

We are aware that a webpage consists of numerous *WebElements* such as text boxes, buttons, lists, etc. We can perform a variety of actions on these *WebElements* using **Selenium** commands like search elements, associate events with web elements, etc. To perform these actions we first need to interact with a web page so that we can use *WebElement Commands/actions*. In this topic, we will discuss the different methods used to *find an element on the webpage using Selenium* so that we can perform actions on these elements. We will cover the following topics in this article.

Find elements using Selenium WebDriver?

Why do we need to find a web element in Selenium?

How to find elements in Selenium?

What is By class in Selenium?

Difference between find Element and find Elements in Selenium.

Find elements using Selenium WebDriver?

As mentioned above to interact with *WebElements*, we first have to find or locate these elements on the webpage. We can find elements on a web page by specifying the attributes such as **Id** of the element or **class name** of the element and such other parameters. These alternatives using which we can find elements on a webpage are called **locator strategies**.

The following are the locator strategies we can use while locating the elements.

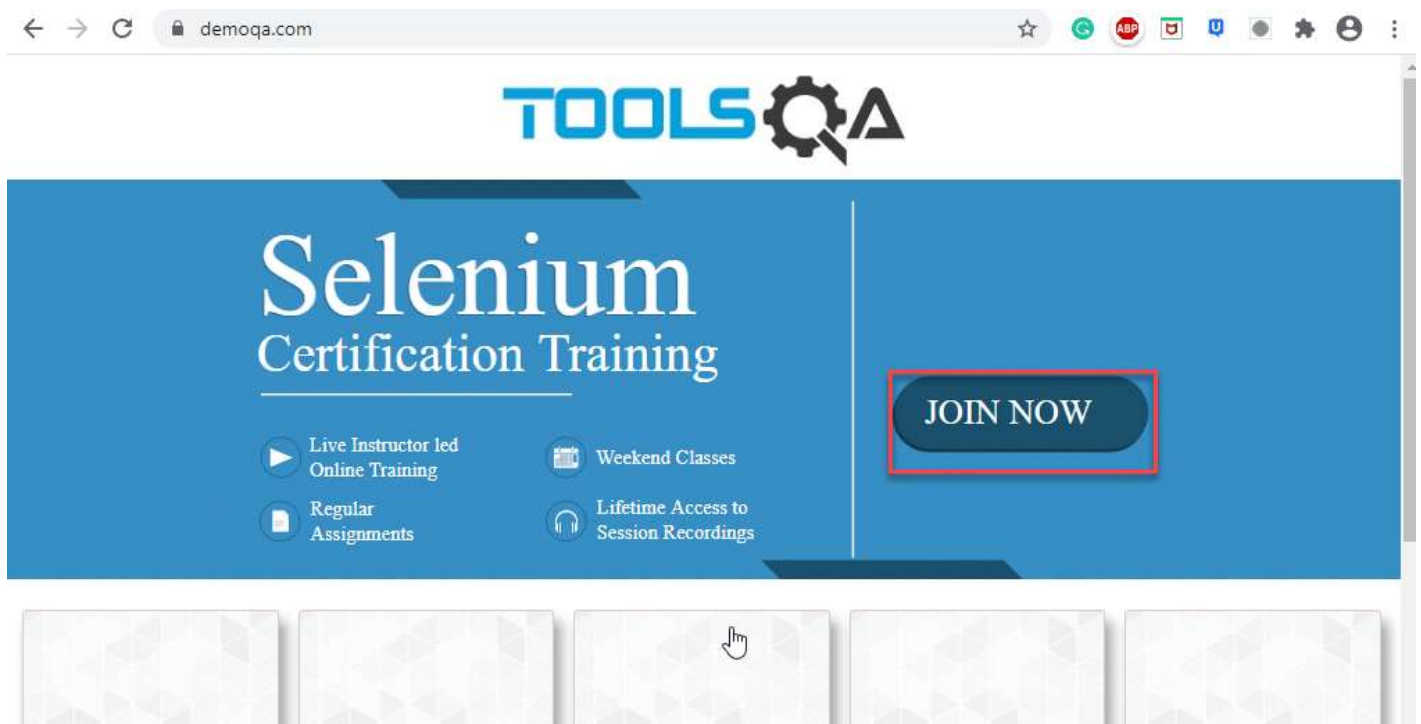
| Locator | Description |
|--------------------------|---|
| id | finds elements by ID attribute. The search value given should match the ID attribute. |
| name | Finds or Locates elements based on the NAME attribute. The name attribute is used to match the search value. |
| class name | Finds elements that match the class name specified. Note that compound classes are not allowed as strategy names. |
| tag name | Finds or Locates elements having tag names that match the search value. |
| CSS selector | Matches CSS selector to find the element. |
| XPath | Matches XPath expression to the search value and based on that the element is located. |
| link text | Here the visible text whose anchor elements are to be found is matched with the search value. |
| partial link text | Here also we match the visible text with the search value and find the anchor value. If we are matching multiple elements, only the first entry will be selected. |

Now before moving to how we can use these various types of locators to locate the elements, let's first understand why exactly there is a need to find the elements in *Selenium*?

Why do we need to find an element in Selenium?

We know that we use *Selenium* mostly for *UI* testing of a web-based application. Since we need to perform automatic feature interaction with the web page, we need to locate web elements so that we can trigger some *JavaScript* events on web elements like click, select, enter, etc. or add/ update values in the text fields. To perform these activities it is important to first locate the element on the web page and then perform all these actions.

For example, suppose given a web page "demoqa.com" as shown below.



Now, let us say we need to perform some actions on the "**JOIN NOW**" button. So before implementing the code for the say *click* event on this button, we will have to first *find* this element on the web page. So, how we are going to *find* the element so that we can carry on with our actions?

We will use two methods '*findElement*' and '*findElements*' provided by *Selenium WebDriver* for this purpose. Now let us go ahead and understand the details of these methods.

How to find elements in Selenium?

As discussed, *Selenium WebDriver* provides two methods using which we can find an *element* or *list of elements* on a web page. These are:

findElement(): This method uniquely finds a web element on the web page.

findElements(): This method finds a list of web elements on the web page.

