**How to Handle Web Table in Selenium WebDriver**

**What is a Web Table in Selenium?**

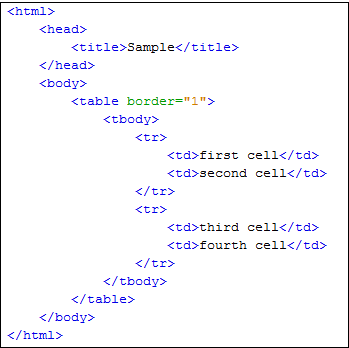
A **Web Table** in Selenium is a WebElement used for the tabular representation of data or information. The data or information displayed can be either static or dynamic. Web table and its elements can be accessed using WebElement functions and locators in Selenium. A typical example of a web table would be product specifications displayed on an eCommerce platform.

**Reading a HTML Web Table**

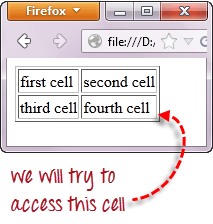
There are times when we need to access elements (usually texts) that are within HTML tables. However, it is very seldom for a web designer to provide an id or name attribute to a certain cell in the table. Therefore, we cannot use the usual methods such as “By.id()”, “By.name()”, or “By.cssSelector()”. In this case, the most reliable option is to access them using the “By.xpath()” method.

**How to write XPath for Table in Selenium**

Consider the HTML code below for handling web tables in Selenium.



We will use[XPath](https://www.guru99.com/xpath-selenium.html)to get the inner text of the cell containing the text “fourth cell.”



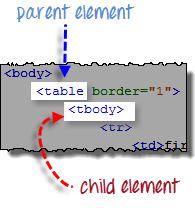
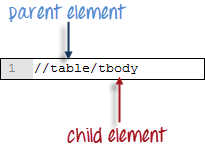
**Step 1 – Set the Parent Element (table)**

**XPath locators in WebDriver always start with a double forward slash “//” and then followed by the parent element**. Since we are dealing with web tables in Selenium, the parent element should always be the <table> tag. The first portion of our Selenium XPath table locator should, therefore, start with “//table”.

Selenium Web Table Example

**Step 2 – Add the child elements**

The element immediately under <table> is <tbody> so we can say that <tbody> is the “child” of <table>. And also, <table> is the “parent” of <tbody>. All child elements in XPath are placed to the right of their parent element, separated with one forward slash “/” like the code shown below.

**Step 3 – Add Predicates**

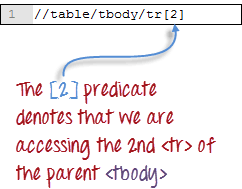
The <tbody> element contains two <tr> tags. We can now say that these two <tr> tags are “children” of <tbody>. Consequently, we can say that <tbody> is the parent of both the <tr> elements.

Another thing we can conclude is that the two <tr> elements are siblings. **Siblings refer to child elements having the same parent**.

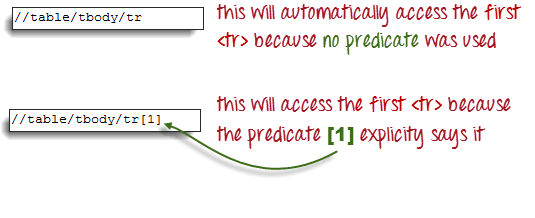
To get to the <td> we wish to access (the one with the text “fourth cell”), we must first access the **second** <tr> and not the first. If we simply write “//table/tbody/tr”, then we will be accessing the first <tr> tag.

So, how do we access the second <tr> then? The answer to this is to use **Predicates**.

**Predicates are numbers or HTML attributes enclosed in a pair of square brackets “[ ]” that distinguish a child element from its siblings**. Since the <tr> we need to access is the second one, we shall use “[2]” as the predicate.



If we won’t use any predicate, XPath will access the first sibling. Therefore, we can access the first <tr> using either of these XPath codes.

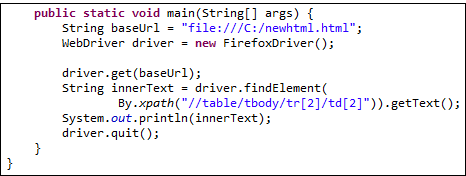


**Step 4 – Add the Succeeding Child Elements Using the Appropriate Predicates**

The next element we need to access is the second <td>. Applying the principles we have learned from steps 2 and 3, we will finalize our XPath code to be like the one shown below.

