32. Handling Static and Dynamic WebTables

WebTable or HTML Table

- ➔ WebTable or HTML Table is used to organize data in tabular format on a web page.

Operations on Web Table

- ➔ count no of rows
- ➔ count no of columns
- ➔ read data from specific row and column
- ➔ read data from all rows and columns,etc

Note

- ➔ To work with data on a table we have to use **Dynamic xpath** i.e, inside the xpath we can pass the parameters dynamically.

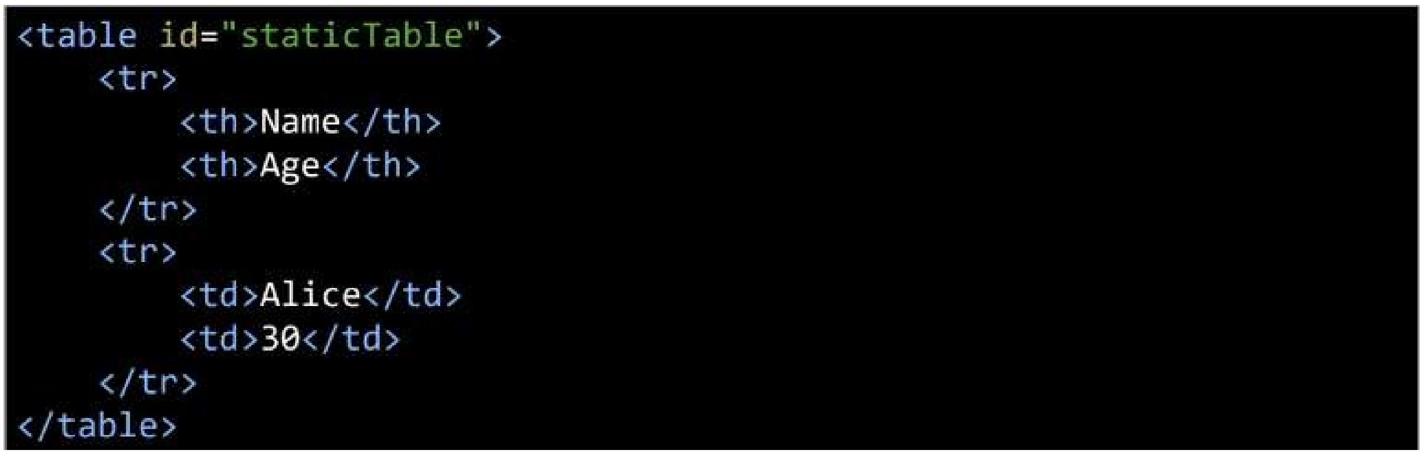
Conditions for writing Dynamic XPATH

- Convert variable into string
- Keep this in Double Quotation
- Add '+' before and after string inside Double Quotation.

```
"+str(r)+"
```

Static Web Tables

- ➔ **Static tables** have fixed data that does not change after the page is loaded. The table contents are embedded directly into the HTML of the web page.
- ➔ **Characteristics**
    - ◆ Data is hardcoded in the HTML.
    - ◆ No interaction or server-side updates after the page load.
    - ◆ Easy to automate since the structure and data remain constant.



```
<table id="staticTable">
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Alice</td>
        <td>30</td>
    </tr>
</table>
```

- ➔ Here, the content remains the same unless the HTML is changed by the developer.
- ➔ Simple to handle using Selenium as the structure and data are predictable.

StaticWebTableDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
options = webdriver.ChromeOptions()
options.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=options)
driver.get("https://testautomationpractice.blogspot.com/")
driver.maximize_window()  # maximize window
driver.implicitly_wait(10) # seconds  # implicit wait
```

Count number of rows & columns

```
noOfRows=len(driver.find_elements(By.XPATH,"//table[@name='BookTable']//tr"))
noOfColumns=len(driver.find_elements(By.XPATH,"//table[@name='BookTable']//tr[1]/th"))
print(noOfRows)  #7
print(noOfColumns) #4
```

Read specific row & Column data - Master In Selenium

```
data=driver.find_element(By.XPATH,"//table[@name='BookTable']/tbody/tr[5]/td[1]").text
print(data)
```

Approach - 1 Read all the rows & Columns data

```
print("printing all the rows and columns data....................")
for r in range(2,noOfRows+1):
  for c in range(1,noOfColumns+1):
      data =
driver.find_element(By.XPATH,"//table[@name='BookTable']/tbody/tr["+str(r)+"]/td["+str(c)+"]").text
      print(data,end='            ')
  print()
```

Approach - 2 Read all the rows & Columns data