Let's understand the usage and details of these methods in the following sections:

findElement() in Selenium

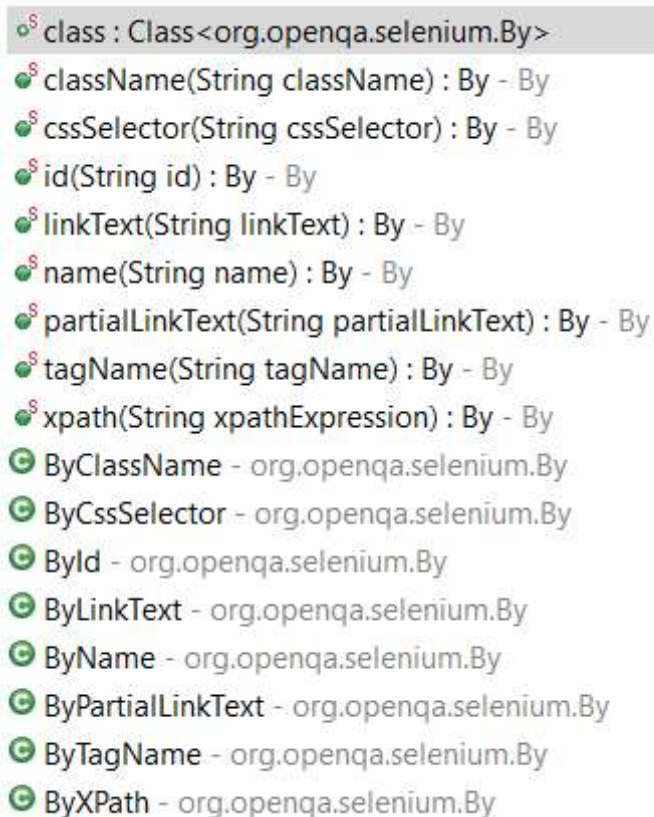
The *findElement()* method of the *Selenium WebDriver* finds a unique web element within the webpage.

It's syntax looks like below:

```
WebElement elementName = driver.findElement  
    (By.LocatorStrategy("LocatorValue"));
```

As shown in the above syntax, this command accepts the **"By "** object as the argument and returns a *WebElement* object.

The **"By"** is a locator or query object and accepts the locator specifier or strategies we discussed above. So if we write the line **"driver.findElement(By.)"** then the *Eclipse IntelliSense* will give the following locator strategies that we can associate with **By object**.



```
class : Class<org.openqa.selenium.By>  
className(String className) : By - By  
cssSelector(String cssSelector) : By - By  
id(String id) : By - By  
linkText(String linkText) : By - By  
name(String name) : By - By  
partialLinkText(String partialLinkText) : By - By  
tagName(String tagName) : By - By  
xpath(String xpathExpression) : By - By  
ByClassName - org.openqa.selenium.By  
ByCssSelector - org.openqa.selenium.By  
ById - org.openqa.selenium.By  
ByLinkText - org.openqa.selenium.By  
ByName - org.openqa.selenium.By  
ByPartialLinkText - org.openqa.selenium.By  
ByTagName - org.openqa.selenium.By  
ByXPath - org.openqa.selenium.By
```

The above screenshot shows all the options that we get when we write 'By'. We will explain each of these strategies in the later sections of this chapter.

Note: In case there is no matching element found, the `findElement` command throws `NoSuchElementException`.

But what happens if there are multiple elements matching the criteria provided in the `findElement()` method? When such a case occurs, the **`findElement()` method returns the first most element within the web page.**

`findElements()` in Selenium

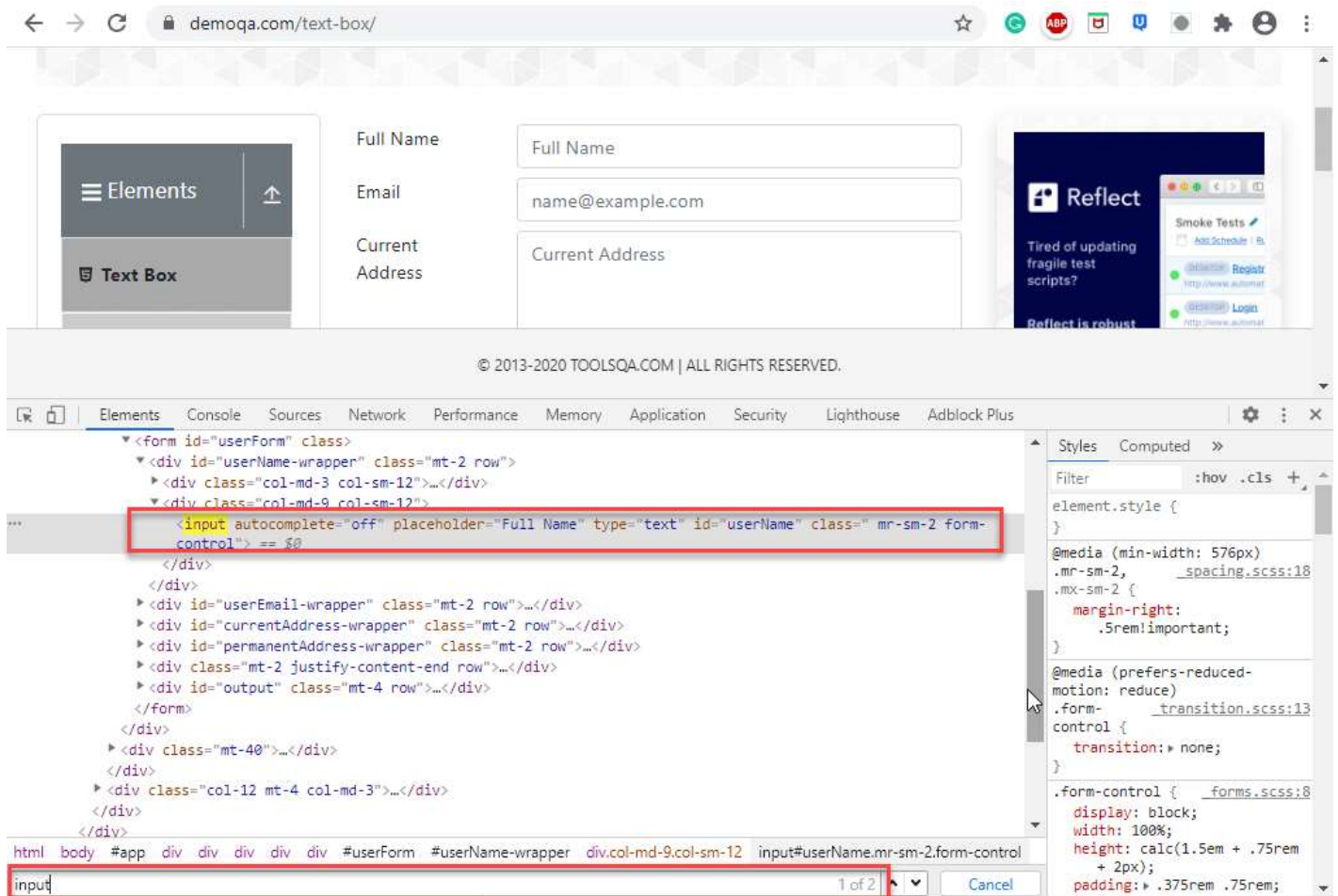
The command `findElements()` returns a list of web elements that match the specified criteria, unlike `findElement()` which returned a unique element. **If there are no matching elements then an empty list returns.**

The general syntax of `findElements()` command in *Selenium WebDriver* is as below:

```
List<WebElement> elementName = driver.findElements(By.LocatorStrategy("LocatorValue"));
```

Like the `findElement()` command, this method also accepts the **"By "** object as the parameter and returns a **`WebElement`** list.

