Summary for locating elements

| **Variation** | **Description** | **Sample** |
| --- | --- | --- |
| **By.className** | finds elements based on the value of the “class” attribute | findElement(By.className(“someClassName”)) |
| **By.cssSelector** | finds elements based on the driver’s underlying CSS Selector engine | findElement(By.cssSelector(“input#email”)) |
| **By.id** | locates elements by the value of their “id” attribute | findElement(By.id(“someId”)) |
| **By.linkText** | finds a link element by the exact text it displays | findElement(By.linkText(“REGISTRATION”)) |
| **By.name** | locates elements by the value of the “name” attribute | findElement(By.name(“someName”)) |
| **By.partialLinkText** | locates elements that contain the given link text | findElement(By.partialLinkText(“REG”)) |
| **By.tagName** | locates elements by their tag name | findElement(By.tagName(“div”)) |
| **By.xpath** | locates elements via XPath | findElement(By.xpath(“//html/body/div/table/tbody/tr/td[2]/table/ tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/td[3]/ form/table/tbody/tr[5]”)) |

**Clicking on an Element**

Clicking is perhaps the most common way of interacting with web elements. The click() method is used to simulate the clicking of any element. The following Selenium Java example shows how click() was used to click on Mercury Tours’ “Sign-In” button.

[Clicking on an Element](https://www.guru99.com/images/image017(2).png)

Following things must be noted when using the click() method.

* It does not take any parameter/argument.
* The method automatically waits for a new page to load if applicable.
* The element to be clicked-on, must be visible (height and width must not be equal to zero).

**Get Commands**

Get commands fetch various important information about the page/element. Here are some important “get” commands you must be familiar with.

| **Commands** | **Usage** |
| --- | --- |
| **get()**  Sample usage: | * It automatically opens a new browser window and fetches the page that you specify inside its parentheses. * It is the counterpart of Selenium IDE’s “open” command. * The parameter must be a **String** object. |
| **getTitle()**  Sample usage: | * Needs no parameters * Fetches the title of the current page * Leading and trailing white spaces are trimmed * Returns a null string if the page has no title |
| **getPageSource()**  Sample usage: | * Needs no parameters * Returns the source code of the page as a String value |
| **getCurrentUrl()**  Sample usage: | * Needs no parameters * Fetches the string representing the current URL that the browser is looking at |
| **getText()**  Sample usage: | * Fetches the inner text of the element that you specify |

**Navigate commands**

These commands allow you to refresh,go-into and switch back and forth between different web pages.

|  |  |
| --- | --- |
| **navigate().to()**  Sample usage: | * It automatically opens a new browser window and fetches the page that you specify inside its parentheses. * It does exactly the same thing as the get() method. |
| **navigate().refresh()**  Sample usage: | * Needs no parameters. * It refreshes the current page. |
| **navigate().back()**  Sample usage: | * Needs no parameters * Takes you back by one page on the browser’s history. |
| **navigate().forward()**  Sample usage: | * Needs no parameters * Takes you forward by one page on the browser’s history. |

**Closing and Quitting Browser Windows**

|  |  |
| --- | --- |
| **close()**  Sample usage: | * Needs no parameters * It closes only the browser window that WebDriver is currently controlling. |
| **quit()**  Sample usage: | * Needs no parameters * It closes all windows that WebDriver has opened. |

Close-up of different types of windows

Description automatically generated

**Waits**

There are two kinds of waits.

1. Implicit wait – used to set the default waiting time throughout the program
2. Explicit wait – used to set the waiting time for a particular instance only

driver.get("http://demo.guru99.com/selenium/deprecated.html");

driver.switchTo().frame("classFrame");

driver.findElement(By.linkText("Deprecated")).click();

driver.close();

driver.get("http://jsbin.com/usidix/1");

driver.findElement(By.cssSelector("input[value=\"Go!\"]")).click();

alertMessage = driver.switchTo().alert().getText();

driver.switchTo().alert().accept();

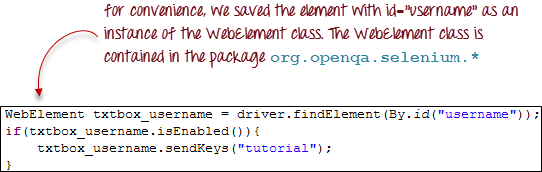
System.out.println(alertMessage);

driver.quit();

**Conditions**

Following methods are used in conditional and looping operations —

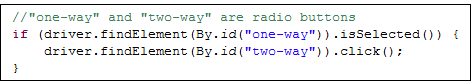
* **isEnabled()** is used when you want to verify whether a certain element is enabled or not before executing a command.

