XPATH

What is XPATH?

- XPATH is defined as XML path, it is a syntax or language for finding any element on the webpage using XML path expression.
- It is a type of locator and it is a path of an element which is present in the HTML tree.

Note:

- XPATH is also called as the XML path language.
- XPATH is a query language for selecting nodes from an XML document.
- XPATH is used to find the location of any element on a web page using HTML DOM structure.
- XPATH contains the path of the element situated at the web page.

Why XPATH?

Dynamic elements

- The element which is changing dynamically during runtime is called as dynamic element.
  Here, changing refers to change in the position or change in property values.
  In order to handle the dynamic elements we use XPTH.

Backward traversing in not allowed in CSS.

- If you want to identify parent of any element, we need to traverse in a backward direction. CSS doesn't support to traverse in a backward direction.
- XPATH is the solution to identify the parent by traversing in a backward direction.

Let's take an another example

```
<html>
    <body>
        UN<input type="text"/>
        PW<input type="text"/>
    </body>
</html>
```

- In the above sample page, we can't use the CSS selector for identifying the element. Coz it identifies the duplicate value coz both the elements have the same name field.

Another Example:

```
<html>
    <body>
        <div>Login</div>
        <div>forgot Password</div>
    </body>
</html>
```

- In the above example, we have two elements, where both the elements doesn't have any attribute value except the text.
- So in order to identify the element using the text of an element. Using CSS selector is not possible to identify the element. We use XPATH for identifying the element using the text of an element.

XPATH is categorized into two types

1. **Absolute XPATH**
2. **Relative XPATH**

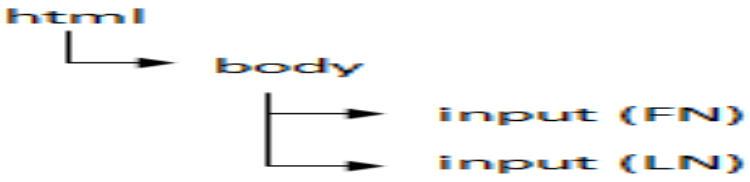
Absolute XPATH

- Specifying complete path of the element form the root till the required element is called as absolute **XPATH**.

Let's take an example of a below HTML.

```
<html>
    <body>
        UN<input type="text"/>
        PW<input type="text"/>
    </body>
</html>
```

HTML tree for the above will look like:



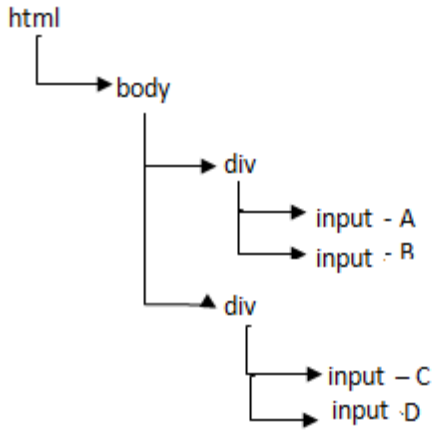
- We write the XPATH expression using '/ '(forward slash). The first forward slash represents beginning of the tree (root).
- After every forward slash we should specify tag of immediate child element. We can also use index which starts from 1.

HTML Code:

```
<html>
    <head>
        <title>Qspiders</title>
    </head>
    <body>
        <center><hr>
            Firstname: <input type="text">
            <br><br><hr>
            Lastname: <input type="text"><hr>
        </center>
    </body>
</html>
```

```
public class Absolute_XPATH1 {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///C:/Users/PriyaPramod/Desktop/IMP/HTML%20Pages/XPath/Demo1.html");
        driver.manage().timeouts().implicitlyWait(100, TimeUnit.SECONDS);

        driver.findElement(By.xpath("/html/body/input[1]")).sendKeys("admin");
        driver.findElement(By.xpath("/html/body/input[2]")).sendKeys("manager");

        driver.close();
    }
}
```

Let's consider the following html tree to deriver XPATH expression

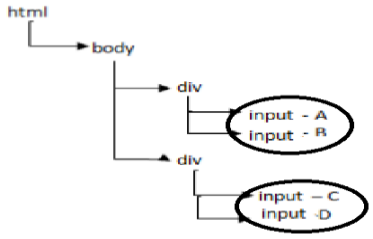


Below is the Absolute XPATH for the above HTML.