Let us consider an example wherein we need to find the number of elements having tag name as **"input "** on the [DemoQA text box page](#). The inspect panel for this is as below.



In the above screenshot, when we search for the tag name 'input' two entries return (shown by the red rectangle around the search tool which says 1/2).

The following program shows the example of the **findElements()** method in which we provide the *By* object with *tagName*.

```
import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByTagName {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.exe");

        WebDriver driver = new ChromeDriver();

        driver.get("https://demoqa.com/text-box/");

        // Find elements using tag name
        List allInputElement = driver.findElements(By.tagName("input"));

        if(allInputElement.size() != 0)
        {
            System.out.println(allInputElement.size() + " Elements found by TagName");
        }
    }
}
```

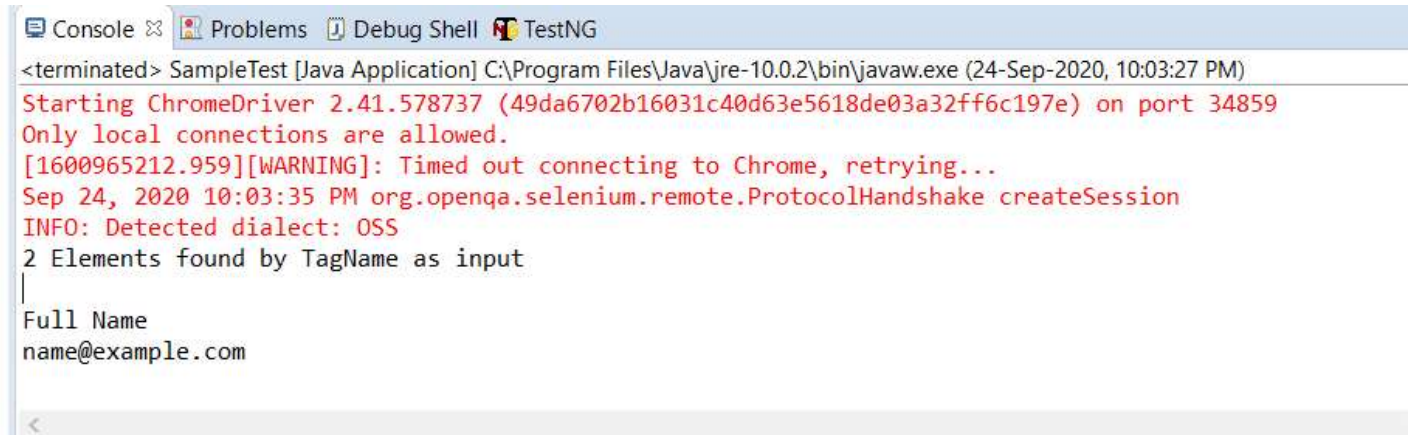


```

        for(WebElement inputElement : allInputElements)
        {
            System.out.println(inputElement.getAttribute("placeholder"));
        }
    }
}

```

Following is the program output.



The screenshot shows an IDE console window with the following output:

```

<terminated> SampleTest [Java Application] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (24-Sep-2020, 10:03:27 PM)
Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 34859
Only local connections are allowed.
[1600965212.959][WARNING]: Timed out connecting to Chrome, retrying...
Sep 24, 2020 10:03:35 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
2 Elements found by TagName as input
|
Full Name
name@example.com

```

Next, let us understand how to use different locator strategies with *findElement()* and *findElements()* commands.

What is By class in Selenium?

In this section, we will understand how to use *Selenium WebDriver's findElement()* and *findElements()* with different strategies using the **By class**. The 'By' class accepts various locator strategies explained above to find an element or elements on a web page. Let us discuss all the By class locator strategies.

How to find an element using the attribute "id" in Selenium?

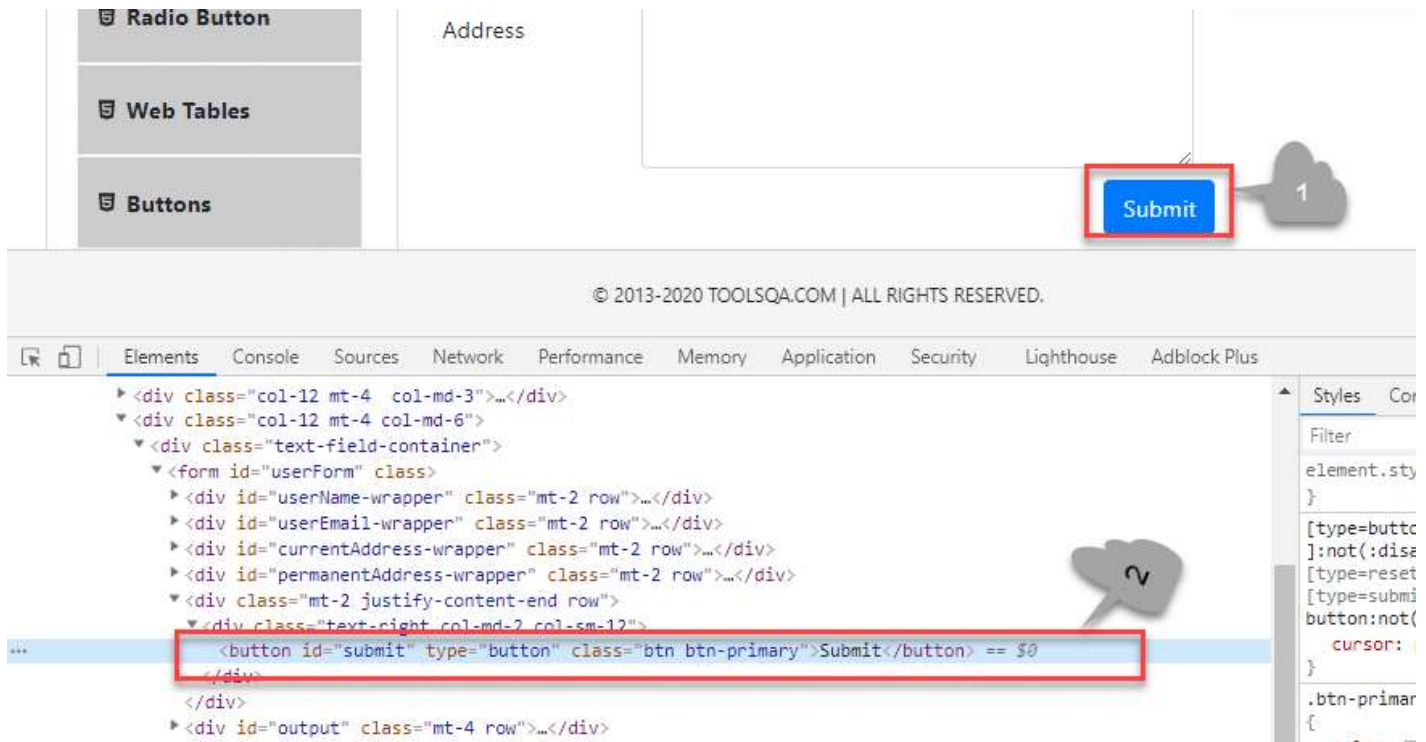
Using **"id"** to find an element is by far the most common strategy used to find an element. Suppose if the webpage uses dynamically generated *ids*, then this strategy returns the first web element that matches the *id*.

