Selenium common Methods:

=======================

**1- IsElementPresent/Text Present  function in Selenium WebDriver :**

In general , we found issues related to element not found in failed cases in general. So here we have three ways to identify element on page.

This selenium command/ method is going to give you one reusable function to use for elemenet on screen.

**Finding elements by using function that take argument of By class**  
  
private boolean isElementPresent(WebDriver driver, By by)  
try{  
driver.findElement(by);  
return true;  
}  
catch(Exception e)  
{  
return false;  
}  
}

**Using the size to decide whether element is there or not**  
  
if(driver.findElements(Locator).size()>0  
{  
return true  
}else  
{  
return false  
}  
}

**Finding the text using the PageSource**driver.PageSource.Contains("TEXT that you want to see on the page");  
  
**For more detail read** : [Handling Element Not Found Exception In Selenium WebDriver](https://abodeqa.com/handling-element-not-found-exception-in-selenium-webdriver/)

**2- Finding WebElement  by using various locators in WebDriver**

In this we have listed all the locator methods available in Selenium WebDriver, which can be used to identify any of the element on webpage.

**Using ID**  WebElement welement = driver.findElement(By.id(“Id from webpage”)); // by using id

**Using Name**  WebElement welement = driver.findElement(By.name(“Name of WebElement”)); //by using name

**Using Tag Name**  WebElement welement = driver.findElement(By.tagName(“tag name”)); // by using tag

**Using Xpath**  WebElement welement = driver.findElement(By.xpath(“xpath of  webElement”)); //

**Using CSS**  WebElement welement = driver.findElement(By.CSS(“CSS locator path”));

**Using LinkText**  WebElement welement = driver.findElement(By.LinkText(“LinkText”));  
Read these articles:  
[XPath in Selenium](https://abodeqa.com/xpath-in-automation-mainly-in-selenium/)

**3- Working With Pop-ups In Selenium WebDriver:**

**This function will fetch message**on Pop-up  
  
public static String getPopupMessage(final WebDriver driver) {  
String message = null;  
try {  
Alert alert = driver.switchTo().alert();  
message = alert.getText();  
alert.accept();  
} catch (Exception e) {  
message = null;  
}  
System.out.println("message"+message);  
return message;  
}

**This function will Canceling pop-up in Selenium-WebDriver**

public static String cancelPopupMessageBox(final WebDriver driver) {  
String message = null;  
try {  
Alert alert = driver.switchTo().alert();  
message = alert.getText();  
alert.dismiss();  
} catch (Exception e) {  
message = null;  
}  
return message;  
}

**Inserting string in Text Field in Selenium-WebDriver**

public static void insertText(WebDriver driver, By locator, String value) {  
WebElement field = driver.findElement(locator);  
field.clear();  
field.sendKeys(value);  
}

**4- Reading ToolTip text in in Selenium-WebDriver**

In this  
public static String tooltipText(WebDriver driver, By locator){  
String tooltip = driver.findElement(locator).getAttribute("title");  
return tooltip;  
}

**5- Selecting Radio Button in Selenium-WebDriver**

public static void selectRadioButton(WebDriver driver, By locator, String value){ List select = driver.findElements(locator);  
for (WebElement element : select)  
{  
if (element.getAttribute("value").equalsIgnoreCase(value)){  
element.click();  
}  
}

**6- Selecting CheckBox in Selenium-WebDriver**

public static void selectCheckboxes(WebDriver driver, By locator,String value)  
{  
List abc = driver.findElements(locator);  
List list = new ArrayListArrays.asList(value.split(",")));  
for (String check : list){  
for (WebElement chk : abc){  
if(chk.getAttribute("value").equalsIgnoreCase(check)){  
chk.click();  
}}}}

**7- Selecting Dropdown in Selenium-WebDriver**

public static void selectDropdown(WebDriver driver, By locator, String value){  
new Select (driver.findElement(locator)).selectByVisibleText(value); }