Writing the XPATH to identify independent element.

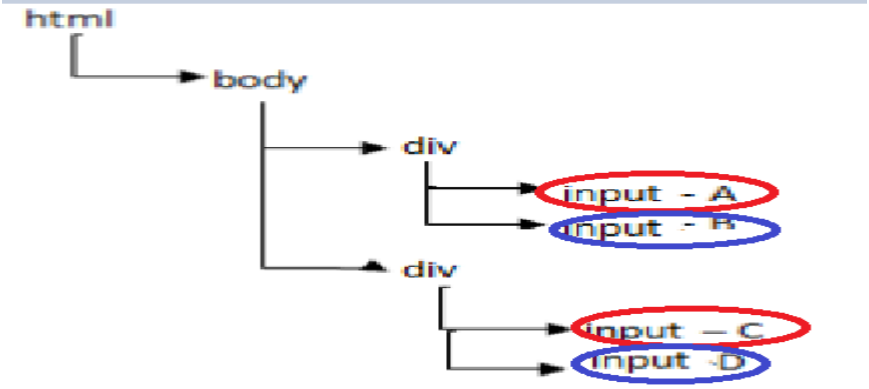
Xpath	Matching Element
/html/body/div[1]/input[1]	A
/html/body/div[1]/input[2]	B
/html/body/div[2]/input[1]	C
/html/body/div[2]/input[2]	D

Xpath to identify A, B and C, D



/html/body/div[1]/input	AB
/html/body/div[2]/input	CD

Xpath to identify A, C and B, D

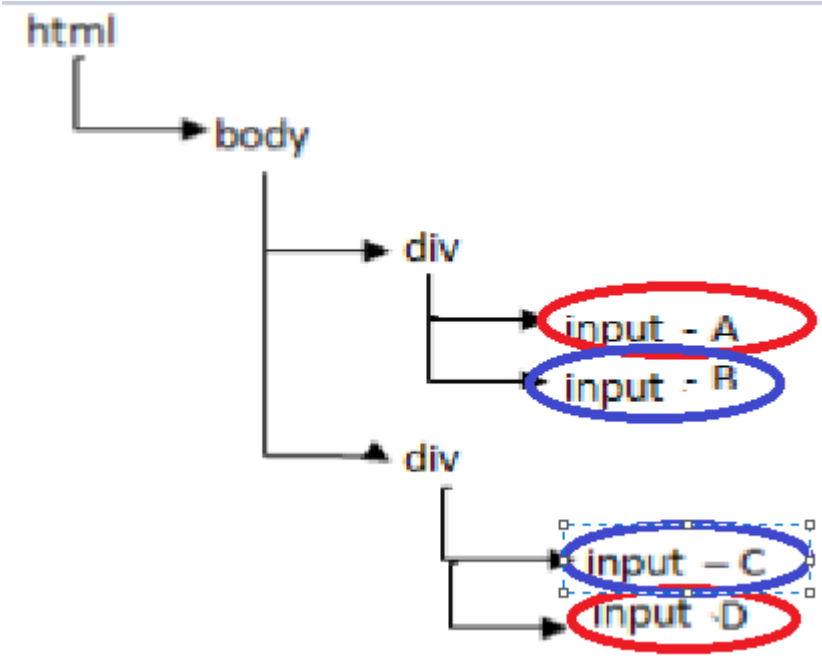


/html/body/div/input[1]	AC
/html/body/div/input[2]	BD

Xpath to identify A, B, C & D

/html/body/div/input	ABCD

Xpath to identify A, D & B, C



- In the above example, consider the scenario where we want to identify A, D & B, C. But both the elements are not in a same position in their respective parent.
- In this case, writing the one XPATH expression to identify both the elements is not possible. Because both the elements are present in the different position.
- So in this case, we derive two XPATH and using the union operator "|", we join two XPATH to make one expression to identify the elements.

/html/body/div[1]/input[1]| /html/body/div[2]/input[2]	AD
/html/body/div[1]/input[2]| /html/body/div[2]/input[1]	BC

- We can join two or more XPATH to make a one expression.

