4. Locating Elements

There are various strategies to locate elements in a page. You can use the most appropriate one for your case. Selenium provides the following methods to locate elements in a page:

* *find\_element\_by\_id*
* *find\_element\_by\_name*
* *find\_element\_by\_xpath*
* *find\_element\_by\_link\_text*
* *find\_element\_by\_partial\_link\_text*
* *find\_element\_by\_tag\_name*
* *find\_element\_by\_class\_name*
* *find\_element\_by\_css\_selector*

**To find multiple elements (these methods will return a list):**

* *find\_elements\_by\_name*
* *find\_elements\_by\_xpath*
* *find\_elements\_by\_link\_text*
* *find\_elements\_by\_partial\_link\_text*
* *find\_elements\_by\_tag\_name*
* *find\_elements\_by\_class\_name*
* *find\_elements\_by\_css\_selector*

Apart from the public methods given above, there are two private methods which might be useful with locators in page objects. These are the two private methods: *find\_element* and *find\_elements*.

Example usage:

**from** **selenium.webdriver.common.by** **import** By

driver.find\_element(By.XPATH, '//button[text()="Some text"]')

driver.find\_elements(By.XPATH, '//button')

These are the attributes available for *By* class:

ID = "id"

XPATH = "xpath"

LINK\_TEXT = "link text"

PARTIAL\_LINK\_TEXT = "partial link text"

NAME = "name"

TAG\_NAME = "tag name"

CLASS\_NAME = "class name"

CSS\_SELECTOR = "css selector"

4.1. Locating by Id

Use this when you know *id* attribute of an element. With this strategy, the first element with the *id* attribute value matching the location will be returned. If no element has a matching *id* attribute, a NoSuchElementException will be raised.

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" />

</form>

</body>

<html>

The form element can be located like this:

login\_form = driver.find\_element\_by\_id('loginForm')

4.2. Locating by Name

Use this when you know *name* attribute of an element. With this strategy, the first element with the *name* attribute value matching the location will be returned. If no element has a matching *name* attribute, a NoSuchElementException will be raised.

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 username & password elements can be located like this:

username = driver.find\_element\_by\_name('username')

password = driver.find\_element\_by\_name('password')

This will give the “Login” button as it occurs before the “Clear” button:

**continue** = driver.find\_element\_by\_name('continue')

4.3. Locating by XPath

XPath is the language used for locating nodes in an XML document. As HTML can be an implementation of XML (XHTML), Selenium users can leverage this powerful language to target elements in their web applications. XPath extends beyond (as well as supporting) the simple methods of locating by id or name attributes, and opens up all sorts of new possibilities such as locating the third checkbox on the page.

One of the main reasons for using XPath is when you don’t have a suitable id or name attribute for the element you wish to locate. You can use XPath to either locate the element in absolute terms (not advised), or relative to an element that does have an id or name attribute. XPath locators can also be used to specify elements via attributes other than id and name.

Absolute XPaths contain the location of all elements from the root (html) and as a result are likely to fail with only the slightest adjustment to the application. By finding a nearby element with an id or name attribute (ideally a parent element) you can locate your target element based on the relationship. This is much less likely to change and can make your tests more robust.

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 elements can be located like this:

login\_form = driver.find\_element\_by\_xpath("/html/body/form[1]")

login\_form = driver.find\_element\_by\_xpath("//form[1]")

login\_form = driver.find\_element\_by\_xpath("//form[@id='loginForm']")

1. Absolute path (would break if the HTML was changed only slightly)
2. First form element in the HTML
3. The form element with attribute named *id* and the value *loginForm*

The username element can be located like this:

username = driver.find\_element\_by\_xpath("//form[input/@name='username']")

username = driver.find\_element\_by\_xpath("//form[@id='loginForm']/input[1]")

username = driver.find\_element\_by\_xpath("//input[@name='username']")

1. First form element with an input child element with attribute named *name* and the value *username*
2. First input child element of the form element with attribute named *id* and the value *loginForm*
3. First input element with attribute named ‘name’ and the value *username*

The “Clear” button element can be located like this:

clear\_button = driver.find\_element\_by\_xpath("//input[@name='continue'][@type='button']")

clear\_button = driver.find\_element\_by\_xpath("//form[@id='loginForm']/input[4]")

1. Input with attribute named *name* and the value *continue* and attribute named *type* and the value *button*
2. Fourth input child element of the form element with attribute named *id* and value *loginForm*

These examples cover some basics, but in order to learn more, the following references are recommended:

* [W3Schools XPath Tutorial](https://www.w3schools.com/xml/xpath_intro.asp)
* [W3C XPath Recommendation](http://www.w3.org/TR/xpath)
* [XPath Tutorial](http://www.zvon.org/comp/r/tut-XPath_1.html) - with interactive examples.

There are also a couple of very useful Add-ons that can assist in discovering the XPath of an element:

* [XPath Checker](https://addons.mozilla.org/en-US/firefox/addon/xpath-checker/) - suggests XPath and can be used to test XPath results.
* [Firebug](https://addons.mozilla.org/en-US/firefox/addon/firebug/) - XPath suggestions are just one of the many powerful features of this very useful add-on.
* [XPath Helper](https://chrome.google.com/webstore/detail/hgimnogjllphhhkhlmebbmlgjoejdpjl) - for Google Chrome

4.4. Locating Hyperlinks by Link Text

Use this when you know link text used within an anchor tag. With this strategy, the first element with the link text value matching the location will be returned. If no element has a matching link text attribute, a NoSuchElementException will be raised.

For instance, consider this page source:

<html>

<body>

<p>Are you sure you want to do this?</p>

<a href="continue.html">Continue</a>

<a href="cancel.html">Cancel</a>

</body>

<html>

The continue.html link can be located like this:

continue\_link = driver.find\_element\_by\_link\_text('Continue')

continue\_link = driver.find\_element\_by\_partial\_link\_text('Conti')

4.5. Locating Elements by Tag Name

Use this when you want to locate an element by tag name. With this strategy, the first element with the given tag name will be returned. If no element has a matching tag name, a NoSuchElementException will be raised.

For instance, consider this page source:

<html>

<body>

<h1>Welcome</h1>

<p>Site content goes here.</p>

</body>

<html>

The heading (h1) element can be located like this:

heading1 = driver.find\_element\_by\_tag\_name('h1')

4.6. Locating Elements by Class Name

Use this when you want to locate an element by class attribute name. With this strategy, the first element with the matching class attribute name will be returned. If no element has a matching class attribute name, a NoSuchElementException will be raised.

For instance, consider this page source:

<html>

<body>

<p class="content">Site content goes here.</p>

</body>

<html>

The “p” element can be located like this:

content = driver.find\_element\_by\_class\_name('content')

4.7. Locating Elements by CSS Selectors

Use this when you want to locate an element by CSS selector syntax. With this strategy, the first element with the matching CSS selector will be returned. If no element has a matching CSS selector, a NoSuchElementException will be raised.

For instance, consider this page source:

<html>

<body>

<p class="content">Site content goes here.</p>

</body>

<html>

The “p” element can be located like this:

content = driver.find\_element\_by\_css\_selector('p.content')