Web Table in Selenium using Xpath

Now that we have the correct XPath locator, we can already access the cell that we wanted to and obtain its inner text using the code below. It assumes that you have saved the HTML code above as “newhtml.html” within your C Drive.



public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/write-xpath-table.html";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

String innerText = driver.findElement(

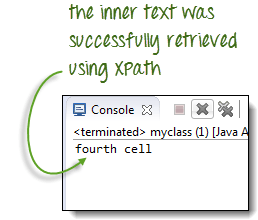
By.xpath("//table/tbody/tr[2]/td[2]")).getText();

System.out.println(innerText);

driver.quit();

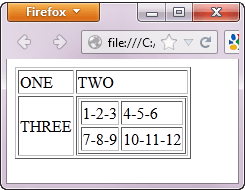
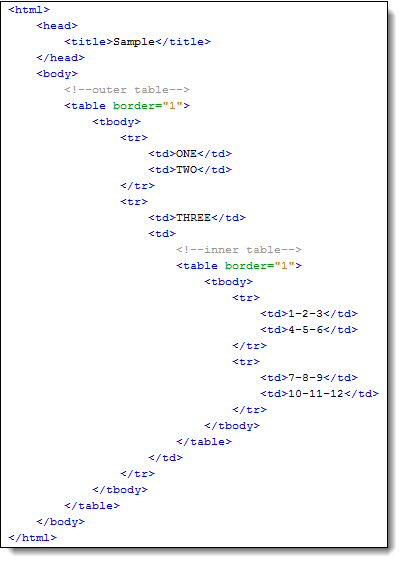
}

}

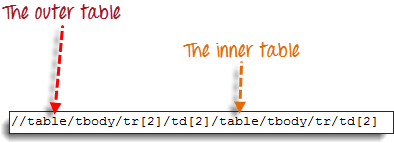


**Accessing Nested Tables**

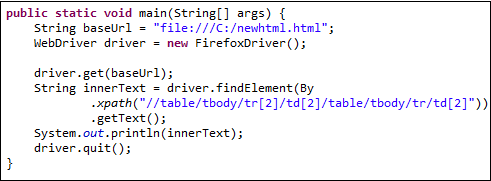
The same principles discussed above applies to nested tables. **Nested tables are tables located within another table**. An example is shown below.



To access the cell with the text “4-5-6” using the “//parent/child” and predicate concepts from the previous section, we should be able to come up with the XPath code below.



The WebDriver code below should be able to retrieve the inner text of the cell which we are accessing.



public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/accessing-nested-table.html";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

String innerText = driver.findElement(

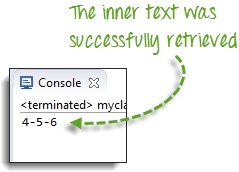
By.xpath("//table/tbody/tr[2]/td[2]/table/tbody/tr/td[2]")).getText();

System.out.println(innerText);

driver.quit();

}

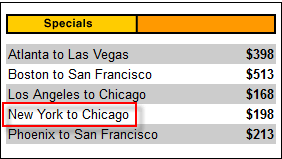
The output below confirms that the inner table was successfully accessed.

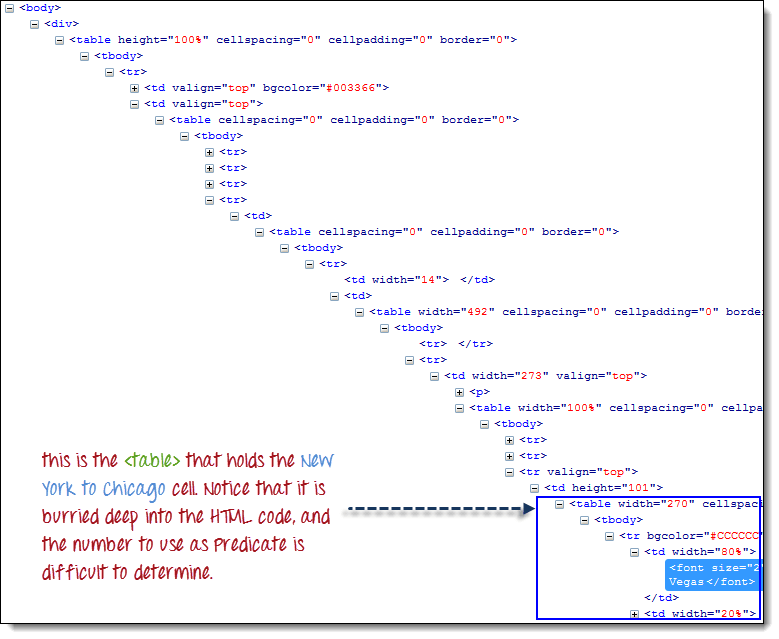


**Using Attributes as Predicates**

If the element is written deep within the HTML code such that the number to use for the predicate is very difficult to determine, we can use that element’s unique attribute instead.

In the example below, the “New York to Chicago” cell is located deep into Mercury Tours homepage’s HTML code.

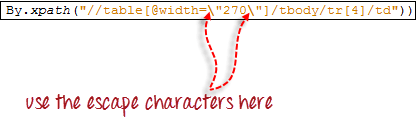




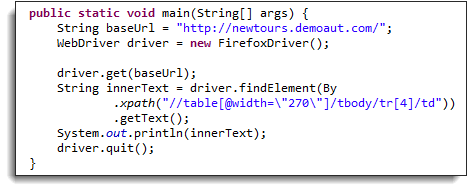
In this case, we can use the table’s unique attribute (width=”270″) as the predicate. **Attributes are used as predicates by prefixing them with the @ symbol**. In the example above, the “New York to Chicago” cell is located in the first <td> of the fourth <tr>, and so our XPath should be as shown below.