/html/body/div[1]/input[1]| /html/body/div[1]/input[2]| /html/body/div[2]/input[1]	ABC

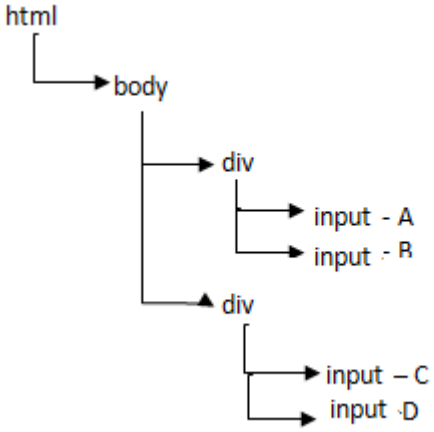
Drawback of absolute XPATH

1. Absolute 'XPATH' is very lengthy. In order to reduce the length of expression we can use relative 'XPATH'.

Relative XPATH

- Specifying the XPATH directly to the web element using web element back end attributes is called as relative XPATH.
- In relative XPATH, we use double forward slash ("//") which represents any child also called as descendent.
- "//" selects node in the document from the current node that match the selection.

Let's take the above HTML to derive the relative XPATH



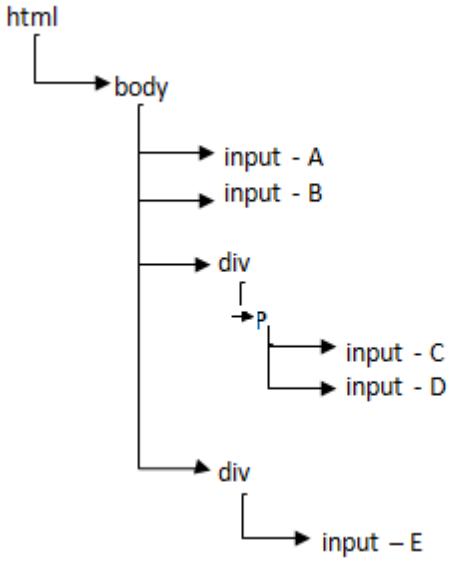
Xpath	Matching element
//div[1]/input[1]	A
//div[1]/input[2]	B
//div[2]/input[1]	C
//div[2]/input[2]	D
//div[1]/input	AB
//div[2]/input	CD
//input[1]	AC
//input[2]	BD
//input	ABCD
//div[1]/input[1]| //div[2]/input[2]	AD
//div[1]/input[2]| //div[2]/input[1]	BC
//div[1]/input[1]| //div[1]/input[2]| //div[2]/input[1]	ABC

Note:

1. **What is the different between single forward slash and double forward slash?**
- Single forward slash represent immediate child whereas double forward slash represents any child (descendent).

2. **What is the difference between '//a' and '//table//a'?**
- '//a' matches with all the links present which are in the entire page. Whereas '//table//a' matches with all the links which are present inside the table.
3. **Derive an 'XPATH' which matches with all the images present on the web page?**
- '//img'
- **Write an 'XPATH' which matches with all the links and all the images present on the web page?**
- '//a|//img'
4. **Important Note:** XPATH matches with hidden elements also.
5. **What is the difference between //input and //div//input?**
- //Input matches with all the inputs present in the entire web page.
- //div//input matches with all the inputs present inside the 'div'.
- //div/input matches with all the immediate child input of 'div'

Below is the HTML code for the above example



//div//input -> CDE
//div/input -> E
//input -> ABCDE
//p/input -> CD

Relative XPATH is categorized into following types

1. XPATH by attribute
2. XPATH by text function
3. XPATH by starts-with function
4. XPATH by normalize-function
5. XPATH by contains function
6. XPATH by traversing
7. Independent and dependent XPATH
8. XPATH by Axes functions
   - ➔ Following
   - ➔ Preceding
   - ➔ Ancestor
   - ➔ Parent
   - ➔ Child
   - ➔ Descendants
   - ➔ Following-Sibling
   - ➔ Preceding-Sibling
9. XPATH by group index

XPATH by attribute

- In XPATH by attribute we use attributes to identify the element uniquely.
- In order to identify the elements using attribute we use a special symbol “@”.
- “@” -> this symbol is used to select the attributes.

