**Basics**

* First, we need the **WebDriver object**. In MrChecker’s Selenium Module, it can be used by calling the **getDriver()** method. Notice that it follows java’s singleton pattern and only creates one object. This WebDriver object represents the whole navigator and we will call methods on it to find elements in it or interact with those elements.
* **findElementDynamic**: Tries to find an element by a speficied criterion. It expects a By object as argument, and returns a WebElement object. It can be used like this:

getDriver().findElementDynamic(By.someCriterion(criterion));

or

By searchElement = By.someCriterion(criterion);

getDriver().findElementDynamic(searchElement);

The second one is prefered as it is POM.  
This method has an implicit wait included in it, unlike Selenium’s *findElement*, so it will try to wait for the element to show up and if that never happens it will raise an exception after waiting for 20 seconds.

* Some workable **By** types:
  + **By.name(someName)**
  + **By.tagName(someTagName)**: For example By.tagName("div") or By.tagName("button").
  + **By.className(someClassName)**: For example By.className("mat-checkbox").
  + **By.id(someId)**
  + **By.cssSelector(someCssSelector)**: For example By.cssSelector("button.mat-icon-button:nth-child(6)") or By.cssSelector("input[formcontrolname='email']").
  + **By.xpath(someXpath)**: For example: By.xpath("//button[@class='text-upper property-text-bold mat-button mat-accent']").
* **findElementDynamics**\_: Unlike the previous method, this one returns a list of WebElements. It’s useful when two elements share the same class or a certain property and you need to stock both of them. It raises an exception if no elements are found.
* **myWebElement.click()**: It performs a click in the web element specified, as if the user was performing the click.
* **myWebElement.sendKeys(someString)**: It takes a string as argument and tries to type those keys in the web element. It’s typically used to fill inputs, but it can be used to send specific keys like **Enter** or **Backspace**, so it can be used to activate a button (by using Enter) like if we were clicking it.
* **myWebElement.getAttribute(attributeName)**: It returns the value of that attribute.
* **myWebElement.getText()**: Returns ALL the text contained in that Web element.
* **findElementQuietly**: Used in the same way as **findElementDynamic**, but this one doesn’t raise an exception if element isn’t found.

After the element was found, we need to perform an action on it, for example:

* **sendKeys**: For example element.sendkeys("hello"). It sends those keys to the specified element. When using Selenium on Angular apps, there is sometimes a bug where because keys are being sent too fast, one or more of those letters simply won’t show up, and therefore any logins will fail. The solution we provided for it is **sendKeysWithCheck** which ensures Selenium didn’t miss a letter. It’s in *src/test/java/com/devonfw/mts/common/utils/Utils.java*
* **click**: For example element.click(). It performs a click in the middle of the specified element.

There are some more, but these ones are the most used.