# **Chapter 2 – WebDriver Fundamentals**

### **1. Basic WebDriver Usage**

* Three core aspects when using WebDriver in Python:  
  + **Creation** – instantiate driver objects for different browsers.
  + **Operations** – interact with pages and elements.
  + **Disposal** – close browser sessions after tests.

### **2. WebDriver Creation**

#### **a) Direct Instantiation (Basic Way)**

* Import and create driver for specific browsers:

from selenium import webdriver

# Chrome

driver = webdriver.Chrome()

# Firefox

driver = webdriver.Firefox()

# Edge

driver = webdriver.Edge()

#### **b) WebDriver Builder Equivalent in Python**

* Python bindings don’t have the RemoteWebDriver.builder() API like Java.
* Instead, Selenium Python relies on **Options** classes and webdriver.Remote for remote sessions.

Example – Remote WebDriver with Chrome:

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

options = Options()

options.add\_argument("--headless") # Example capability

driver = webdriver.Remote(

command\_executor="http://localhost:4444/wd/hub",

options=options

)

#### **c) WebDriver Manager (Recommended for Python)**

* Use webdriver\_manager package for automatic driver handling.

Example – Chrome:

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from webdriver\_manager.chrome import ChromeDriverManager

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

Example – Firefox:

from selenium.webdriver.firefox.service import Service

from webdriver\_manager.firefox import GeckoDriverManager

driver = webdriver.Firefox(service=Service(GeckoDriverManager().install()))

### **3. WebDriver Methods (Core API in Python)**

* **Navigation & Page Info**

driver.get("https://example.com") # Open URL

print(driver.current\_url) # Get current URL

print(driver.title) # Get page title

print(driver.page\_source) # Get page source

* **Element Location**

from selenium.webdriver.common.by import By

element = driver.find\_element(By.ID, "username") # Single element

elements = driver.find\_elements(By.CLASS\_NAME, "input") # Multiple elements

* **Window & Navigation Management**

driver.back() # Go back

driver.forward() # Go forward

driver.refresh() # Refresh page

print(driver.current\_window\_handle) # Current window handle

print(driver.window\_handles) # All window handles

* **Switch Contexts**

driver.switch\_to.frame("frame\_name") # Switch to frame

driver.switch\_to.window(driver.window\_handles[1]) # Switch window

driver.switch\_to.alert.accept() # Handle alert

* **Browser Management**

driver.maximize\_window()

driver.minimize\_window()

driver.fullscreen\_window()

driver.delete\_all\_cookies()

driver.get\_cookies()

* **Disposal**

driver.close() # Close current tab

driver.quit() # Quit entire browser session

### **4. Locating Web Elements in Python**

* Strategies available via By class:  
  + By.ID
  + By.NAME
  + By.CLASS\_NAME
  + By.TAG\_NAME
  + By.LINK\_TEXT / By.PARTIAL\_LINK\_TEXT
  + By.CSS\_SELECTOR
  + By.XPATH

Example:

element = driver.find\_element(By.XPATH, "//input[@name='q']")

### **5. User Actions (Keyboard & Mouse Simulation)**

* Use the ActionChains class:

from selenium.webdriver.common.action\_chains import ActionChains

from selenium.webdriver.common.keys import Keys

actions = ActionChains(driver)

# Mouse actions

element = driver.find\_element(By.ID, "submit")

actions.move\_to\_element(element).click().perform()

# Keyboard actions

search = driver.find\_element(By.NAME, "q")

search.send\_keys("Selenium with Python")

search.send\_keys(Keys.RETURN)

### **6. Waiting for Elements**

#### **a) Implicit Waits**

driver.implicitly\_wait(10) # Wait max 10s for elements

#### **b) Explicit Waits**

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

wait = WebDriverWait(driver, 10)

element = wait.until(EC.presence\_of\_element\_located((By.ID, "username")))

#### **c) Fluent Wait (custom polling)**

from selenium.webdriver.support.wait import WebDriverWait

wait = WebDriverWait(driver, timeout=15, poll\_frequency=2, ignored\_exceptions=[Exception])

element = wait.until(EC.element\_to\_be\_clickable((By.ID, "login")))

### **7. Session Identifier (Python Equivalent)**

* In Python, access session\_id directly from the driver:

print(driver.session\_id)

### **8. WebDriver Disposal (Python Best Practice)**

* Always ensure disposal in test teardown:

def teardown():

driver.quit()

✅ **Key Takeaways (Python)**

* Use **webdriver\_manager** to simplify driver management.
* Most Java WebDriver APIs map directly to Python methods.
* Locators + waits are the foundation of stable tests.
* Use ActionChains for complex user interactions.
* Always call driver.quit() after tests.