**#Task -2**

## ****What are Locators in Selenium?****

Locators are used to Find the web elements which are present on the webpage ,

Locators are the method of “By” class.

Selenium locators are very power full commands, in other word we can say that locators are the building block of selenium automation script.

## **Types of Locators in Selenium WebDriver ..**

Selenium WebDriver uses 8 types of locators to locate an element on the web page. They are as follows:

* ID
* Name
* ClassName
* TagName
* LinkText
* PartialText
* CSS
* Xpath

****Locate Elements by ID****

### ID locator is the safest and fastest locator to find the location of an element based on the value of “ID” attribute on a web page. It is the most efficient and preferred way to find a web element.Since ID is unique for each element on the web page so it can be easily identified. It is always the first choice for unique.

**Syntax**

**WebElement element = driver.findElement(By.locatorType("Attribute value"));**

### ****Locate Elements by Name****

Name locator is the second safest and fastest locator to locate an element based on the value of “name” attribute on the web page. The name cannot be unique for each element at all times.

If there are multiple elements with the same name on a web page then Selenium will always perform the action on the first matching element.

**Syntax-**

**WebElement element = driver.findElement(By.name("Attribute value"));**

### ****Locate Elements by Class name****

Class Name locator finds an element based on the value of “class” attribute on the web page. The general syntax is like this:

**Syntax-**

**WebElement element = driver.findElement(By.className("Attribute value"));**

****Locate Elements by** TagName Locator**

TagName locator is used to finding an element from the group elements like checkboxes, drop-downs, etc. It locates an element by its tag name.

**Syntax-**

**WebElement element = driver.findElement(By.tagName("Attribute value"));**

****Locate Elements by** LinkText**

This locator finds an element by the link text. If there is one unique element on the web page, you can easily find an element with the link text.But if there are multiple links with the same link text such as repeated header and footer on the web page. In such cases, selenium will select the first matching element with the link.

The general syntax to identify an element by link text is as follows:

**Syntax-**

**WebElement linktext = driver.findElement(By.linkText("Atrribute value"));**

## ****Locate Elements by** Partial LinkText**

## This locator locates an element by a partial match of its link text and then performs actions on it. It works in the same way as the link text. The general syntax is given below:

**Syntax-**

**WebElement partialLinktext = driver.findElement(By.partialLinkText("Attribute value"));**

## ****Locate Elements by** CSS Selector Locator**

This locator finds an element by the CSS selector on the web page. CSS selector makes the execution of test scripts faster as compared to the XPath locator.

It is the best way to find an element on the web page. The CSS stands for Cascading Style Sheets.

**Syntax-**

**WebElement element = driver.findElement(By.cssSelector("Attribute value"));**

****Locate Elements by** XPath**

XPath stands for XML (Extensible Markup Language) Path language. It is a query language that is used to search and select the node in an XML document. All the major web browsers support XPath.

Selenium web driver supports XPath to find the location of any element on a web page using XML path expression. XPath produces a reliable locator but the performance of XPath is slower than CSS selector.

The path of the element at the webpage is selected by XML path syntax. The standard syntax for creating XPath is shown below:

**XPath = //tagname[@attribute = 'value']**

## **Types of XPath in Selenium**

There are two types of XPath in Selenium. They are:

* ****Absolute XPath****
* ****Relative XPath****

## **Absolute Xpath**

It is the easiest way to find the element but if any changes are made in the path of the element then this XPath gets failed. So, This is the main disadvantage of absolute XPath. It begins with a single forward slash (/) which selects the element from the root HTML node. The example of an absolute XPath expression of the element is given below:

**Syntax-**

**Absolute XPath = /html/body/div/div/form/inputWebElement userName = driver.findElement(By.xapth("/html/body/div/div/form/input"));**

## **Relative XPath**

It starts from the double forward slash(//) and selects the element from anywhere on the webpage.

It is the best practice to find an element through a relative XPath and helps us to reduce the chance of “element not found exception”.

With a relative XPath, we can locate an element directly irrespective of its location in the DOM. The example of relative XPath is given below:

**Syntax-**

**Relative XPath = //input[@class='social-media']**

**WebElement username = driver.findElement(By.xpath("//input[@class='social-media']"));**

**Relative X path are of Different Types -**

1. **X path by using contains functions**

e.g. //tagname[Contains(@attribute,”value”)];

1. **X path by using Text functions**

//tagename[Contains(text),’value)];

If we want to find element which is present along with link

E.g. //tage name[text()=’value’];

1. **X path By using attribute name**

We can use different attribute and it’s value and create path

E.g. //tagname [@attribute=’value’];

1. **X path by index**

When we write an X path and if duplicates x path occurs then we can use index and give number to it.

E.g. (Whole xpath expression )[number ];