**Handling Dynamic Element:**

In the application sometimes, we need to identify the element w.r.t some other element such type of elements are called as dependent elements and the other element on which it is dependent is called as independent element.

In order to derive xpath expression for dynamic element, we should follow below steps:

**Steps:**

**1.Note down independent and dependent elements:**

Independent= A1

Dependent=check box

**2.Inspect the independent element:**

Right click on A1 and inspect element

**3.Find the common parent** (the element which is common for both independent and dependent element):

Place the mouse pointer on source code of independent element and move the mouse pointer in upward direction till independent and dependent elements are highlighted.

In this example, the common parent is “tr”

**4.Note down the tree** structure which should contain independent, common parent and dependent element.

|  |
| --- |
| **tr**  (Common Parent)  **<td>A1</td>**  **<td>[2]**  (Dependent Element) |

**5.Derive the xpath for independent element:**

//td[text()='A1']

**6.Derive the xpath for common parent using the above xpath expression:**

//tr[td[text()='A1']]

**7.Derive the xpath for dependent element:**

//tr[td[text()='A1']]/td[2]this will not work bcoz single quote used for mumbai is not supported while checking so type single quotes in firepath as mentioned below

//tr[td[text()='A1']]/td[2]

**How do you login to application without using click() method?**

By pressing the enter key on the login button, this can be done using sendKeys() method -see the program.

**Xpath by Group Index:**

In xpath by ‘Group Index’, we write the **xpath expression** inside the bracket ’()’ & then we write **Index** outside the bracket.

* In such cases first it executes the xpath expression which is present in the bracket then it stores matching element in xpath array.
* Index starts from 1 & the index specified outside the bracket is applied on xpath array.

**Ex:- (//a)[1]** 🡪 Here ‘//a’ is xpath expression & ‘[1]’ is index.

|  |
| --- |
| <html>  <body>  <div>  <a href=" ">Link1</a>  <a href=" ">Link2</a>  </div>  <div>  <a href=" ">Link3</a>  <a href=" ">Link4</a>  </div>  </body>  </html> |

**Example for *Xpath by Group Index* & its matching elements**.

|  |  |
| --- | --- |
| **Xpath** | **Matching Element** |
| //a | All links (**Link1, Link2, Link3, Link4**) |
| //a[1] | All first links (**Link1, Link3**) |
| (//a)[1] | Only first link (**Link1**) |
| (//a)[3] | Only third link (**Link3**) |
| (//a)[last()] | Only once & that is last link (**Link4**) |
| (//a)[last()-1] | Last but one link (**Link3**) |
| (//a)[position() <=3] | Matches with first 3-links (**Link1, Link2, Link3**) |
| (//a)[position() > last()-3) | Matches with last 3-links (**Link4, Link3, Link2**) |
| (//a)[position() mod 2=0 | Matches with all even links (**Link2, Link4**) |
| (//a)[position() mod 2=1 | Matches with all odd links (**Link1, Link3**) |

**How do you verify whether the checkbox or radio button is selected or not?**

Using isSelected() method which returns true if selected else false.

**How do you replace the text present in the textbox with new text?**

First we should remove the text using **clear()** method and then we should enter the new text using **sendKyes()** method

**How do you verify whether ‘Text field/textbox’ is empty or not?**

By using ‘**getAttribute()**’ method. This method is present in ‘**WebElement’** interface. ‘getAttribute()’ method returns ‘**String**’ result. We have to pass “**value**” argument to this method.

**Retrieving the information from element:**

In order to get the content of the webelement such as property-values(attributes), we use **getAttribute()** method. To get text we use **getText()** method. To get tag name of an element we use **getTagName()** method. **Return type** of all the 3 methods is **String.**

Example: see program

**JavascriptExecutor:**

**How do you enter value to textbox and how do you remove value from textbox without using sendKeys() & clear()? Write sample code for this?**

**How do you perform the action even if the element is disabled? How do you enter the value into disabled textbox?**

**🡪** Using ‘JavascriptExecutor’.

**How do you click on element using java-script?**

|  |
| --- |
| String element="document.getElementById('keepLoggedInCheckBox').click()";  JavascriptExecutor j=(JavascriptExecutor)driver;  j.executeScript(element); |

**What are the uses of JavascriptExecutor?**

**🡪** Using JavascriptExecutor we can

1. Scroll the page
2. We can enter the value to a textbox.
3. We can use to remove the value present in the textbox. (clear)
4. We can click on element.
5. We can use to perform the action even if the element is disabled.

**Synchronization:**

Process of matching Selenium speed with the application speed is called as Synchronization.

When we automate the test, there are 2-components such as the Software application that to be tested & the test automation tool (Selenium) that is used for executing test. Both these components have their own speed & the test script should be written such that both these components work with same speed else we get Exception.

So, Synchronization will help both the components to work with same speed.

**Implicit Wait:**

* Implicit wait is used only for synchronization of **findElement** & **findElements** statements.

**Syntax**:- **driver.manage().timeouts().*implicitlyWait(TimeOut, TimeUnit.SECONDS);***

This method takes 2-arguments (Parameters)

1. First argument is ‘**TimeOut’**, **time as integer value** (Ex:- 10), it is the maximum time till it has to wait.
2. Second argument is ‘**TimeUnit’**. Its value can be in days, hours, minutes, seconds, milliseconds, microseconds or nanoseconds.