[](https://www.guru99.com/images/image043(1).png)

* **isDisplayed()** is used when you want to verify whether a certain element is displayed or not before executing a command.

[Conditions isDisplayed()](https://www.guru99.com/images/image044(1).png)

* **isSelected()** is used when you want to verify whether a certain **check box, radio button, or option in a drop-down box** is selected. It does not work on other elements.

[](https://www.guru99.com/images/image045.png)

**Using ExpectedConditions**

The ExpectedConditions class offers a wider set of conditions that you can use in conjunction with WebDriverWait’s until() method.

Below are some of the most common ExpectedConditions methods.

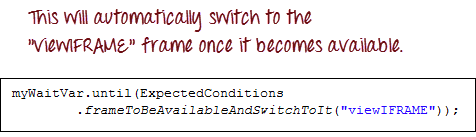
* **alertIsPresent()**– waits until an alert box is displayed.

[Using ExpectedConditions alertIsPresent()](https://www.guru99.com/images/image046(1).png)

* **elementToBeClickable()** – Waits until an element is visible and, at the same time, enabled. The sample Selenium Code below will wait until the element with id=”username” to become visible and enabled first before assigning that element as a WebElement variable named “txtUserName”.

[Using ExpectedConditions](https://www.guru99.com/images/image047(1).png)

* **frameToBeAvailableAndSwitchToIt()**– Waits until the given frame is already available, and then automatically switches to it.

[](https://www.guru99.com/images/image048(1).png)

**Catching Exceptions**

When using isEnabled(), isDisplayed(), and isSelected(), WebDriver assumes that the element already exists on the page. Otherwise, it will throw a **NoSuchElementException**. To avoid this, we should use a try-catch block so that the program will not be interrupted.

WebElement txtbox\_username = driver.findElement(By.id("username"));

try{

if(txtbox\_username.isEnabled()){

txtbox\_username.sendKeys("tutorial");

}

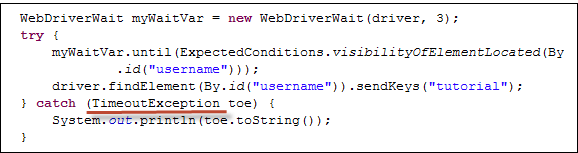
}

catch(NoSuchElementException nsee){

System.out.println(nsee.toString());

}

If you use explicit waits, the type of exception that you should catch is the “TimeoutException”.

[](https://www.guru99.com/images/image050(2).png)

**Summary**

* To start using the WebDriver API, you must import at least these two packages.
* org.openqa.selenium.\*
* org.openqa.selenium.firefox.FirefoxDriver
* The get() method is the equivalent of Selenium IDE’s “open” command.
* Locating elements in WebDriver is done by using the findElement() method.
* The following are the available options for locating elements in WebDriver:
* **By.className**
* **By.cssSelector**
* **By.id**
* **By.linkText**
* **By.name**
* **By.partialLinkText**
* **By.tagName**
* **By.xpath**
* The By.cssSelector() **does not** support the **“contains”** feature.
* You can instantiate an element using the WebElement class.
* Clicking on an element is done by using the click() method.
* **WebDriver provides these useful get commands:**
* get()
* getTitle()
* getPageSource()
* getCurrentUrl()
* getText()
* **WebDriver provides these useful navigation commands**
* navigate().forward()
* navigate().back()
* navigate().to()
* navigate().refresh()
* The close() and quit() methods are used to close browser windows. Close() is used to close a single window; while quit() is used to close all windows associated to the parent window that the WebDriver object was controlling.
* The switchTo().frame() and switchTo().alert() methods are used to direct WebDriver’s focus onto a frame or alert, respectively.
* Implicit waits are used to set the waiting time throughout the program, while explicit waits are used only on specific portions.
* You can use the isEnabled(), isDisplayed(),isSelected(), and a combination of **WebDriverWait** and **ExpectedConditions** methods when verifying the state of an element. However, they do not verify if the element does not exists.
* When isEnabled(), isDisplayed(),or isSelected() was called while the element was not existing, WebDriver will throw a **NoSuchElementException**.
* When WebDriverWait and ExpectedConditions methods were called while the element was not existing, WebDriver would throw a TimeoutException.