# Element locators

Before starting with selenium test automation learning, You should be aware of different ways of locating an elements in webdriver for web application. Locators allow us to find elements on a page that can be used in our tests. In this chapter we will learn about locators.

Selenium supports many different element locators which are as follows

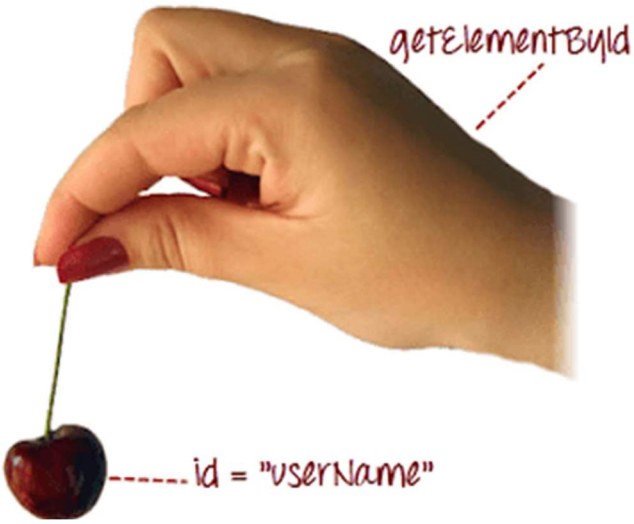
1. By ID
2. By ClassName
3. By Name
4. By TagName
5. By Link Text
6. By Partial Link Text
7. By cssSelector
8. By XPath

We will use Find Element and Find Elements methods with locator to locate element on the web page. You can use various tools to locate the web elements. We have already covered those tools in last chapter.

Lets understand the locators one by one.

**1) By ID**

This is the most common way of locating elements since ID's are supposed to be unique for each element. This mechanism returns the location of web element who's ID is matching with the specified ID in your script. If no element has a matching id attribute, NoSuchElementException will be raised.



Example: If you have a webpage like following


```
1  <html>
2   <body>
3    <form id="login">
4      <label>Username</label>
5      <input id="username" type="text" name="login" />
6      <br /><br />
7      <label>Password</label>
8      <input id="password" type="password" name="password" />
9      <br /> <br />
10     <input type="submit" name="signin" value="Login" />
11   </form>
12  </body>
13 </html>
```

You can easily select an element with the help of ID locator from the above example:

id = “username”
id = “password”

Use the above ID in your selenium script :

```
driver.findElement(By.id("username"));
driver.findElement(By.id("password"));
```

Please note: Avoid using this technique when the ID's are not unique and randomly generated.

**2) By ClassName**

Use *By ClassName* locator when you want to locate an element by class attribute name. In this strategy, the location of the web element is returned who's className attribute is matching with the specified class Name in your script. If no element has a matching class attribute name, a NoSuchElementException will be raised.

Example:
For instance, consider this page source:

```
1 <html>
2   <body>
3     <input id="FirstName" type="text" class="fname" />
4   </body>
5 <html>
```

The *input* tag element can be located like this:
driver.findElement(By.className("fname"));

**3) By Name**
You can use this locator when you know the name attribute of an element. This mechanism returns the location of web element who's name attribute is matching with the specified name attribute in your script. If no element has a matching name attribute, a NoSuchElementException will be raised.

Example:
For instance, consider this page source:

```
1 <html>
2  <body>
3   <form id="loginForm">
4     <input name="username" type="text" />
5     <input name="password" type="password" />
6     <input name="continue" type="submit" value="Login" />
7   </form>
8 </body>
9 <html>
```

The username & password elements can be located like this:
driver.findElement(By.name('username'));
driver.findElement(By.name('password'));

**4) By TagName**
Use this locator when you want to locate an element by tag name. In this mechanism, the location of the web element can be identified by matching the tag name. If no element has a matching tag name, a NoSuchElementException will be raised.

Example:
For instance, consider this page source:

```
1 <html>
2  <body>
3    <h1>Welcome</h1>
4    <p>Start here..</p>
```

```
5  </body>
6 <html>
```

The heading (h1) element can be located like this:
driver.findElements(By.tagName(“h1”));

**5) By LinkText**
Use this locator when you know link text used within an anchor tag. In this mechanism, the location of web element is returned who's link text value is matching with the link text specified in your script. If no element has a matching link text attribute, a NoSuchElementException will be raised.

Example:
For instance, consider this page source:

```
1 <html>
2  <body>
3     <p>Please confirm ..</p>
4     <a href="continue.html">Continue</a>
5     <a href="cancel.html">Cancel</a>
6  </body>
7 <html>
```

The continue.html link can be located like this:
driver.findElement(By.LinkText('Continue'));

**6) By PartialLinkText**
Use this locator when you want to select link element which contains text matching the specified partial link text.

Example:
For instance, consider this page source:

```
1 <html>
2  <body>
3     <p>Please confirm ..</p>
4     <a href="continue.html">Continue to website</a>
5     <a href="cancel.html">Cancel</a>
6  </body>
7 <html>
```

The continue.html link can be located like this:
driver.findElement(By.partialLinkText("to website"));

**7) By cssSelector**

CSS Selectors are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selector is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.

Example:
For instance, consider this page source2222:

```
1 <html>
2  <body>
3     <input type="email" class="inputtext" name="email" id="email" value="" tabindex="1">
4  </body>
5 <html>
```

Locating by CSS Selector - Tag and ID
driver.findElement(By.cssSelector("input[id='email']"));

**8) By Xpath**

1. It is most popular and best way to locate element in selenium webdriver.
2. Xpath is used to locate a web element based on its XML path.
3. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data.
4. It stores data in a key-value pair which is very much similar to HTML tags. Both being mark up languages and since they fall under the same umbrella, xpath can be used to locate HTML elements.
5. The fundamental behind locating elements using Xpath is the traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

## Xpath can be created in two ways:

**a) Relative Xpath:**
Relative Xpath begins from the current location and is prefixed with "//".
For example: //span[@class='Name']

**b) Absolute Xpath:**
Absolute Xpath begins with a root path and is prefixed with a "/".
For example: /html/body/div/div[@id='Address']

Absolute Xpath can break if the HTML is changed slightly. So it is less recommended.

Pros of using Xpath: It can access almost any element, even those without class, name, or id attributes.

Cons of using Xpath: It is the most complicated method of identifying elements because of too many

different rules and considerations.

Example:
For instance, consider this page source:

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
 </body>
<html>
```

The form element can be located as below

```
driver.findElement(By.xpath("/html/body/form[1]"));
driver.findElement(By.xpath("//form[1]"));
driver.findElement(By.xpath("//form[@id='loginForm']"));
```

The username element can be located as below

```
driver.findElement("//form[input/@name='username']"));
driver.findElement("//form[@id='loginForm']/input[1]"));
driver.findElement("//input[@name='username']"));
```

The “Clear” button element can be located as below

```
driver.findElement("//input[@name='continue'][@type='button']"));
driver.findElement("//form[@id='loginForm']/input[4]"));
```

Following are the ways to write dynamic Xpath:1) start-with method:
1) startwith method: You can use the *start-with* in xpath to locate an attribute value that starts with a certain text.

For example, assume you have the following link on the page:

```
<a href="mylink_somerandomstuff">link text</a>
```

Then you can use the following xpath to find links that have an href that starts with 'mylink'

```
driver.findElement("//a[starts-with(@href, "mylink")]"));
```

2) contains method: Similarly you can use *contains* method to locate an attribute. Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically.

```
driver.findElement("//a[contains(@href, "somerandomstuff")]"));
```