Syntax:

```
//tagName[@attribute = attribute's value]
```

// - points to the any node in the webpage

TagName - tag name is nothing but the name which is present after the < (angular bracket)

@: It is used to select to select attribute.

attribute - whatever is present inside < and > bracket except tag name is attribute, any number of attributes can present in html code

attribute's value - it is corresponding value to the attribute.

Example:

```
<input type="text" name="username" id="username" class="textField" placeholder="Username">
```

```
//input[@name='username']
```

Example program to login to actiTIME using XPATH attribute expression.

```
package locator_XPATH;

import org.openqa.selenium.By;

public class Xpath_by_attribute
{
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///C:/Users/PriyaPramod/Desktop/HTML%20Pages/Folder/ID.html");

        WebElement username = driver.findElement(By.xpath("//input[@id='username']"));
        username.sendKeys("admin");
        WebElement password = driver.findElement(By.xpath("//input[@class='textField pwdfield']"));
        password.sendKeys("admin");
        WebElement loginbutton = driver.findElement(By.xpath("//a[@id='loginButton']"));
        loginbutton.click();

        driver.close();
    }
}
```

Note:

- We can use more than one attribute or property name to identify the web element.

Example:

```
<input type="text" name="username" id="username" class="textField" placeholder="Username">

//input[@name='username'][@class='textField']
```

Logical operations in XPATH expression

Anding:

- Consider a scenario, where we need to identify the elements only if multiple attributes are present for the element, then we go for performing the anding operation in the XPATH expression to identify the elements by multiple attribute values.
- Below expression will identify the element only if both the property values are present, if one property is present and another property value is not present, it will through the no such element exception.

Syntax:

```
//TagName[@AttributeName='AttributeValue' and @AttributeName='AttributeValue']
```

Example:

```
<input type="text" data-validate="isEmail" id="email_create" name="email_create" value="">

//input[@id='email_create' and @data-validate='isEmail']
```

```
package locator_XPATH;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_Attribute_Anding {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        WebElement username = driver.findElement(By.xpath("//input[@type='text' and @id='username']"));
        username.sendKeys("admin");
        WebElement password = driver.findElement(By.xpath("//input[@type='password' and @name='pwd']"));
        password.sendKeys("admin");
        WebElement loginbutton = driver.findElement(By.xpath("//a[@id='loginButton']"));
        loginbutton.click();

        driver.close();
    }
}
```

ORING:

- Consider a scenario, where we need to identify the elements if any one of the property value is present in that element. In those cases we develop the XPATH expression using ORING.
- Oring XPATH expression will identify the element, if any one property is present in the DOM.

Syntax:

```
//TagName[@AttributeName='AttributeValue' or @AttributeName='AttributeValue']
```

Example:

```
<input type="text" data-validate="isEmail" id="email_create" name="email_create" value="">
```

```
//input[@id='email_create' or @data-validate='isEmail']
```

```
package locator_XPATH;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_Attribute_Anding {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        WebElement username = driver.findElement(By.xpath("//input[@type='text' or @id='username']"));
        username.sendKeys("admin");
        WebElement password = driver.findElement(By.xpath("//input[@type='password' or @name='pwd']"));
        password.sendKeys("admin");
        WebElement loginbutton = driver.findElement(By.xpath("//a[@id='loginButton']"));
        loginbutton.click();

        driver.close();
    }
}
```

Not in XPATH

- Not function in XPATH, identifies the element if the specified attributes are not available.

Syntax:

```
//TagName [not(source='value')]
```

Here, Source is either, text of an element or attribute.

Example:

```
//a[not(text()='xpath')]
```

- The above expression identifies all the element, which doesn't have a text value "xpath"

```
//a[not(@id)]
```

- The above expression identifies all the element, which doesn't have a "@id".

XPATH by text function

- In order to identify the element by visible text we use a function "text ()" in XPATH to locate the element.