This strategy is preferred as most web pages are designed by associating *ids* with the elements. This is because using *IDs* is the easiest and quickest way to locate elements because of its simplicity while coding a web page. The **value of the id attribute** is a String type parameter.

The general syntax of `findElement()` command using **By id** strategy is :

```
WebElement elm = driver.findElement(By.id("Element_Id"));
```

As an example consider the following element in the [DemoQA text box page](#):



Here we have selected the **"submit"** button (marked 1). The element code for this is marked 2 in the above screenshot.

The `findElement()` command corresponding to the above element:

```
WebElement element = driver.findElement(By.id("submit"));  
// Action can be performed on Button element  
element.submit();
```

Note: If none of the web elements within the web page matches the id attribute then a **"NoSuchElementException"** is raised.

Note: UI developers have to ensure that the ids on the page are unique. Auto-generated or dynamically generated ids are usually non-unique.

The complete program to find an element using the **"By.id"** object is as seen below:

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementById {

    public static void main(String[] args) {

        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.exe")

        WebDriver driver = new ChromeDriver();

        driver.get("https://demoqa.com/text-box/");
        WebElement element = driver.findElement(By.id("submit"));

        if(element != null) {
            System.out.println("Element found by ID");
        }
    }
}

```

This program gives the following output.

```

Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 37461
Only local connections are allowed.
[1600441342.886][WARNING]: Timed out connecting to Chrome, retrying...
Sep 18, 2020 8:32:24 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Element found by ID

```

The above program is a program to find an element using the id (*By.id*) of that element. We provide an appropriate *URL* from which we need to search an element and then call "***findElement()***" with the argument *By.id("elementID")*. This call returns the given element with the specified id.

How to find an element using the attribute "name" in Selenium?

This strategy is the same as ***id*** except that the locator locates an element using the "***name***" instead of "***id***".

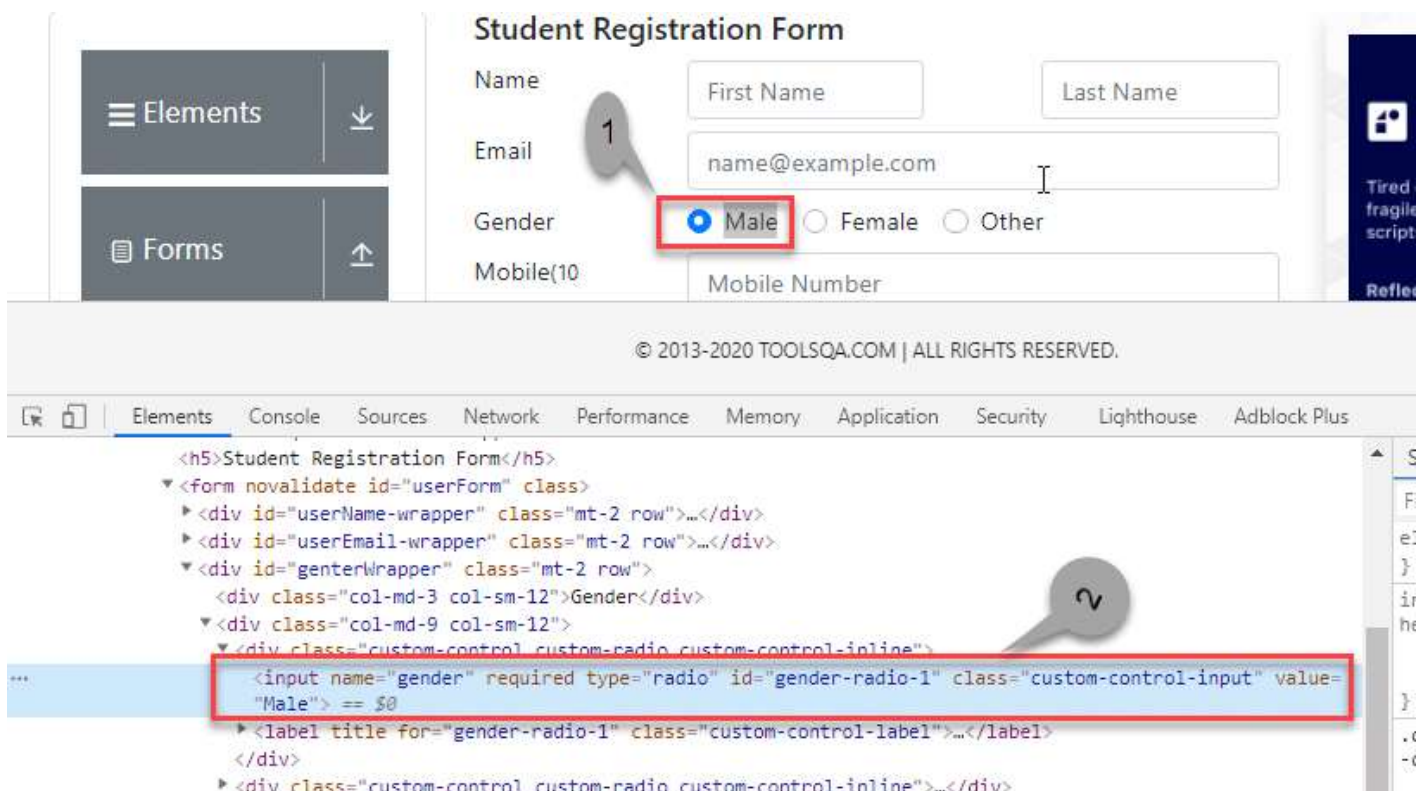
The ***value of the NAME*** attribute accepted is of type String. The general syntax of the *findElement()* method with By Name strategy is as below.

