.common.action_chains import ActionChains

from selenium.webdriver.common.keys import Keys

Some keyboard controls

- ***key_down(Keys.value)***

Simulates a Key Press without releasing it. To be used with modifier keys (such as the Ctrl, Alt, and Shift keys)

- ***key_up(Keys.value)***

Releases a pressed key. To be used with modifier keys (such as the Ctrl, Alt, and Shift keys)

- ***send_keys("String value")***

Sends the String value to the element that has current focus

- ***send_keys_to_element(element, "String value")***

Sends the String value to a specific element

Keyboard and Mouse Actions (2)

Simulate Keyboard and Mouse Events

Some mouse actions

- ***click(on_element)***

Performs the click operation on_element. If None, clicks on the current mouse position

- ***double_click (on_element)***

Performs the double-click operation on an element. If None, clicks on the current mouse position

- ***click_and_hold (on_element)***

Simulates holding down left mouse button operation on an element

- ***release (on_element)***

Releases a held mouse button

- ***drag_and_drop(source, target)***

Performs the drag-and-drop operation of an element. At Source it will simulate mouse down and at Target it will simulate mouse up

- ***perform()***

Performs the stored actions on an element

Keyboard and Mouse Actions (3)

Mouse Action-Example

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.action_chains import ActionChains


 driver = webdriver.Chrome()
 driver.get("http://jqueryui.com/resources/demos/droppable/default.html")
 driver.implicitly_wait(5000)
driver.maximize_window()

SourceElement = driver.find_element_by_id("draggable");
TargetElement = driver.find_element_by_id("droppable");

 ActionChains(driver).drag_and_drop(SourceElement, TargetElement).perform()
```


Module Summary

Now, you should be able to:

- Explain working with Selenium WebDriver API
- Identify web elements using Selenium locator methods
- Implement Selenium commands such as WebDriver, WebElement, Wait, Select
- Discuss commands for mouse and keyboard actions

Thank You