```
for r in range(2, noOfRows + 1):
  row_data = [ ]
  for c in range(1, noOfCols + 1):
      data = driver.find_element(By.XPATH, f"//table[@name='BookTable']/tbody/tr[{r}]/td[{c}]").text
      row_data.append(data)
  print(" | ".join(row_data))
print()
```

Read data based on condition(List books name whose author is Mukesh)

```
for r in range(2,noOfRows+1):
authorName=driver.find_element(By.XPATH,"//table[@name='BookTable']/tbody/tr["+str(r)+"]/td[2]").te
xt
  if authorName=="Mukesh":
      bookName = driver.find_element(By.XPATH, "//table[@name='BookTable']/tbody/tr[" + str(r) +
"]/td[1]").text
      price = driver.find_element(By.XPATH, "//table[@name='BookTable']/tbody/tr[" + str(r) +
"]/td[4]").text
      print(bookName,"        ",authorName,"       ",price)
driver.close()
```

Print max price from the prices in table

```
prices = [ ]
for r in range(2,noofrows+1):
  data = driver.find_element(By.XPATH, "//table[@name= 'BookTable']/tbody/tr["+str(r)+"]/td[4]").text
  prices.append(data)
```

Approach 1	Approach 2
`print(max(int(price) for price in prices))`	`print(max(map(int,prices)))`

Approach 3

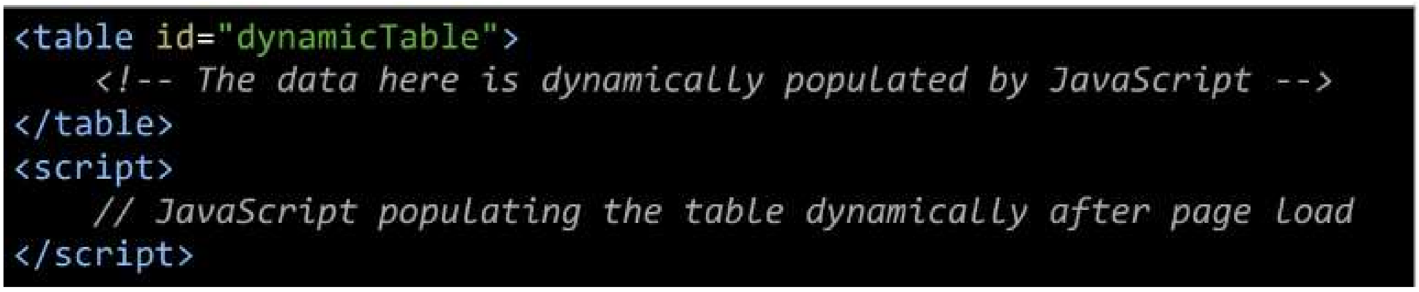
```
max_price = int(prices[0])
print(prices)
for price in prices:
  price_int = int(price)
  if price_int > max_price:
     max_price = price_int
print(max_price)
```

Dynamic Web Tables

➔ **Dynamic Web Tables** are generated or updated based on user interactions, server requests, or client-side scripts (e.g., JavaScript or AJAX).

➔ **Characteristics**

- ◆ Data can change dynamically based on user input or updates from the server.
- ◆ Often populated after the page loads, using JavaScript or AJAX.
- ◆ Data can change without reloading the page.



```
<table id="dynamicTable">
    <!-- The data here is dynamically populated by JavaScript -->
</table>
<script>
    // JavaScript populating the table dynamically after page load
</script>
```

➔ Dynamic Web Tables contents may load or update asynchronously.

➔ More complex to handle with Selenium since data may not be available immediately after the page loads. You may need to use explicit waits or check for table refresh events before interacting with the table.

DynamicWebTableDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_argument("--start-maximized")
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.implicitly_wait(10)
driver.get("https://practice.expandtesting.com/dynamic-table")
rows = driver.find_elements(By.XPATH, "//table[@class='table table-striped']//tbody//tr")
print("Number of rows:", len(rows))
```

Loop through rows to find "Chrome" and verify CPU load

```
for r in range(1, len(rows) + 1):
    name = driver.find_element(By.XPATH, f"//table[@class='table table-striped']//tbody//tr[{r}]/td[1]")
    if name.text == "Chrome":
        cpu_load = driver.find_element(By.XPATH,
                                "//td[normalize-space()='Chrome']//following-sibling::*[contains(text(),'%')]").text
        value = driver.find_element(By.ID, "chrome-cpu").text
        if cpu_load in value:
            print("CPU load of Chrome process is equal...")
        else:
            print("CPU load of Chrome process is not equal...")
        break
driver.quit()
```