```

WebElement elm = driver.findElement(By.name("Element_NAME"));

```


For example, consider the following element on the page [DemoQAAutomationPracticeForm](#) :



In the above screenshot, we select the first gender value (*marked 1*). Its corresponding element in the *DOM* is highlighted (*marked 2*).

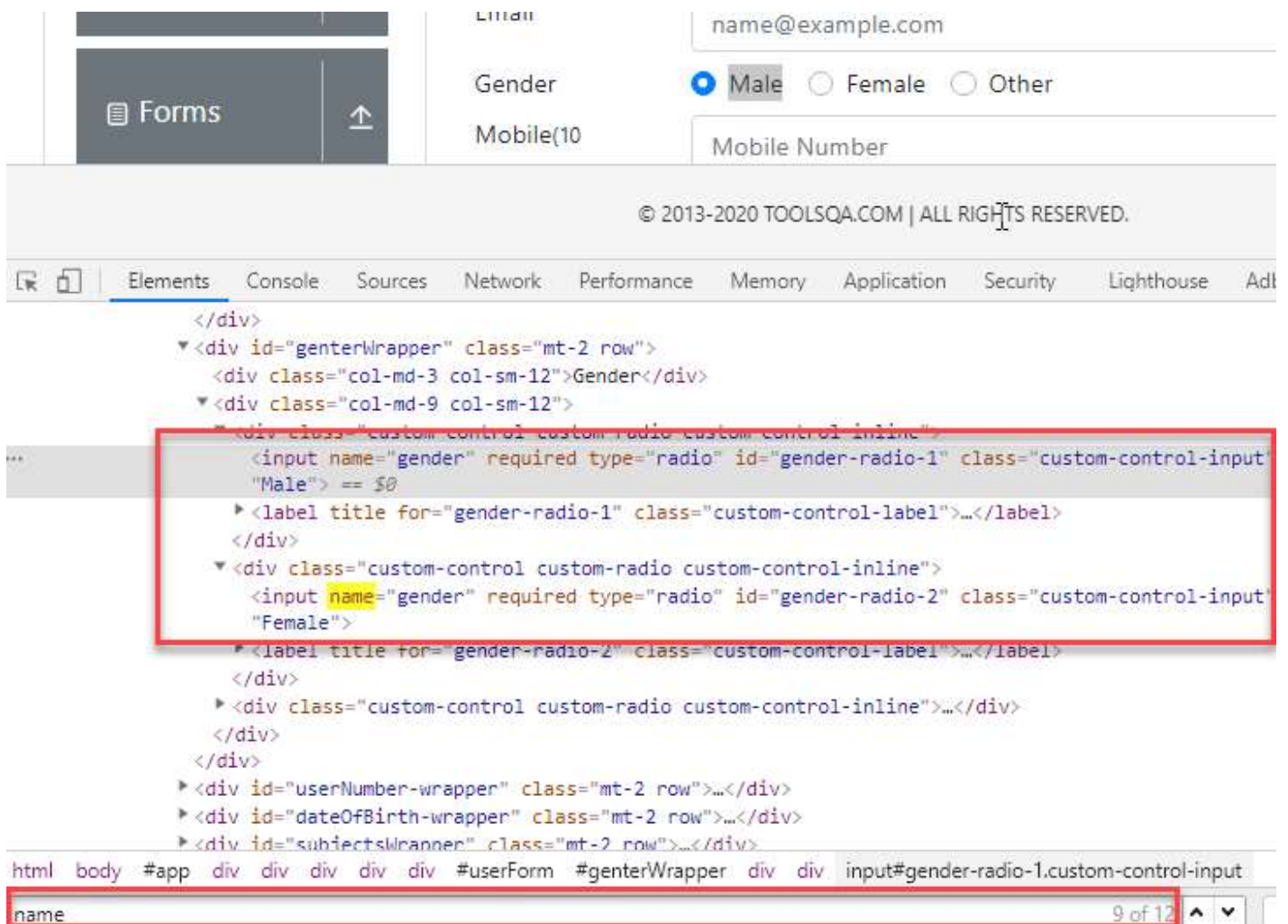
The corresponding *findElement()* method call for the above element is:

```
WebElement element = driver.findElement(By.name("gender"));
// Action can be performed on Input Text element
element.sendKeys("ToolsQA");
```

As a result of this method call, the first element matching the given name attribute value returns. If we can not find the match, **NoSuchElementException** raises.

Providing name as a strategy is also an efficient way to find an element but again if the names are not unique then the method suffers.

For example, consider the following element:



In the above screenshot, there are two elements with the same *name* = *gender*. In this case, the *findElement()* method returns the first element.

Following code shows the program to find an element using Name (*By.name*):

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByName {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.exe");
        WebDriver driver = new ChromeDriver();

        driver.get("https://demoqa.com/automation-practice-form");

        WebElement element = driver.findElement (By.name("gender"));
        if(element != null) {
            System.out.println("Element found by Name");
        }
    }
}
```

The program gives the following output.

```
Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 36377
Only local connections are allowed.
[1600521397.320][WARNING]: Timed out connecting to Chrome, retrying...
Sep 19, 2020 6:46:39 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Element found by Name
```

The above program finds an element in Selenium using the name. We provide the name of the element that we can have to search as an argument to the *By object* in the '*findElement()*' call.

How to find an element using the attribute "class name" in Selenium?

Here the value of the **"class"** attribute is passed as the locator. This strategy is mostly used to find multiple elements that use similar CSS *classes*.