Selenium Web table example with Attributes as Predicates

Remember that when we put the XPath code in Java, we should use the escape character backward slash “\” for the double quotation marks on both sides of “270” so that the string argument of By.xpath() will not be terminated prematurely.



We are now ready to access that cell using the code below.



public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/newtours/";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

String innerText = driver.findElement(By

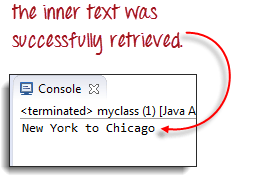
.xpath("//table[@width=\"270\"]/tbody/tr[4]/td"))

.getText();

System.out.println(innerText);

driver.quit();

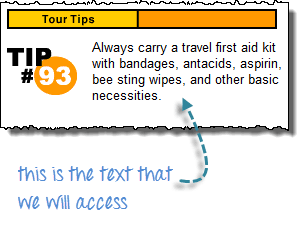
}



**Shortcut: Use Inspect Element for Accessing Tables in Selenium**

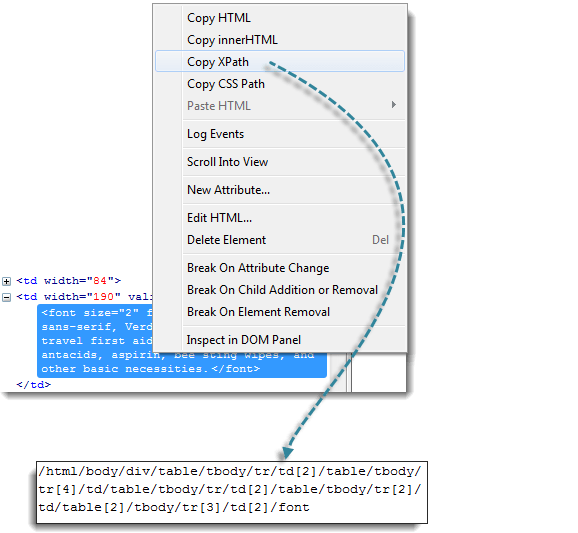
If the number or attribute of an element is extremely difficult or impossible to obtain, the quickest way to generate the XPath code is using Inspect Element.

Consider the example below from Mercury Tours homepage.



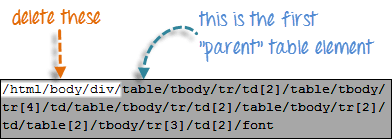
**Step 1**

Use Firebug to obtain the XPath code.



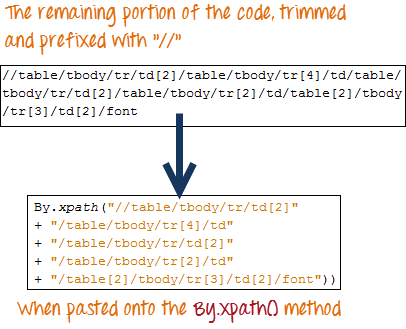
**Step 2**

Look for the first “table” parent element and delete everything to the left of it.

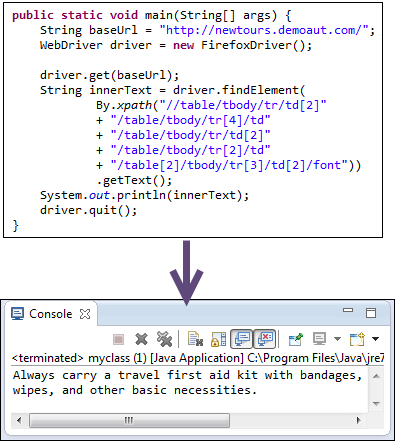


**Step 3**

Prefix the remaining portion of the code with double forward slash “//” and copy it over to your WebDriver code.



The WebDriver code below will be able to successfully retrieve the inner text of the element we are accessing.



public static void main(String[] args) {

String baseUrl = "http://demo.guru99.com/test/newtours/";

WebDriver driver = new FirefoxDriver();

driver.get(baseUrl);

String innerText = driver.findElement(By

.xpath("//table/tbody/tr/td[2]"

+ "//table/tbody/tr[4]/td/"

+ "table/tbody/tr/td[2]/"

+ "table/tbody/tr[2]/td[1]/"

+ "table[2]/tbody/tr[3]/td[2]/font"))

.getText();

System.out.println(innerText);

driver.quit();

}

WebTable Xpaths

**//For Rows**

List<WebElement> rows = driver.findElements(By.xpath("//table[@id='customers']/tbody/tr"));

int rowCount = rows.size();