Syntax:

```
//TagName[text()='Visible text/text of an element']
```

Example 1:

```
package locator_XPATH;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_text {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        WebElement checkBox = driver.findElement(By.xpath("//label[text()='Keep me logged in']"));
        checkBox.click();

        driver.close();
    }
}
```

Example 2:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_text_Example {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://accounts.google.com/");

        WebElement createAccount = driver.findElement(By.xpath("//span[text()='Create account']"));
        createAccount.click();

        driver.close();
    }
}
```

- Text function is used to navigate to entire HTML document and check for the expected value in UI, this return true if complete string matches with UI or else it returns no such element exception.
- Text function is used to identify any web element using the visible text in the UI.
- Text function is exact string matching function, which can't identify object part of a string.

Drawback back of Text function:

- Text function cannot identify the element, if that text contains the spaces before or after the text.
- It can't identify web element using the part of a string. We need to give the complete string to identify the element.
- Text function can't be used with back end attributes.

Note: In order to overcome from the drawback of text function. We use text function with another function.

Below are the two function, with which we use text function to identify the element using the visible text.

XPATH by normalize-space

The normalize-space function is used to normalize a string i.e. to remove any leading or trailing spaces from the string passed as parameter to the XPATH function.

Syntax:

```
//htmlTag[normalize-space(source)='value']

here,
    source may be a text() function or @attributeName
```

Example using text function:

```
//label[normalize-space(text())='Keep me logged in']
```

Example using the attribute:

```
//input[normalize-space(@name)='username']
```

Will look into few examples to see how the normalize space function works.

Example:

```
<html>
    <head>
        <title>Qspiders</title>
    </head>
    <body>
        <form>
            <input type="text" id="username">
            <input type="text" id="password">
            <br><br>
            <button type="button" name="pwd" value="Login">   Login   </button>
            <br><br>
            <a href="http://qspiders.com/">  Forgot Password?  </a>
        </form>
    </body>
</html>
```

- In the above HTML source, if we try to identify the login and forgot password element using the text function. We won't be able to identify the element.
- Coz, both the element has a spaces before and after the text.
- So to identify the element, even if the string contains the trailing spaces, we use normalize-space function.

Example program for text function in normalize-space:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_normalize_space_example1 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///C:/Users/PriyaPramod/Desktop/HTML%20Pages/XPath/ID.html");

        WebElement loginButton = driver.findElement(By.xpath("//button[normalize-space(text())='Login']"));
        loginButton.click();

        driver.close();
    }
}
```

Example program for attribute in normalize-space:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_normalize_space_example2 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///C:/Users/PriyaPramod/Desktop/HTML%20Pages/XPath/ID.html");

        WebElement userName = driver.findElement(By.xpath("//input[normalize-space(@id)='username']"));
        userName.sendKeys("admin");

        driver.close();
    }
}
```

XPATH by Contains

By using 'contains' function in XPath, we can extract all the elements which matches a particular text value and attribute value.

Syntax:

```
//HtmlTag[contains(source,'value')]

Here,
    as a source, we can use text() function or @Attribute
```

Example for using the text () function in Contains

```
//label[contains(text(),'Keep me logged in')]
```

Example for using the @Attribute in contains

```
//input[contains(@id,'username')]
```

Example:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_by_contains {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        WebElement userName = driver.findElement(By.xpath("//input[contains(@id,'username')]"));
        userName.sendKeys("admin");
        WebElement password = driver.findElement(By.xpath("//input[contains(@name,'pwd')]"));
        password.sendKeys("manager");
        WebElement loginButton = driver.findElement(By.xpath("//div[contains(text(),'Login')]"));
        loginButton.click();

        driver.close();
    }
}
```

Note:

Situations where contains function should be used:

- When the values of the attributes are dynamic i.e. changing.

- When we would like to create a list of web elements contains same partial attribute value.
- Contains function is a sub string matching function, which can verify web element using part of the string.
- Contains function automatically ignore white spaces before and after the string.

XPATH by Starts-with

Starts-with function is used when we know about the initial partial attribute value or initial partial text associated with the web element.

User can also use this function to locate we elements those are consists of both the static (initial) and dynamic (trailing) values.

Syntax:

```
//htmlTag[starts-with(source, 'value')]

Here,
     In source, we can use text() function or @attributes
```

Example to use text function in starts-with:

```
//div[starts-with(text(),'Log')]
```

Example to use attribute in starts-with:

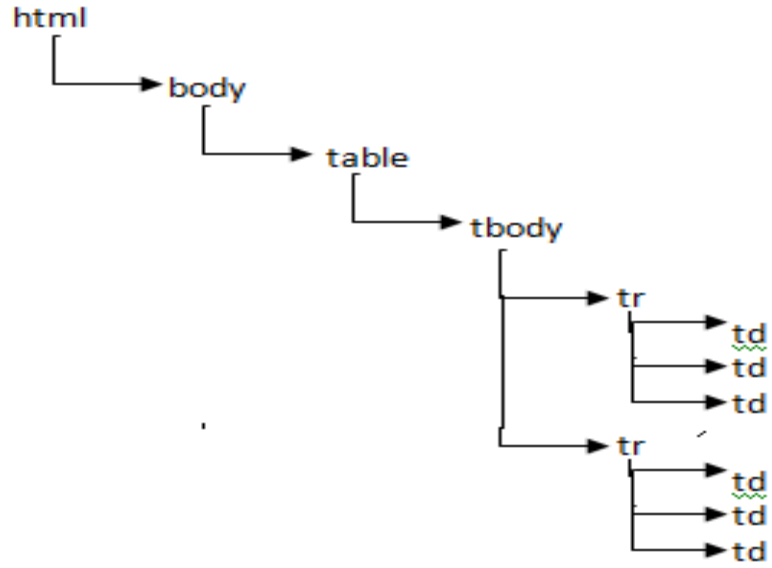
```
//input[starts-with(@id,'user')]
```

```
public class Xpath_Starts_with {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        WebElement userName = driver.findElement(By.xpath("//input[starts-with(@id,'username')]"));
        userName.sendKeys("admin");
        WebElement password = driver.findElement(By.xpath("//input[starts-with(@name,'pwd')]"));
        password.sendKeys("manager");
        WebElement loginButton = driver.findElement(By.xpath("//div[starts-with(text(),'Login')]"));
        loginButton.click();

        driver.close();
    }
}
```

XPATH by traversing

We can write 'XPATH' expression which navigates from one element to another element which is called as traversing. In 'XPATH' there are 2 types of traversing.

- Forward Traversing
- Backward Traversing.



Forward Traversing

- Navigating from parent element to its any of the child element is called as forward traversing.

Example:

1. Navigating from table node to java cell

```
//table/tbody/tr[1]/td[2]
```

2. Navigating from table to Unix cell

```
//table/tbody/tr[2]/td[2]
```

Backward Traversing

- Navigating from child element to any of its parent element is called as backward traversing.

Example:

There are two ways to travel in a backward direction.

- **Enclosing the child expression with in []**
- **Using the "/.."**

Navigating from java cell to table node

```
//table[tbody[tr[td[text()='Java']]]]

//td[text()='Java']/../../..
```

Independent and dependent Xpath

- If the element is completely dynamic or if the element is duplicate, then we can identify that element using some other element by applying a technique called independent, dependent XPATH.
- The element which is identified with respect to some other element is called as dependent element.

Let's take the below example:

1	Java	200
2	Unix	300

'XPATH' to identify cost of Java:

```
//td[text()='300']
```

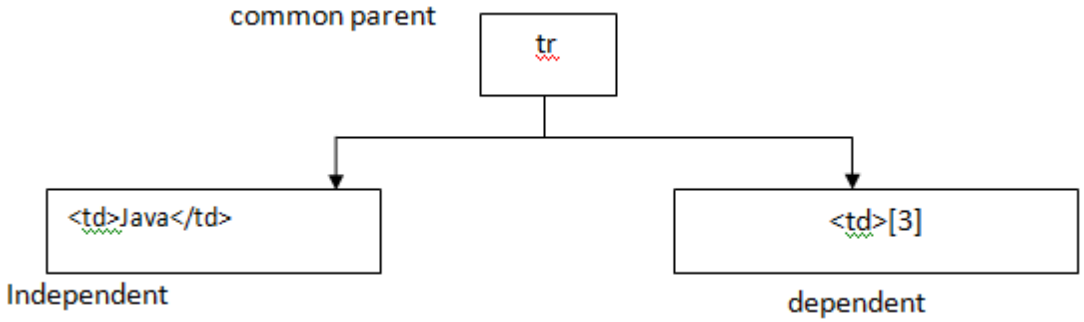
- In the above 'XPATH' expression we are identifying the cost directly.
- If the cost of the Java book changes then this 'XPATH' will not identify the element.

```
//td[text()='Java']/../td[3]
```

- In the 2nd 'XPATH' expression we are identifying the cost using the name of the subject.
- In this example Java is called as independent element and cost is called as dependent element. This will identify the cost even if it completely changes.