The locator strategy '*By Class Name*' finds the elements on the web page based on the **CLASS attribute value**. The strategy accepts a parameter of type String. The general syntax with the *Class name strategy* is given by:

```
List<WebElement> webList = driver.findElements(By.className(<Element_CLASSNAME>)) ;
```

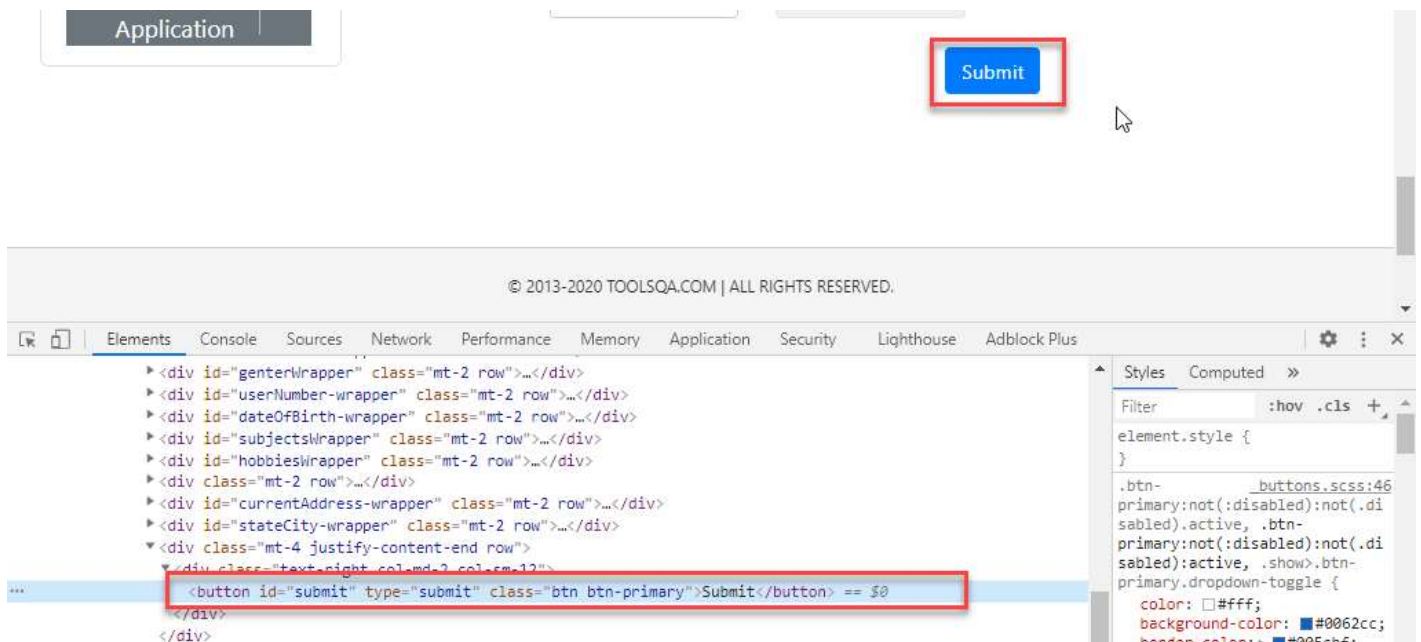
or

```
WebElement elm = driver.findElement(By.className(<Element_CLASSNAME>)) ;
```

The first syntax is to obtain a list of matching elements based on *Class Name* while the second syntax is to get only one matching element.

In case the element has many classes, then this strategy will match each of the classes.

Consider the following element (*submit button*) on [DemoQAAutomationPracticeForm](#) :



The corresponding command for finding the element marked above is:

```
WebElement parentElement = driver.findElement(By.className("button"));
parentElement.submit();
```

Note: Finding elements using class name strategy is helpful when we end up with non-unique IDs and names. That time we just go for the Class Name strategy and try to find the elements. When we use the Class Name strategy, once Selenium finds the particular class, it then looks for ID in the specified class.

The program to find an element using By.className is as below:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByClassName {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.exe");

        WebDriver driver = new ChromeDriver();

        driver.get("https://demoqa.com/automation-practice-form");

        WebElement parentElement = driver.findElement (By.className("button"));

        if(parentElement != null) {
            System.out.println("Element found by ClassName");
        }
    }
}
```

This program gives the following output.

```
Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 47316
Only local connections are allowed.
[1600524093.979][WARNING]: Timed out connecting to Chrome, retrying...
Sep 19, 2020 7:31:36 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Element found by ClassName
```

In this program, we have provided a *class name* **"button"** as a *By object* argument in the `findElement()` call. It scans the page and returns an element with `className = "button"`.

How to find an element using the attribute "HTML tag name" in Selenium?

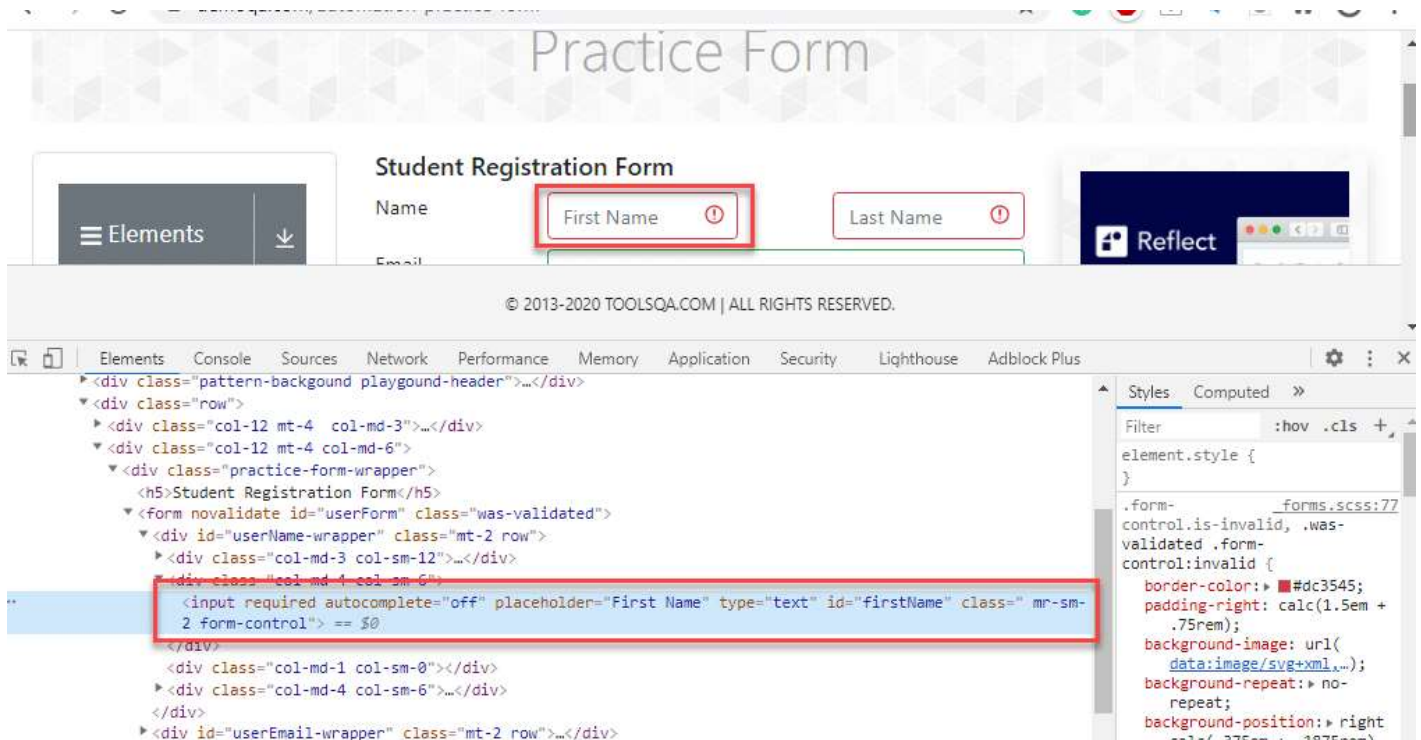
The **Tag Name** strategy uses an *HTML* tag name to locate the element. We use this approach rarely and we use it only when we are not able to find elements using any other strategies.

The **value of the TAG attribute** is a *String type* parameter. The syntax of the `findElement()` method using this strategy is as below.