System.out.println("Total rows in web table:"+ rowCount);

String beforeXpath = "//\*[@id='customers']/tbody/tr[";

String afterXpath = "]/td[1]";

for(int i = 2; i<= rowCount; i++){

String actualXpath = beforeXpath + i + afterXpath;

WebElement element = driver.findElement(By.xpath(actualXpath));

System.out.println(element.getText());

if(element.getText().equals("Island Trading")){

System.out.println("Company name:" + element.getText() + "is Found" + "at position:" + (i-1));

break;

}

}

**//Columns**

String colBeforeXpath = "//\*[@id='customers']/tbody/tr[1]/th[";

String colAfterXpath = "]";

List<WebElement> cols= driver.findElements(By.xpath("//table[@id='customers']/tbody/tr[1]/th"));

System.out.println("Total number of columns are:" + colCount);

System.out.println("Columns values are:");

for(int = 1; i<=colCount; i++){

WebElement element = driver.findElement(By.xpath(colBeforeXpath + i + colAfterXpath));

String colText = element.getText();

System.out.println(colText);

}

# Dynamic Web Table Handling in Selenium (Language used Python)

Web tables or data tables are often used in scenarios where you need to display the information in a tabular format. The data being displayed can either be static or dynamic in nature. You’d often see such examples in e-commerce portals, where product specifications are displayed in a web table. With its wide use, you’d often come across scenarios where you’ll need to handle them in your [Selenium test automation](https://www.lambdatest.com/selenium-automation) scripts.

In this [Selenium WebDriver](https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/) tutorial, I’ll take a look at how to handle a web table in Selenium along with a few useful operations that can be performed on web tables. By the end of this tutorial, you’ll gain a thorough understanding of web tables in Selenium test automation along with methodologies used to access content in the web table. To know more about [What is Selenium](https://www.lambdatest.com/selenium), you can refer to our detailed page on the topic.

## What is a Web Table in Selenium?

Web table in [Selenium](https://www.lambdatest.com/selenium) is a WebElement just like any other popular WebElements like text boxes, radio buttons, checkboxes, drop-down menus, etc. Web table and its contents can be accessed by using the WebElement functions along with [Selenium locators](https://www.lambdatest.com/learning-hub/selenium-locators) to identify the element (row/column) on which the operation needs to be performed.

A table consists of rows and columns. The table created for a web page is called a web table. Below are some of the important tags associated with a web table:

* **< table >** – Defines an HTML table
* **< th >** – Contains header information in a table
* **< tr >** – Defines a row in a table
* **< td >** – Defines a column in a table

## Types of Web Tables in Selenium

There are two broad categories of tables namely:

#### Static Web Table

As the name indicates, the information in the table is static in nature.

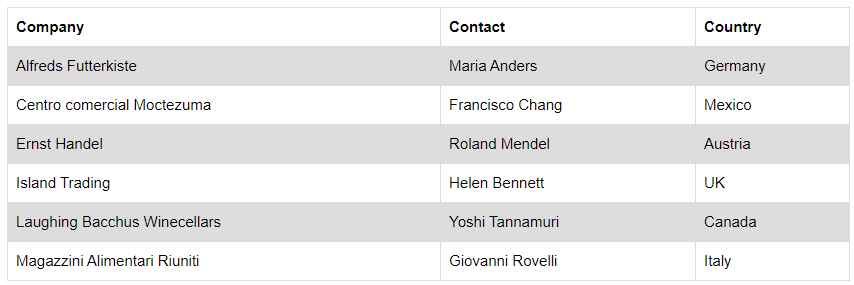
#### Dynamic Web Table

The information displayed in the table is dynamic. E.g. Detailed Product information on e-commerce websites, sales reports, etc.

For the demonstration to handle the table in Selenium, we make use of a table that is available in the [w3school HTML table](https://www.w3schools.com/html/html_tables.asp) page. Though there are fewer cross browser testing issues when using tables, some of the [old browser versions](https://caniuse.com/#search=HTML%20table) of Internet Explorer, Chrome, and other web browsers do no support HTML Table APIs.

## Handling Web Tables in Selenium

I’ll use the local Selenium WebDriver for performing browser actions to handle table in Selenium, present on [w3schools html table](https://www.w3schools.com/html/html_tables.asp) page. The HTML code for the web table used for demonstration is available in the [tryit adapter](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_table_intro) page.



