37. Handle Cookies and File Upload

Cookies

- ➔ A Cookie is a small piece of data that is sent from a website and stored in a computer.
- ➔ Cookies are used to recognise the user and load the stored information.
- ➔ Whenever a user browses a website, the information about the user and their favorites is stored as a cookie in the form of key-value pairs.
- ➔ When the user visits the website again, the information which was stored would be used for identification.
- ➔ A dictionary object stores all attributes of a single cookie as key-value pairs.

Methods for Cookie Handling

1. **add_cookie({"name": "cookie_name", "value": "cookie_value"})** ⇒ Adds a cookie to the browser.
2. **get_cookie("cookie_name")** ⇒ Retrieves a cookie by its name.
3. **get_cookies()** ⇒ Returns all cookies in the current browser session.
4. **delete_cookie("cookie_name")** ⇒ Deletes a specific cookie by its name.
5. **delete_all_cookies()** ⇒ Deletes all cookies in the current browser session.

HandleCookiesDemo_1.py

```
from selenium import webdriver
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
opt.add_argument("--start-maximized")
driver = webdriver.Chrome(options=opt)
driver.get("http://www.automationpractice.pl/index.php")
driver.implicitly_wait(10)
```

1.Add a cookie

```
new_cookie = {"name": "MyCookie", "value": "123456"}
driver.add_cookie(new_cookie)
print("Cookie added!")
```

2.Retrieve a specific cookie by name

```
retrieved_cookie = driver.get_cookie("MyCookie")  # We need to pass the name of the cookie
print("Retrieved cookie details:", retrieved_cookie)
```

3.Retrieve all the cookies created by the browser

```
all_cookies = driver.get_cookies()
print("Total number of cookies created:", len(all_cookies))
print("Printing all the Cookies...")
for cookie in all_cookies:
  print(cookie["name"], ":", cookie["value"])
```

4.Delete specific cookie

```
driver.delete_cookie("MyCookie")
print("Cookie 'MyCookie' is deleted..")
```

Verify the number of cookies after deletion

```
all_cookies = driver.get_cookies()
```

```
print("Number of cookies after deletion:", len(all_cookies))
```

5.Delete all the cookies

```
driver.delete_all_cookies()
```

Verify the number of cookies after deletion

```
all_cookies = driver.get_cookies()
print("Number of cookies after deletion:", len(all_cookies))  # Expected to be 0
driver.quit()
```

HandleCookies_2.py

```
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=options)
driver.get("https://www.facebook.com")
driver.maximize_window()
```

Capture Cookies from the browser

```
cookies = driver.get_cookies()
print("Size of cookies:",len(cookies))
```

Print details of all cookies

```
for c in cookies:
   print(c)
   print(c.get('name'),":",c.get('value'))
```

Add new cookie to the browser

```
driver.add_cookie({"name": "SessionID", "value": "abc123456"})
driver.add_cookie({"name": "UserID", "value": "user789"})
driver.add_cookie({"name": "AuthToken", "value": "token987654"})
driver.add_cookie({"name": "Theme", "value": "dark"})
driver.add_cookie({"name": "CartID", "value": "cart456789"})
cookies = driver.get_cookies()
print("Size of cookies after adding new one:",len(cookies))
```

Delete specific cookie from the browser

```
driver.delete_cookie("SessionID")
cookies  = driver.get_cookies()
print("Size of cookies after deleted one:",len(cookies))
```

Delete all the cookies

```
driver.delete_all_cookies()
cookies = driver.get_cookies()
print("Size of cookies after deleted all:",len(cookies))
driver.quit()
```

File Upload

- ➔ Selenium cannot interact with the file upload dialog, it provides a way to upload files without opening the dialog using send_keys().

➔ If the element is an input element with type file, you can use the send_keys() method to send the full path to the file that will be uploaded.

UploadFileDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
opt.add_argument("--start-maximized")
driver = webdriver.Chrome(options=opt)
driver.implicitly_wait(10)
driver.get("https://testautomationpractice.blogspot.com/")
file1 = "C:\\Automation\\automationFiles\\Test1.txt"
file2 = "C:\\Automation\\automationFiles\\Test2.txt"
```

Single file upload

```
driver.find_element(By.XPATH, "//input[@id='singleFileInput']").send_keys(file1)
driver.find_element(By.XPATH, "//button[normalize-space()='Upload Single File']").click()
msg = driver.find_element(By.XPATH, "//p[@id='singleFileStatus']").text
if "Test1.txt" in msg:
  print("Single file upload is successful...")
else:
  print("Upload Failed.")
```

Multiple files upload

```
driver.find_element(By.XPATH, "//input[@id='multipleFilesInput']").send_keys(f"{file1}\n{file2}")
driver.find_element(By.XPATH, "//button[normalize-space()='Upload Multiple Files']").click()
```

Count number of line breaks to estimate files uploaded