Steps to derive independent and dependent XPATH

1. First identify dependent and independent XPATH
2. Inspect the independent element and note the source code.
3. Find the common parent.
   To find the common parent: Place the mouse pointer on source code of independent element and move the mouse pointer on upward direction step by step till it highlights both independent and dependent element. It will be the common parent. Add it to HTML tree.
4. Navigates from common parent to dependent element using arrow key and add it to HTML tree as shown below.

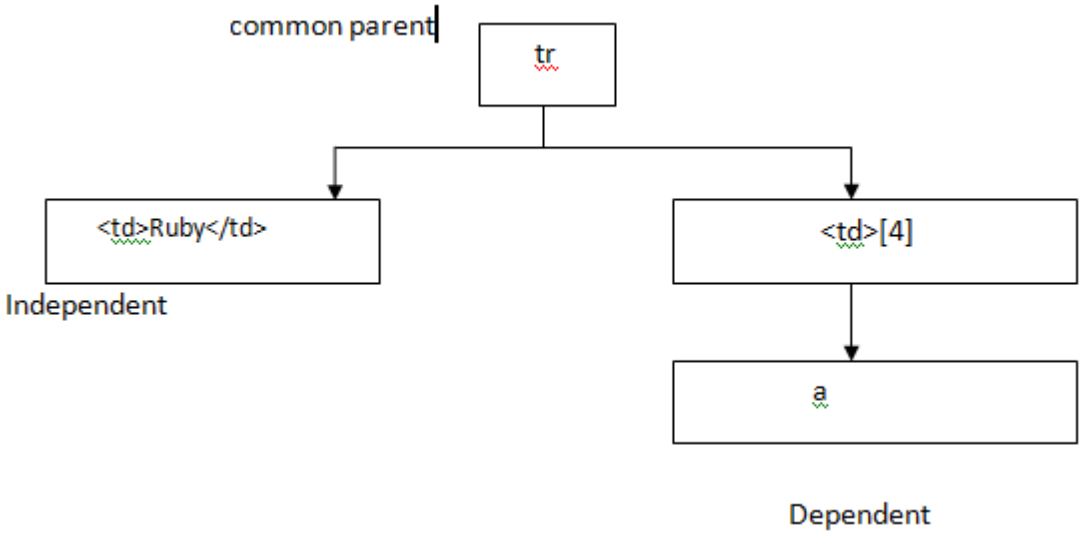


5. Derive the 'XPATH' which navigates from independent element to common parent and then to dependent element.
   - ➔ First derive the xpath expression for independent element
   - ➔ Second derive the xpath expression from independent element to common parent
   - ➔ Lastly derive the xpath expression from common parent to dependent element.

Example:

```
//td[text()='Java']/../td[3]
```

Derive an 'XPATH' which identifies download link of ruby which is present in Selenium download page?




```
Xpath: //td[text()='Ruby']/../td[4]/a
```

- The above 'XPATH' expression identifies the download link only if it is present in 4th column. If the column keeps changing we can use the below 'XPATH'.

```
Xpath: /td[text()='Ruby']/..//a[text()='Download']
```

Wild card character with XPATH

"*"

- Is the one of most used wild card character with XPATH in selenium webdriver, we can use it instead of tag name and attribute

//*

- Matches all the elements present in the html (including html)

//div/*

- Matches all the immediate element(s) inside the **div** tag

//input[@*]

- Matches all the element(s) with **input** tag and have at least one attribute, attribute value may or may not present

//*[@*]

- Matches all the element(s) which have at least one attribute.

How to write XPath if I have 'apostrophe' in my XPath element?

Let's take the below example,

/ Women's fashion /

- In the above example, the visible text contains the special character called apostrophe.
- In this case the only reliable way of using XPath in Selenium WebDriver for text with apostrophes (single quotes) is to use double quotes for the expression of the XPath.

```
//*[@attribute/text()=\"text's\"]
```

Example:

```
//span[text()=\"Women's fashion\"]
```

Example:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Xpath_By_apostrophe {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.craftsvilla.com/cvfeeds/craftsvilla-brands");

        WebElement womensFashion = driver.findElement(By.xpath("//span[text()=\"Women's fashion\"]"));
        womensFashion.click();

        driver.close();
    }
}
```