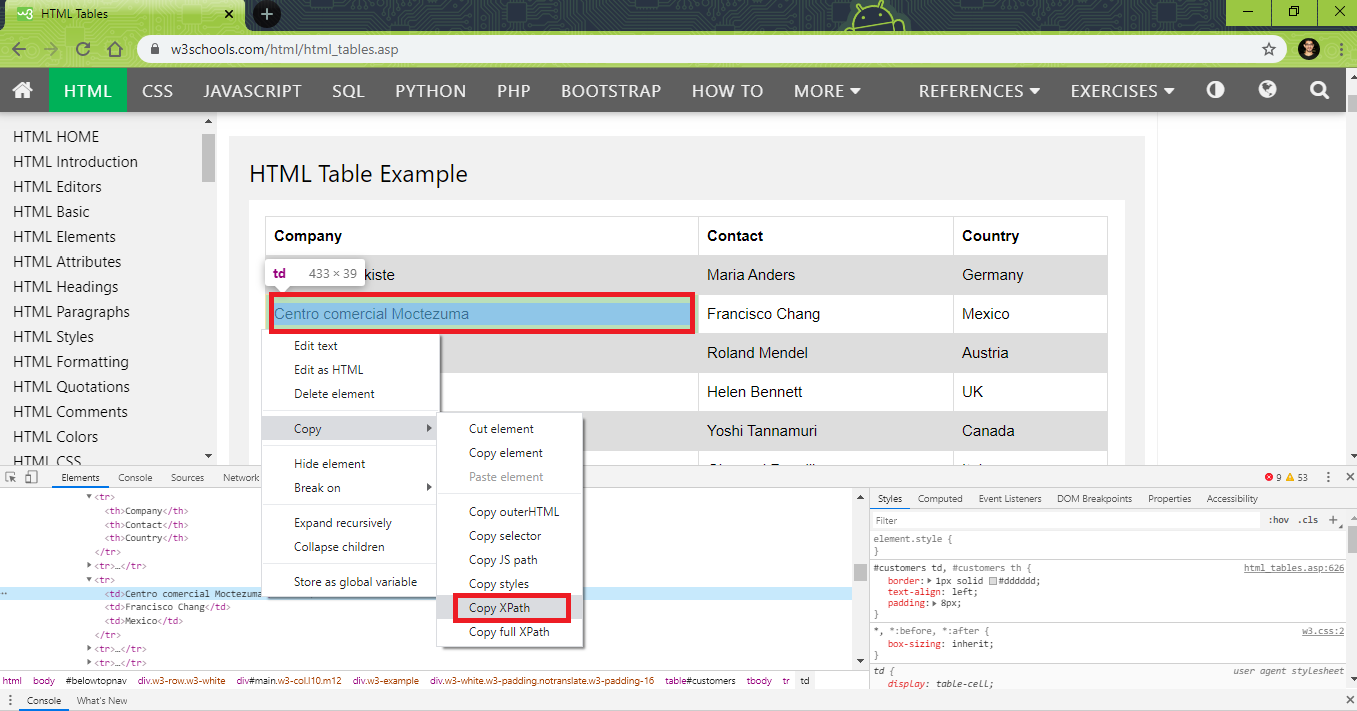
The Selenium WebDriver for popular browsers can be downloaded from the locations mentioned below:

|  |  |
| --- | --- |
| **BROWSER** | **DOWNLOAD LOCATION** |
| Opera | <https://github.com/operasoftware/operachromiumdriver/releases> |
| Firefox | <https://github.com/mozilla/geckodriver/releases> |
| Chrome | <http://chromedriver.chromium.org/downloads> |
| Internet Explorer | <https://github.com/SeleniumHQ/selenium/wiki/InternetExplorerDriver> |
| Microsoft Edge | <https://blogs.windows.com/msedgedev/2015/07/23/bringing-automated-testing-to-microsoft-edge-through-webdriver/> |

I’ll use the Python unittest framework to handle tables in [Selenium WebDriver](https://www.lambdatest.com/selenium#selenium-webdriver). The core logic for accessing elements in web tables still remains the same even if you are using other programming languages for Selenium test automation.

### **Handling Number Of Rows & Columns In Web Table**

The < tr > tag in the table indicates the rows in the table and that tag is used to get information about the number of rows in it. Number of columns of the web table in Selenium are calculated using XPath (//\*[@id=’customers’]/tbody/tr[2]/td). XPath of the rows and columns are obtained using the inspect tool in the browser to handle tables in Selenium for [automated browser testing](https://www.lambdatest.com/).



Source: [*W3School*](https://www.w3schools.com/html/html_tables.asp)

Though the header in a web table does not the < td >, the < th > tag could still be used in the current example to calculate the number of columns. The XPath for computing the number of columns using < th > tag is //\*[@id=’customers’]/tbody/tr/th

A WebDriverWait of 30 seconds is added to ensure that the loading of the Web Table (CLASS\_NAME = w3-example) is complete before any operations are performed to handle the table in Selenium.