**8- Selecting searched dropdown in Selenium-WebDriver**

public static void selectSearchDropdown(WebDriver driver, By locator, String value){  
driver.findElement(locator).click();  
driver.findElement(locator).sendKeys(value);  
driver.findElement(locator).sendKeys(Keys.TAB);  
}

**9- Uploading file using  Selenium-WebDriver**

public static void uploadFile(WebDriver driver, By locator, String path){  
driver.findElement(locator).sendKeys(path);  
}

**10- Downloading file in Selenium-WebDriver**

Here we will click on a link and will download the file with a predefined name at some specified location.  
public static void downloadFile(String href, String fileName) throws Exception{  
URL url = null;  
URLConnection con = null;  
int i;  
url = new URL(href);  
con = url.openConnection();  
// Here we are specifying the location where we really want to save the file.  
File file = new File(".//OutputData//" + fileName);  
BufferedInputStream bis = new BufferedInputStream(con.getInputStream());  
BufferedOutputStream bos = new BufferedOutputStream(  
new FileOutputStream(file));  
while ((i = bis.read()) != -1) {  
bos.write(i);  
}  
bos.flush();  
bis.close();  
}

**11- Handling multiple Pop ups**

read  [**Handling Multiple Windows in WebDriver**](https://abodeqa.com/2013/09/12/767/)

**12- Wait() in Selenium-WebDriver**

1. Implicit Wait :  
   driver.manage.timeouts().implicitlyWait(10,TimeUnit.SECONDS);
2. Explicit Wait:WebDriverWait wait = new WebDriverWait(driver,10);  
   wait.until(ExpectedConditons.elementToBeClickable(By.id/xpath/name("locator"));
3. Using Sleep method of java  
   Thread.sleep(time in milisecond)

**13- Navigation method of Selenium WebDriver Interface**

* **to()** method (its a alternative of **get() method**)  
  driver.navigate().to(Url);  
  This will open the URL that you have inserted as argument
* **back()** – use to navigate one step back from current position in recent history syntax == driver.navigate().back();
* **forward()** – use to navigate one step forward in browser history driver.navigate().forward();
* **refresh()** – This will refresh you current open url driver.navigate().refresh();

**14- Deleting all Cookies before doing any kind of action**

driver.manage().deleteAllCookies();  
This will delete all cookies

**15- Pressing any Keyboard key using Action builder class of Selenium WebDriver**

WebDriver has rewarded us with one class Action to handle all keyboard and Mouse action. While creating a action builder its constructor takes WebDriver as argument. Here I am taking example of pressing Control key  
Actions builder = new Actions(driver);  
builder.keyDown(Keys.CONTROL).click(someElement).click(someOtherElement).keyUp(Keys.CONTROL).build().perform();  
  
When we press multiple keys or action together then we need to bind all in a single command by using build() method and perform() method intend us to perform the action.  
In the same way you can handle other key actions.

**16- Drag and Drop action in Selenium Webdriver**

In this we need to specify both WebElement  like Source and target and for draganddrop Action class has a method with two argument so let see how it normally look like  
WebElement element = driver.findElement(By.name("source"));  
WebElement target = driver.findElement(By.name("target"));  
(new Actions(driver)).dragAndDrop(element, target).perform();

**17- Handling iFrame In Selenium WebDriver**

Iframe is one element on webpage which are one independent pages inside a web page and some time it creates issue to any of the new Selenium Learners. Due to its present user first needs to switch inside this iframe. Only after this, user can perform action on any of the element present on the independent website which is inside iframe.

So here are the functions which you can directly use in your utlity class or selenium base class.

public void selectIFrameUsingIndex(WebDriver driver, int index) {

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | //Selecting index  public void selectIFrameUsingIndex(WebDriver driver, int index) {        driver.switchTo().frame(index);  }    //Using iframe name  public void selectIFrameUsingIndex(WebDriver driver, String name) {        driver.switchTo().frame(name);  }    //Using WebElement  public void selectIFrameUsingIndex(WebDriver driver, WebElement element) {        driver.switchTo().frame(element);  } |