* ‘Implicit Wait’ can handle only “**NoSuchElement**” exception.
* In ‘Implicit Wait’ the waiting condition is inbuilt & the condition is “***Element is present in the html or not***”.

|  |
| --- |
|  |

* When control comes to ‘findElement’ it will check whether the element is present in html page or not? If it is present then it will return the address of the matching element.
* If the element is not present then it will check whether “Time is over?”
* If time is over then it will throw ‘NoSuchElement’ exception.
* If time is not over then it will wait for 500ms (this duration is called as **Polling period**) & it will continue to search element.
* “Polling period” is defined in ‘FluentWait’ class & it is constant.

|  |
| --- |
| String key="webdriver.chrome.driver";  String value="./Drivers/chromedriver.exe";  System.setProperty(key, value);  WebDriver driver=new ChromeDriver();  driver.manage().window().maximize();  **driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);**  driver.get("http://localhost:8080/login.do");  driver.findElement(By.id("username")).sendKeys("admin");  driver.findElement(By.name("pwd")).sendKeys("manager");  driver.findElement(By.xpath("//div[.='Login ']")).click();  WebElement logout=driver.findElement(By.xpath("//a[.='Logout']"));  String text = logout.getText();  System.out.println(text);  logout.click();  driver.close(); |

**Explicit Wait:**

In order to handle the synchronization of other methods (other than findElement) and to handle other exceptions we use explicit wait.

* In selenium ‘WebDriver’ page is called as explicit wait because we have to specify **waiting condition** explicitly.
* The **waiting conditions** are present in a class called “**ExpectedConditions**”.
* Using ‘Explicit Wait’ we can handle synchronization of findElement & findElements statements also but we have to write the waiting condition for each statement.

**Syntax**:- **WebDriverWait** wait**=new WebDriverWait(*driver, 3*);**

|  |
| --- |
| String key="webdriver.chrome.driver";  String value="./Drivers/chromedriver.exe";  System.setProperty(key, value);  WebDriver driver=new ChromeDriver();  driver.manage().window().maximize();  **WebDriverWait *wait*=new WebDriverWait(driver, 3);**  driver.get("http://localhost:8080/login.do");  driver.findElement(By.id("username")).sendKeys("admin");  driver.findElement(By.name("pwd")).sendKeys("manager");  driver.findElement(By.xpath("//div[.='Login ']")).click();  **wait.until(ExpectedConditions.*titleContains("Enter")*);**  String homePage = driver.getTitle();  System.out.println(homePage);  driver.findElement(By.xpath("//a[.='Logout']")).click();  **wait.until(ExpectedConditions.*titleContains("Login")*);**  String loginPage = driver.getTitle();  System.out.println(loginPage);  driver.close(); |

|  |
| --- |
|  |

* When the condition comes to ‘**WebDriverWait.until()**’ statement it will check the condition. If the condition is **True** then it will go to next step. If the condition is **False** then it will check for ‘TimeOut’.
* If the ‘Timeout’ is over then it will throw ‘**TimeOut’** exception.
* If the ‘Timeout’ is not over it will continue to check condition.

**Q) What are the difference between ‘Implicit & Explicit’ Wait?**

|  |  |
| --- | --- |
| **Implicit Wait** | **Explicit Wait** |
| 1. We do not specify the waiting condition. 2. We can handle synchronization of only ‘**findElement’** & ‘**findElements’** statements. **All at a time**. 3. TimeUnit can be in days, hours, minutes, seconds, milliseconds, microseconds or nanoseconds. 4. After the specified duration we get ‘NoSuchElement’ exception. | 1. We have to specify the waiting condition like ‘WebDriverWait.until()’ 2. We can handle synchronization of any statement but only one at a time. 3. TimeUnit can be only in **seconds**. 4. After the specified duration we get ‘TimeOut’ exception. |

**Q) How do you login & logout from the application without using Implicit wait, Explicit wait & Thread.Sleep() ?**

🡪 We can use Custom Wait (Writing our own code)

|  |
| --- |
| String key="webdriver.chrome.driver";  String value="./Drivers/chromedriver.exe";  System.setProperty(key, value);  WebDriver driver=new ChromeDriver();  driver.manage().window().maximize();  driver.get("https://demo.actitime.com/login.do");  driver.findElement(By.id("username")).sendKeys("admin");  driver.findElement(By.name("pwd")).sendKeys("manager");  driver.findElement(By.xpath("//div[.='Login ']")).click()  while(true)  {  **try**  {  driver.findElement(By.id("logoutLink"))]")).click()  ;  break;  }  catch (Exception e**)**  **{**    **}**  **}**  driver.close(); |

**Q) How do you check whether the page is loaded or not within specified time?**

🡪 By using ‘**pageLoadTimeout()**’

|  |
| --- |
| String key="webdriver.chrome.driver";  String value="./Drivers/chromedriver.exe";  System.setProperty(key, value);  WebDriver driver=new ChromeDriver();  driver.manage().window().maximize();  **driver.manage().timeouts().*pageLoadTimeout***(Duration.ofSeconds(3));  try  {  driver.get("https://demo.actitime.com/login.do");  System.out.println("Page Loaded in 3-Seconds");  }  catch (Exception e)  {  System.out.println("Page is Not Loaded in 3-Seconds");  }  driver.close(); |

**Q) What are the different Synchronization options you have used in Selenium?**

1. Implicit Wait
2. Explicit Wait (WebDriverWait)
3. Custom Wait
4. Thread.Sleep() : It is also called as blind wait
5. pageLoadTimeout()