**Get number of rows for a web table in Selenium**

|  |
| --- |
| **num\_rows = len (driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr"))** |
|  |

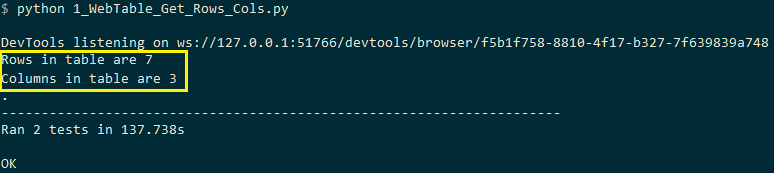
**Get number of columns for a web table in Selenium**

|  |
| --- |
| **num\_cols = len (driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr[2]/td"))** |

**Complete Implementation**

|  |
| --- |
| import unittest |
| import time |
| from selenium import webdriver |
| from selenium.webdriver.support.select import Select |
| from selenium.webdriver.common.by import By |
| from selenium.webdriver.support.ui import WebDriverWait |
| from selenium.webdriver.support import expected\_conditions as EC |
| test\_url = "https://www.w3schools.com/html/html\_tables.asp" |
|  |
| class WebTableTest(unittest.TestCase): |
| def setUp(self): |
| self.driver = webdriver.Chrome() |
| self.driver.maximize\_window() |
| def test\_1\_get\_num\_rows\_(self): |
| driver = self.driver |
| driver.get(test\_url) |
| WebDriverWait(driver, 60).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, "w3-example"))) |
| num\_rows = len (driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr")) |
| print("Rows in table are " + repr(num\_rows)) |
| def test\_2\_get\_num\_cols\_(self): |
| driver = self.driver |
| driver.get(test\_url) |
| WebDriverWait(driver, 60).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, "w3-example"))) |
| # num\_cols = len (driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr/th")) |
| num\_cols = len (driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr[2]/td")) |
| print("Columns in table are " + repr(num\_cols)) |
| def tearDown(self): |
| self.driver.close() |
| self.driver.quit() |
| if \_\_name\_\_ == "\_\_main\_\_": |
| unittest.main() |

Below is the output snapshot



### **Print Content Of The Web Table In Selenium**

To access the content present in every row and column to handle the table in Selenium, we iterate each and every row (< tr >) in the web table. Once the details about the rows are obtained, we iterate the < td > tags under that row.

In this case for this Selenium WebDriver tutorial, both the rows (< tr >) and columns (< td >) would be variable. Hence, the row numbers and column numbers are computed dynamically. Shown below is the XPath for accessing information in specific rows and columns:

* **XPath to access Row :** 2, Column : 2 – //\*[@id=”customers”]/tbody/tr[2]/td[1]
* **XPath to access Row :** 3, Column : 1 – //\*[@id=”customers”]/tbody/tr[3]/td[1]

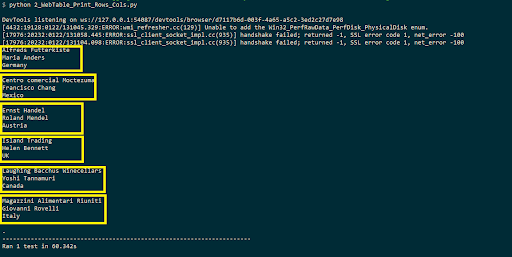
The table on which Selenium test automation is being performed has 7 rows and 3 columns. Hence, a nested for loop is executed with rows ranging from 2..7 and columns ranging from 1..4. The variables factors i.e. row number and column number are added to formulate the final XPath.

|  |
| --- |
| for t\_row in range(2, (rows + 1)):    for t\_column in range(1, (columns + 1)):        FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath + str(t\_column) + aftertr\_XPath        cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |

Shown below in this Selenium WebDriver tutorial, is the complete implementation to get all the contents present to handle table in Selenium.

|  |
| --- |
| import unittest |
| import time |
| test\_url = "https://www.w3schools.com/html/html\_tables.asp" |
| before\_XPath = "//\*[@id='customers']/tbody/tr[" |
| aftertd\_XPath = "]/td[" |
| aftertr\_XPath = "]" |
| def test\_get\_row\_col\_info\_(self): |
| driver = self.driver |
| driver.get(test\_url) |
| # time.sleep(30) |
| WebDriverWait(driver, 60).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, "w3-example"))) |
| rows = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr")) |
| # print (rows) |
| columns = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr[2]/td")) |
| # print(columns) |
| # print("Company"+" "+"Contact"+" "+"Country") |
| for t\_row in range(2, (rows + 1)): |
| for t\_column in range(1, (columns + 1)): |
| FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath + str(t\_column) + aftertr\_XPath |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| # print(cell\_text, end = ' ') |
| print(cell\_text) |
| print() |

The output snapshot to print content to handle table in Selenium is below:



### **Read Data In Rows To Handle Table In Selenium**

For accessing the content present in every row, to handle table in Selenium, the rows (< tr >) are variable whereas the columns (< td >) would remain constant. Hence, the rows are computed dynamically. Below in this Selenium WebDriver tutorial is the XPath for accessing information with rows being the variable factor and columns remaining constant for Selenium test automation.