```
WebElement elem = driver.findElement(By.tagName("Element_TAGNAME"));
```

As already mentioned, note that this strategy is not very popular and we use it only when there is no other alternative to locate the element.

As an example consider the following element on [DemoQAAutomationPracticeForm](#) :



The corresponding command for the above element (*input tag*) is as below:

```
WebElement element = driver.findElement(By.tagName("input"));
```

Following is the program to find elements using **By.tagName** object.

```
import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByTagName {

    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.exe");

        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/automation-practice-form");
        WebElement element = driver.findElement (By.tagName("input"));
        if(element != null) {
            System.out.println("Element found by tagName");
        }
    }
}
```

The output of this program is as seen below.

```
Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 47026
Only local connections are allowed.
[1600523415.961][WARNING]: Timed out connecting to Chrome, retrying...
Sep 19, 2020 7:20:17 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: CSS
Element found by TagName
```

Above program uses **By.tagName** object in 'findElement()' call to find an element based on **tagName = "input"**.

How to find an element using the "CSS Selector" in Selenium?

We can also use the **CSS Selector strategy** as an argument to *By* object when finding the element. Since *CSS Selector* has native browser support, sometimes the *CSS Selector* strategy is faster than the *XPath* strategy.

Again we will choose an element from the page [DemoQAAutomationPracticeForm](#) :

The screenshot displays a web browser window with a 'Student Registration Form'. The form includes fields for Name (First Name and Last Name), Email (name@example.com), and Gender (Male, Female, Other). The 'First Name' field is highlighted with a red box. Below the form, the browser's developer tools are open, showing the 'Elements' tab. The HTML structure is expanded, revealing the following code for the 'First Name' input field:

```
<input required autocomplete="off" placeholder="First Name" type="text" id="firstName" class="mr-sm-2 form-control"> == $0
```

The *CSS Selector* for the above input field is **#firstName**. So the corresponding command to find element by *CSS Selector* is:

```
WebElement inputElem = driver.findElement(By.cssSelector("input[id = 'firstName']"));
inputElem.SendKeys("demoQA");
```

The following program shows how to find elements using the **By.cssSelector** construct.

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByCssSelector {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/automation-practice-form");
        WebElement inputElem = driver.findElement (By.cssSelector("input[id = 'first
        if(inputElem != null) {
            System.out.println("Element found by cssSelector");
        }
    }
}

```

The program gives the following output.

```

Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 20469
Only local connections are allowed.
[1600522040.604][WARNING]: Timed out connecting to Chrome, retrying...
Sep 19, 2020 6:57:22 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Element found by cssSelector

```

The above program finds an element using *CSS Selector* for the field '*firstName*' by using the **By.cssSelector** locator strategy. The program returns an element having the specified CSS *selector*.

How to find an element using the "XPath" in Selenium?

This strategy is the most popular one for finding elements. Using this strategy we navigate through the structure of the **HTML** or **XML** documents.

This strategy accepts a String type parameter, *XPath Expression*. The general syntax of using this strategy is as given below:

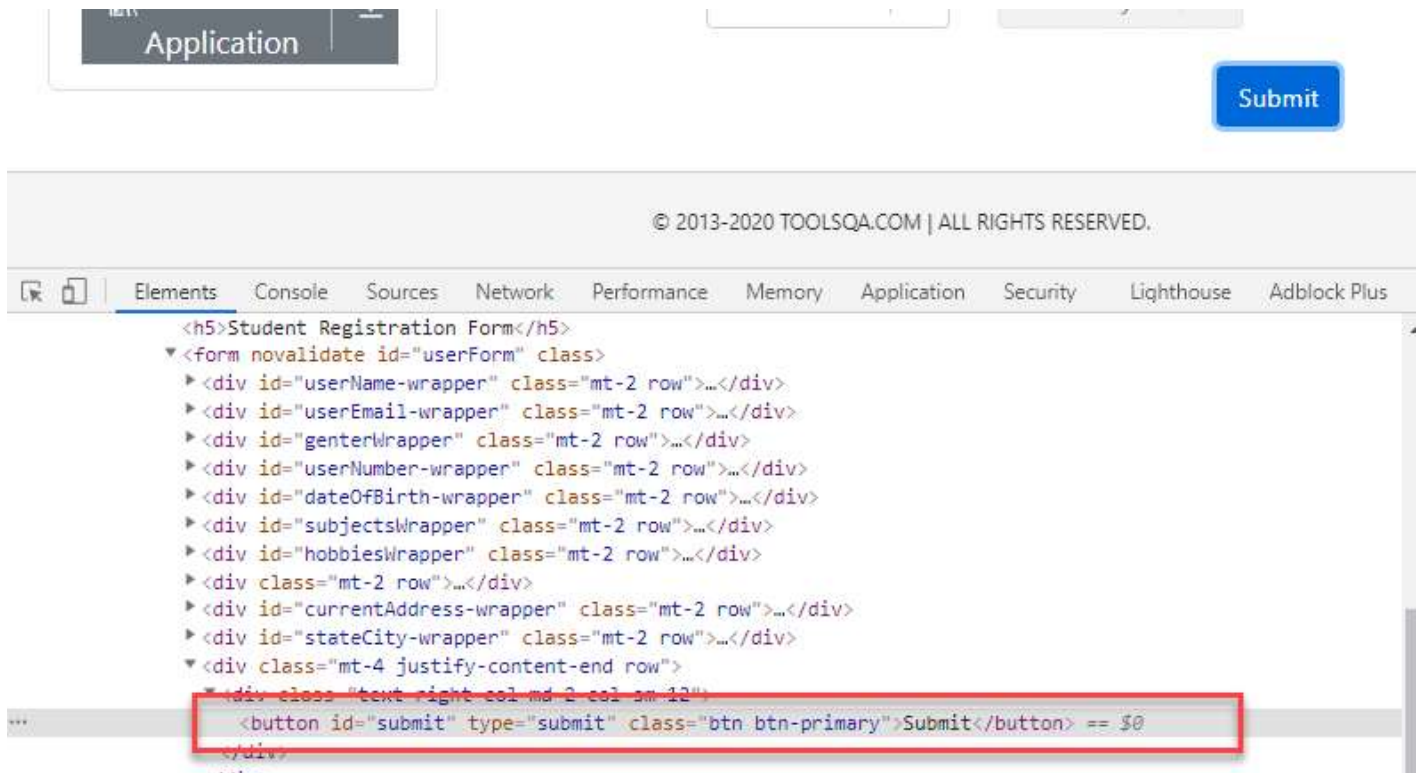
```

WebElement elem = driver.findElement(By.xpath("ElementXPathExpression"));

