Day 1: Element Finding Techniques

Syntax:

WebElement elementName = driver.findElement(By.LocatorStrategy("LocatorValue"));

No of Ways to find:

* ID
* Name
* Class Name
* CSS Selector
* Tag Name
* Link Text
* Partial Link Text
* XPATH

**1.Handling Hidden Elements in Web Page**

In some cases Web Elements might be Hidden in the Web page. Due to this issue We will get (ElementNotVisible Exception). This will cause Automation cases will fail.

In order to solve this issue we need to handle this Exception by checking the Size of the Webelement.

List<WebElement> elements= driver.findElements(By.id(“locator”));

Int size=elements.size();

For(int i=0;i<size;i++){

Int x=elements.get(i).getLocation().getx();

If(x!=0){

elements.get(i).click()}

}

Or

We can use javascriptexecutor to run the Command

First we need to find element and then we need to click on the element using Javascript

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("arguments[0].click();", element);

**Finding Shadow Dom elements**

Shadow DOM elements are the elements which are present in side another main DOM, which directly can not be identified using above mentioned locator strategies

For the elements which are present inside the Shadow dome we need to use JavascriptExecutor to get shadowelemts.

Inorder to find this elements we need to find Shadowroot elements using Javascriptexecutors then we need to find elements inside Shadowroot.

Refer week 1 day1 Exapmles in git for the examples

**Youtube link : https://www.youtube.com/watch?v=bpzyjNZ0jaw&list=RDCMUC46vj6mN-6kZm5RYWWqebsg&start\_radio=1&t=3074s**

**XPATH AXES**

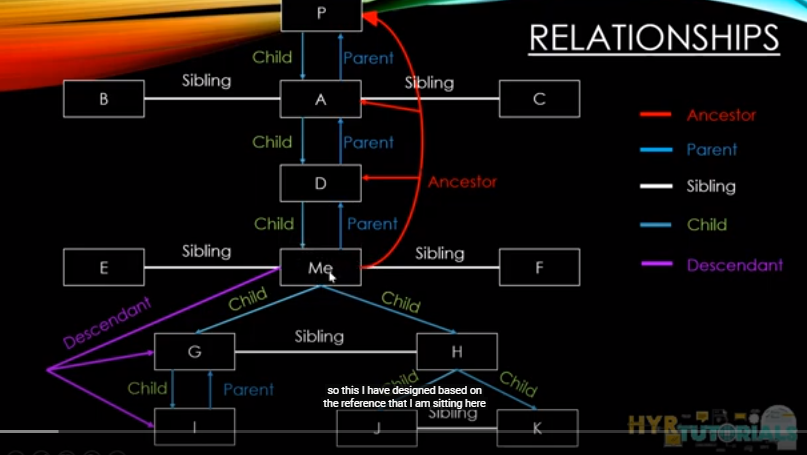
Xpath axes are used when we don’t have any unique id for the elements. We can use these methods to find elements which are not having id or name

Or

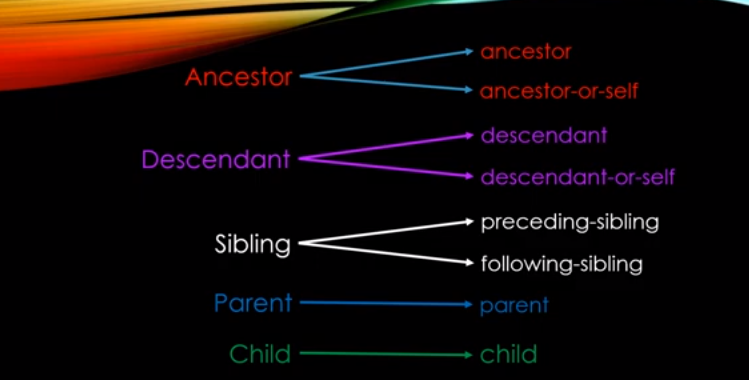
Xpath axes are used to find multiple elements which are not having unique identifiers for elements or all having duplicate elements

Based on classification we have following are Axes



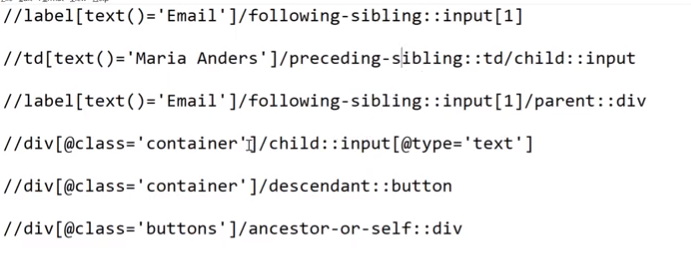


Again based on classification it has divided into following categories they are as follows

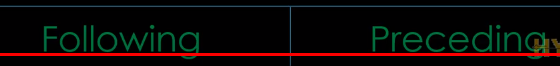


Syntax for each of them are as follows





Apart from these we have 2 more special categories those are

se

The difference between foolowing siblib and following are simple

Following sibling will only give the sibling which is just before you where as following will give all the elements before you. Same applicable for preceding and preceding sibling.

Parent: gives parent of the element

Child : gives child of element

Ancestor: gives all parent elements

Ancestor-or-self : gives all the parent elements including current element

Descendant: gives all the child elements

Descendants-or-self: gives all the child elements including current element

Following-sibling: gives siblings just before you

Preceding-sibling : gives siblings after you.

Example cases are provided in GIT day1 week1 module

**Refer link ; https://www.youtube.com/watch?v=aAWvwGFkySI**

WebElement is a interface which extends 2 more interfaces

SearchContext, TakesScreenshot

1.By is an abstract class which is having all static methods which we used to call and static clases inside it

Forexample: By.cssSelector()

Here in the above CSS selector also a static class which is extending By class and implements serializable.

1. @FindBy(id = "foo") WebElement foo;
2. @FindBy(how = How.ID, using = "foo") WebElement foo;

Both WebDriver and WebElement are interfaces and extending search context Interface

**Finding DropDown values:**

1. Import the package **org.openqa.selenium.support.ui.Select**
2. Instantiate the drop-down box as an object, Select in Selenium WebDriver

**Examples:**

Select drpCountry = new Select(driver.findElement(By.name("country")));

drpCountry.selectByVisibleText("ANTARCTICA");

//Selecting Items in a Multiple SELECT elements

driver.get("your link");

Select fruits = new Select(driver.findElement(By.id("fruits")));

fruits.selectByVisibleText("Banana");

fruits.selectByIndex(1);

|  |  |  |
| --- | --- | --- |
| **Drop-Down Box** | *selectByVisibleText()/* *deselectByVisibleText()* | selects/deselects an option by its displayed text |
| *selectByValue()/* *deselectByValue()* | selects/deselects an option by the value of its "value" attribute |
| *selectByIndex()/* *deselectByIndex()* | selects/deselects an option by its index |
| *isMultiple()* | returns TRUE if the drop-down element allows multiple selection at a time; FALSE if otherwise |
| *deselectAll()* | deselects all previously selected options |

**common ways to handle any type of drop downs.**

1. Click on drop down to make options visible.
2. Get all options of drop down.
3. Iterate through it and match with option to be selected.
4. If found then click on option for selection.