* **XPath to access Row :** 1, Column : 1 – //\*[@id=”customers”]/tbody/tr[1]/td[1]
* **XPath to access Row :** 2, Column : 2 – //\*[@id=”customers”]/tbody/tr[2]/td[2]
* **XPath to access Row :** 3, Column : 2 – //\*[@id=”customers”]/tbody/tr[3]/td[2]

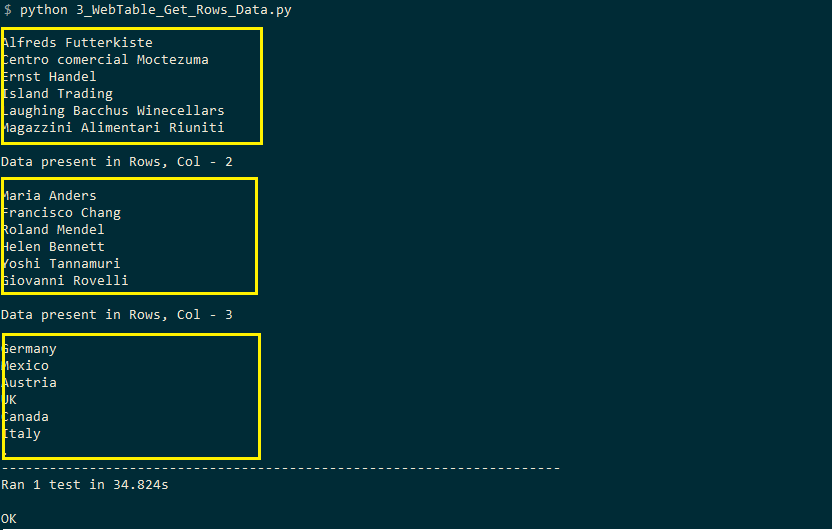
A for loop is executed with rows ranging from 2..7. The column values are appended to the XPath are td[1]/td[2]/td[3] depending on the row & column that has to be accessed to handle the table in Selenium.

|  |
| --- |
| before\_XPath = "//\*[@id='customers']/tbody/tr["  aftertd\_XPath\_1 = "]/td[1]"  aftertd\_XPath\_2 = "]/td[2]"  aftertd\_XPath\_3 = "]/td[3]"    for t\_row in range(2, (rows + 1)):      FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath\_1      cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text      print(cell\_text) |

**Complete Implementation**

|  |
| --- |
| #Selenium webdriver tutorial to handletable in Selenium for Selenium test automation |
| import unittest |
| import time |
| from selenium import webdriver |
| from selenium.webdriver.support.select import Select |
| from selenium.webdriver.common.by import By |
| from selenium.webdriver.support.ui import WebDriverWait |
| from selenium.webdriver.support import expected\_conditions as EC |
| test\_url = "https://www.w3schools.com/html/html\_tables.asp" |
| before\_XPath = "//\*[@id='customers']/tbody/tr[" |
| aftertd\_XPath\_1 = "]/td[1]" |
| aftertd\_XPath\_2 = "]/td[2]" |
| aftertd\_XPath\_3 = "]/td[3]" |
| #aftertr\_XPath = "]" |
| def test\_get\_row\_col\_info\_(self): |
| driver = self.driver |
| driver.get(test\_url) |
| # time.sleep(30) |
| WebDriverWait(driver, 60).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, "w3-example"))) |
| rows = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr")) |
| # print (rows) |
| columns = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr[2]/td")) |
| # print(columns) |
| print("Data present in Rows, Col - 1") |
| print() |
| for t\_row in range(2, (rows + 1)): |
| FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath\_1 |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| print(cell\_text) |
| print() |
| print("Data present in Rows, Col - 2") |
| print() |
| for t\_row in range(2, (rows + 1)): |
| FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath\_2 |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| print(cell\_text) |
| print() |
| print("Data present in Rows, Col - 3") |
| print() |
| for t\_row in range(2, (rows + 1)): |
| FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath\_3 |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| print(cell\_text) |

The output snapshot to read data in rows to handle table in Selenium is below:



### **Read Data In Columns To Handle Table In Selenium**

For column-wise access to handle table in Selenium, the rows remain constant whereas the column numbers are variable i.e. the columns are computed dynamically. Below in this Selenium WebDriver Tutorial is the XPath for accessing information where columns are variable and rows are constant.