```

Using **XPath** we can locate a single element using various ways. It provides many different and easy ways to locate elements.

As an example let us take the following element in the page [DemoQAAutomationPracticeForm](#) :



The *XPath* for the above button element is `[@id="submit"]`. Please refer [How To Inspect Elements](#) using Web Inspector for more information. So we use it in the `findElement()` command as below:

```
WebElement buttonLogin = driver.findElement(By.xpath("//button[@id = 'submit']"));
```

A program to find elements using **By.XPath** is as follows:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByXPath {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver.exe");
        WebDriver driver = new ChromeDriver();

        driver.get("https://demoqa.com/automation-practice-form");
        WebElement buttonSubmit = driver.findElement(By.xpath("//button[@id = 'submit']"));
        if(buttonSubmit != null) {
            System.out.println("Element found by xpath");
        }
    }
}
```

This program displays the following output.

```
Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 26041
Only local connections are allowed.
[1600522407.202][WARNING]: Timed out connecting to Chrome, retrying...
Sep 19, 2020 7:03:29 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Element found by xpath
```

Here we have provided the **XPath** of the **"submit "** button as an argument to the **By.xpath** locator. The program returns the element that matches the specified **XPath**.

How to find an element using the "Link Text/Partial Link Text" in Selenium?

This strategy finds links within the webpage. It specially finds elements having *links*. This means we can use this strategy to find elements of **"a "** (*links*) tags that have matching link names or partial link names.

The strategy accepts the **value of the LINKTEXT attribute** as a String type parameter.

The syntax of findElement using this strategy is as seen below.

```
WebElement elem = driver.findElement(By.linkText("Element LinkText"));
```

The above syntax is for finding elements using full link text. This is used when we know the link text that is used within the **anchor (a)** tag.

We can also use the partial link and find elements. Following is the syntax:

```
WebElement elem = driver.findElement(By.partialLinkText("ElementLinkText"));
```

Here we can provide partial link names.

As an example, we can take the following element ([DemoQAHomeLink](#)). We have highlighted the element as shown below:



We can use a link strategy if the targetted text is link text. So for the above link element, the `findElement()` command for the *link* and *partial link* strategy is as follows:

```
WebElement element = driver.findElement(By.linkText("Home"));

//Or can be identified as
WebElement element = driver.findElement(By.partialLinkText("HomehY"));
```

In the first example, we use `By.linkText` strategy and provide the entire '*linkname*'. This will look for a link with the "**Home**" word. In the second example, we use `By.partialLinkText` and only provide a part of '*linkname*' ('HomehY'). Since this is a partial link, it will look for links starting with 'HomehY'. As shown above, there is a link 'HomehYtil' on the page. So **`By.partialLinkText`** will find this link.

Let us implement a code to find element using `By.linkText/By.partialLinkText`.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FindElementByLinkTextAndPartialLinkText {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "C:/testSelenium/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/links");
        WebElement element = driver.findElement (By.linkText("Home"));

        if(element != null) {
            System.out.println("Element found by LinkText");
        }
    }
}
```

```

        element= driver.findElement (By.partialLinkText("HomehY"));

        if(element!= null) {
            System.out.println("Element found by PartialLinkText");
        }
    }
}

```

The Output:

```

Starting ChromeDriver 2.41.578737 (49da6702b16031c40d63e5618de03a32ff6c197e) on port 19521
Only local connections are allowed.
[1600522742.790][WARNING]: Timed out connecting to Chrome, retrying...
Sep 19, 2020 7:09:04 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
Element found by LinkText
Element found by PartialLinkText

```

This program finds an element using *By.linkText* and *By.partialLinkText* locator. When *By.linkText* is specified, the '*findElement*' matches the entire *linkText* specified. It matches the partial text when *By.partialLinkText* is specified.

Difference between find Element and find Elements in Selenium

Let us discuss some differences between *findElement()* and *findElements()* methods provided by *Selenium WebDriver*.

| <i>FindElement()</i> | <i>FindElements()</i> |
|---|---|
| Returns the first web element out of all the elements found by the same locator. | Finds and returns a list of web elements. |
| This method finds only one element. | This method returns the collection of elements matching the locator. |
| If no element matches the locator, an exception " <i>NoSuchElementException</i> " is thrown. | No exception is thrown if no matching elements are found. Simply returns an empty list. |
| NNo indexing required since only one element is returned. | Each web element is indexed starting from 0. |

Key TakeAways

We need to find or locate web elements within the webpage so that we can perform actions on these elements.

Additionally, we have two methods `find Element` and `find Elements` methods in Selenium WebDriver using which we can locate elements.

Also, the method `findElement()` in Selenium returns a unique web element from the webpage.

The method `findElements()` in Selenium returns a list of web elements

*Lastly, to locate or find the elements each method uses a locator strategy which is provided via an argument to the **"By"** object. These strategies are `by Id`, `by Name`, `by Class Name`, `by XPath`, `by link`, etc.*