```
no_of_files_uploaded = len(driver.find_elements(By.XPATH, "//p[@id='multipleFilesStatus']/br")) - 1
print("Number of files uploaded:", no_of_files_uploaded)
msg = driver.find_element(By.XPATH, "//p[@id='multipleFilesStatus']").text
if "Test1.txt" in msg and "Test2.txt" in msg:
  print("Multiple files uploaded successfully...")
else:
  print("File names do not match. Failed to upload.")
driver.quit()
```

FileUpload_FoundIt_Demo.py

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
opt.add_argument("--start-maximized")
driver = webdriver.Chrome(options=opt)
```

```
driver.implicitly_wait(10)
driver.get("https://www.foundit.in/upload")
Click on "Upload Resume" button
driver.find_element(By.XPATH, "//div[contains(text(),'Upload Resume')]").click()
Upload the file
driver.find_element(By.XPATH,
"//input[@id='file-upload']").send_keys("C:\\Automation\\automationFiles\\Test1.txt")
Wait for the file to upload and check status
time.sleep(3)
msg = driver.find_element(By.XPATH, "//input[@name='parsingResponse']").get_attribute("value")
if "Test1.txt" in msg:
    print("File Uploaded Successfully...")
else:
    print("Failed to upload file.")
driver.quit()
```

FileUpload_FoundIt_pyautoguiDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import pyautogui, pyperclip
def upload_file(file_path):
Copy the file path to the system clipboard
    pyperclip.copy(file_path)
Use pyautogui to simulate pressing keys for file upload
    time.sleep(3)  ⇒ Wait for the dialog to open
Simulate Ctrl + V to paste the file path
    pyautogui.hotkey('ctrl', 'v')
    time.sleep(3)  ⇒ Wait after pasting the file path
Simulate pressing Enter to confirm the file selection
    pyautogui.press('enter')
    time.sleep(3)  ⇒ Wait after pressing Enter
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
opt.add_argument("--start-maximized")
driver = webdriver.Chrome(options=opt)
driver.implicitly_wait(10)
driver.get("https://www.foundit.in/upload")
Click on "Upload Resume" button
driver.find_element(By.XPATH, "//div[contains(text(),'Upload Resume')]").click()
Click on the "Select File to upload" button using JavaScript
driver.execute_script("arguments[0].click();",driver.find_element(By.XPATH,
"//input[@id='file-upload']"))
```

Call the upload_file function to upload the file

```
upload_file("C:\\Automation\\automationFiles\\Test1.txt")
```

Check the status of the file upload

```
msg = driver.find_element(By.XPATH, "//input[@name='parsingResponse']").get_attribute("value")
if "Test1" in msg:
  print("File Uploaded Successfully...")
else:
  print("Failed to upload file.")
driver.quit()
```

Note

1. **pyautogui** ⇒ Used to automate mouse movements, clicks, and keyboard actions on the screen.
2. **pyperclip** ⇒ Used to copy text to and paste text from the clipboard.

FileUploadpyautoguiUtility.py

```
import time
import pyautogui
import pyperclip
```

Method to upload a file on Windows

```
def upload_file_with_pyautogui_windows(file_path):
```

Copy the file path to the system clipboard (Ctrl+C equivalent)

```
  pyperclip.copy(file_path)
```

Wait for the file dialog to open

```
  time.sleep(3)
```

Paste the file path using Ctrl+V

```
  pyautogui.hotkey('ctrl', 'v')
  time.sleep(0.5)
```

Press Enter to confirm the file selection

```
  pyautogui.press('enter')
  time.sleep(0.5)
```

Method to upload a file on MacOS

```
def upload_file_with_pyautogui_mac(file_path):
```

Copy the file path to the system clipboard

```
  pyperclip.copy(file_path)
```

Wait for the file dialog to open

```
  time.sleep(3)
```

Use Cmd+Tab to switch focus back to the file upload dialog (if needed)

```
  pyautogui.hotkey('command', 'tab')
  time.sleep(0.5)
```

Open the "Go to Folder" window using Cmd+Shift+G

```
  pyautogui.hotkey('command', 'shift', 'g')
  time.sleep(0.5)
```

Paste the file path from the clipboard using Cmd+V

```
pyautogui.hotkey('command', 'v')
time.sleep(0.5)
```

Press Enter to confirm the file path and close the dialogs

```
pyautogui.press('enter')
time.sleep(0.5)
```

Press Enter multiple times if necessary to finalize file selection

```
pyautogui.press('enter')
time.sleep(0.5)
pyautogui.press('enter')
time.sleep(0.5)
```

Lab Assignments

1. Single File Upload (try use send_keys() method)

   https://the-internet.herokuapp.com/upload

2. Single & Multiple Files Upload (try use send_keys() method)

   https://davidwalsh.name/demo/multiple-file-upload.php

3. Shadow DOM elements with File Upload feature (try to use Robot Class API for file upload)

   http://watir.com/examples/shadow_dom.html

4. Shadow DOM elements with File Upload feature (try to use Robot Class API for file upload)

   https://testautomationpractice.blogspot.com/