* **XPath to access Row : 2, Column : 2** – //\*[@id=”customers”]/tbody/tr[2]/td[2]
* **XPath to access Row : 2, Column : 3** – //\*[@id=”customers”]/tbody/tr[2]/td[3]
* **XPath to access Row : 2, Column : 4** – //\*[@id=”customers”]/tbody/tr[2]/td[4]

A for loop is executed with columns ranging from 1..4 The row values are appended to the XPath are tr[1]/tr[2]/tr[3] depending on the row & column that has to be accessed.

|  |
| --- |
| before\_XPath\_1 = "//\*[@id='customers']/tbody/tr[1]/th["  before\_XPath\_2 = "//\*[@id='customers']/tbody/tr[2]/td["  after\_XPath = "]"    for t\_col in range(1, (num\_columns + 1)):     FinalXPath = before\_XPath\_1 + str(t\_col) + after\_XPath     cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text     print(cell\_text) |

**Complete Implementation**

|  |
| --- |
| import unittest |
| import time |
| from selenium import webdriver |
| from selenium.webdriver.support.select import Select |
| from selenium.webdriver.common.by import By |
| from selenium.webdriver.support.ui import WebDriverWait |
| from selenium.webdriver.support import expected\_conditions as EC |
| test\_url = "https://www.w3schools.com/html/html\_tables.asp" |
| before\_XPath\_1 = "//\*[@id='customers']/tbody/tr[1]/th[" |
| before\_XPath\_2 = "//\*[@id='customers']/tbody/tr[2]/td[" |
| after\_XPath = "]" |
| def test\_get\_row\_col\_info\_(self): |
| driver = self.driver |
| driver.get(test\_url) |
| # time.sleep(30) |
| WebDriverWait(driver, 60).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, "w3-example"))) |
|  |
| num\_rows = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr")) |
| # print (rows) |
| num\_columns = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr[2]/td")) |
| # print(columns) |
| print("Data present in Col - 1 i.e. Title") |
| print() |
| for t\_col in range(1, (num\_columns + 1)): |
| FinalXPath = before\_XPath\_1 + str(t\_col) + after\_XPath |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| print(cell\_text) |
| print("Data present in Col - 2") |
| print() |
| for t\_col in range(1, (num\_columns + 1)): |
| FinalXPath = before\_XPath\_2 + str(t\_col) + after\_XPath |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| print(cell\_text) |

As seen in the execution snapshot, the header column is also read to fetch the title of the columns.

A picture containing graphical user interface

Description automatically generated

### **Locating An Element To Handle Table In Selenium**

The intention of this test for this Selenium WebDriver tutorial is to look for the presence of an element in the web table. For doing the same, content in each and every cell of the web table is read and compared with the search term. If the element is present, the corresponding row and element are printed to handle the table in Selenium.

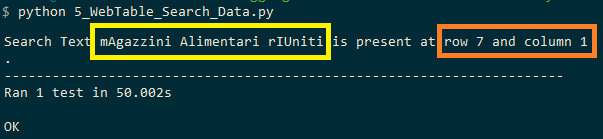
As it involves reading the data in every cell, we make use of the logic covered in the section titled Print content of the web table in Selenium. A case insensitive search is performed to validate the presence of the search term to handle table in Selenium.

|  |
| --- |
| for t\_row in range(2, (num\_rows + 1)):    for t\_column in range(1, (num\_columns + 1)):        FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath + str(t\_column) + aftertr\_XPath        cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text        if ((cell\_text.casefold()) == (search\_text.casefold())):           print("Search Text "+ search\_text +" is present at row " + str(t\_row) + " and column " + str(t\_column))           elem\_found = True           break |

**Complete Implementation**

|  |
| --- |
| import unittest |
| import time |
| from selenium import webdriver |
| from selenium.webdriver.support.select import Select |
| from selenium.webdriver.common.by import By |
| from selenium.webdriver.support.ui import WebDriverWait |
| from selenium.webdriver.support import expected\_conditions as EC |
| test\_url = "https://www.w3schools.com/html/html\_tables.asp" |
| before\_XPath\_1 = "//\*[@id='customers']/tbody/tr[1]/th[" |
| before\_XPath\_2 = "//\*[@id='customers']/tbody/tr[2]/td[" |
| after\_XPath = "]" |
| search\_text = "mAgazzini Alimentari rIUniti" |
| def test\_get\_row\_col\_info\_(self): |
| driver = self.driver |
| driver.get(test\_url) |
| # time.sleep(30) |
| WebDriverWait(driver, 60).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, "w3-example"))) |
|  |
| num\_rows = len(driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr")) |
| num\_columns = len (driver.find\_elements\_by\_xpath("//\*[@id='customers']/tbody/tr[2]/td")) |
| elem\_found = False |
| for t\_row in range(2, (num\_rows + 1)): |
| for t\_column in range(1, (num\_columns + 1)): |
| FinalXPath = before\_XPath + str(t\_row) + aftertd\_XPath + str(t\_column) + aftertr\_XPath |
| cell\_text = driver.find\_element\_by\_xpath(FinalXPath).text |
| if ((cell\_text.casefold()) == (search\_text.casefold())): |
| print("Search Text "+ search\_text +" is present at row " + str(t\_row) + " and column " + str(t\_column)) |
| elem\_found = True |
| break |
| if (elem\_found == False): |
| print("Search Text "+ search\_text +" not found") |

As seen in the execution snapshot for this Selenium WebDriver tutorial, the search term was present at row-7 and column-1



Though there are many such operations can be carried out on web table in Selenium, we have covered the core aspects in this Selenium WebDriver tutorial.