**Assisted Practice: 1.2 Locating Web Page Elements**

This section will guide you to:

* How to locate elements in Multiple ways using selenium web driver

This lab has mainly eight subsections, namely :

1.2.1 Using ID as a Locator

1.2.2 Using class name as a Locator

1.2.3 Using name as a Locator

1.2.4 Using Link Text as a Locator

1.2.5 Using Xpath as a Locator

1.2.6 Using CSS Selector as a Locator

1.2.7 Using XPath handling complex and dynamic elements

1.2.8 Pushing the code to GitHub repositories

**Step 1.2.1:** Using ID as a Locator

* Open Eclipse
* Finding Web element using Locator **ID**

1. Syntax : id = id of the element
2. Example : driver.findElement(By.id(“Email”));

**Step 1.2.2** Using class name as a Locator

* Finding Web element using Locator **ClassName**
  1. Syntax : class = Class Name of the element
  2. Example : driver.findElement(By.class(“classname”));

**Step 1.2.3** Using Name as a Locator

* Finding Web element using Locator **Name**
  1. Syntax : name = Name of the element
  2. Example : driver.findElement(By.name(“name”));

**Step 1.2.4** Using LinkText as a Locator

* Finding Web element using Locator **Link Text**
  1. Syntax : link = partialLink of the element
  2. Example : driver.findElement(By.partialLinkText(“plink”));

**Step 1.2.5** Using Xpath as a Locator

* Finding Web element using Locator **Xpath**
* Xpath can be created in two ways
  1. **Relative Xpath**
* Syntax : relativeXpath : //\*[@class=’relativexapath’]
* Example : driver.findElement(By.xpath(“//\*[@class=’relativexapath’]”));
  1. **Absolute Xpath**
* Syntax : absoluteXpath : html/body/div[1]/div[1]/div/h4[1]/b
* Example : driver.findElement(By.xpath(“html/body/div[1]/div[1]/div/h4[1]/b”));

**Step 1.2.6** Using Xpath as a **CSS Selector**

* CSS Selector have many formats, namely

1. **Tag and ID**
   * Syntax :”css = tag#id”
   * Example : driver.findElement(By.cssSelector(“input#email”));
2. **Tag and Class**
   * Syntax : “css = tag.class”
   * Example : driver.findElement(By.cssSelector(”input.inputtext”));
3. **Tag and Attribute**
   * Syntax : “css = tag[attribute=value]”
   * Example : driver.findElement(By.cssSelector(“input[name=lastName]”));
4. **Tag, Class and Attribute**
   * Syntax : “tag.class[attribute=value]”
   * Example : driver.findElement(By.cssSelector(“input.inputtext[tabindex=1]”));
5. **Inner text**
   * Syntax : ”css = tag.contains(“innertext”)”
   * Example : driver.findElement(By.cssSelector(font:contains(“Boston”)));

**Step 1.2.7** Using Xpath Handling complex and Dynamic elements

* Dynamic Xpath has many formats, Namely

1. **Contains();**
   * Syntax : “xpath = //\*[contains(text(),’text’)]
   * Example : driver.findElement(By.xpath(”//\*[contains(text(),’sub’]”));
2. **Using OR & AND**
   * Syntax : xpath=//\*[@type=’submit’ or @name=’btnReset’]
   * Example :

driver.findElement (By.xpath(”=//\*[@type=’submit’ or @name=’btnReset’]”));

1. **Start-with function**
   * Syntax : xpath= //label[starts-with(@id,’message’)]
   * Example :

driver.findElement (By.xpath(”//label[starts-with(@id,’message’)]”));

1. **Text();**
   * Syntax : xpath=//td[text()=’UserID’]
   * Example : : driver.findElement (By.xpath(”=//td[text()=’UserID’]”));
2. **Following**
   * Syntax : xpath=//\*[@type=’text’]//following::input
   * Example : driver.findElement(By.xpath(”=//\*[@type=’text’]//following::input”));
3. **Preceding**
   * Syntax : xpath=//\*[@type=’text’]//preceding::input
   * Example : driver.findElement(By.xpath(”//\*[@type=’text’]//preceding::input”));
4. **Following - sibling**
   * Syntax : xpath=//\*[@type=’submit’]//preceding::input
   * Example :

driver.findElement (By.xpath (”//\*[@type=’text’]//following-sibling::input”));

**Step 1.2.8:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**