XPATH Tutorial:

XPath, the XML path language, is a query language for selecting nodes from an XML

document. All the major browsers support XPath as HTML pages are represented as XHTML

documents in DOM. The XPath language is based on a tree representation of the XML document and provides the

ability to navigate around the tree, selecting nodes using a variety of criteria.

Selenium WebDriver supports XPath for locating elements using XPath expressions or queries.

Locating elements with XPath works very well with a lot of flexibility. However, this is the least

preferable locator strategy due its slow performance.

XPath we can search elements backward or forward in the DOM hierarchy, this means that with XPath we can locate a parent element using a child element.

Types of XPATH:

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

Absolute XPATH:

If any Xpath starts with roots element or tag name <html> that indicates Absolute Xpath

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
namespace SeleniumAutomation
{
    class Program
    {
        public static IWebDriver oBrowser = null;
        static void Main(string[] args)
        {
            LaunchBrowser();
            Navigate();
            XpathUsingAbsoluteApproach();

        }

        static void LaunchBrowser()
        {
            oBrowser = new ChromeDriver();
            oBrowser.Manage().Window.Maximize();
            Thread.Sleep(4000);
        }
```

```
        static void Navigate()
        {
            oBrowser.Navigate().GoToUrl("file:///E:/EXAMPLE/Sample.html");
            Thread.Sleep(4000);
        }

        static void XpathUsingAbsoluteApproach()
        {

              oBrowser.FindElement(By.XPath("/html/body/div/form/input")).SendKeys("DemoU
              ser1");
        }
    }
}
```

Relative XPATH:

With relative path, we can locate an element directly irrespective of its location in the DOM.

Case 1: Identify Element based on tag name alone

Syntax:

//<tagname>

Below Method Enters value in User Name text field

```
static void RelativeXPathUsingTagNameAlone()
{
    oBrowser.FindElement(By.XPath("//input")).SendKeys("DemoUser1");
}
```

Case 2: Identify Element based on index

Syntax:

//<tagname>[index]

The below Method enter value in Password text field

```
static void RelativeXPathUsingTagNameWithIndex()
{
    oBrowser.FindElement(By.XPath("//input[2]")).SendKeys("DemoUser1");
}
```

Case 3: Identify the Element based on Tag Name with Attribute Name and Value combination

Syntax:

//<tagname>[@attributename='attribute value']

The below function click on Submit button

```
static void RelativeXPathUsingTagNameWithAttributeNameAndValue()
{
    oBrowser.FindElement(By.XPath("//input[@value='Submit']")).Click();
}
```

Case 4: Identify the Element based on Irrespective of Tag Name with Attribute Name and Value combination
Syntax:

//*[@attributename='attribute value']

The below Method click on Submit button

```
static void RelativeXPathUsingIrrespectiveOfTagNameWithAttributeNameAndValue()
{
    oBrowser.FindElement(By.XPath("//*[@value='Submit']")).Click();
}
```

Case 5: Identify the Element based on Tag Name with Multiple Attribute Names and Values combination

Syntax:

//<tagname>[@attributename1='attribute value1'] [@attributename2='attribute value2']

The below Method click on Submit button

```
static void RelativeXPathUsingTagNameWithMultipleAttributeNamesAndValues()
{
    oBrowser.FindElement
        (By.XPath("//input[@value='Submit'][@name='submit1btn1']")).Click();
}
```

Case 6: Identify the Element based on Tag Name with Multiple Attribute Names and Values combination with and Operator

Syntax:

//<tagname>[@attributename1='attribute value1' and @attributename2='attribute value2']

The below Method click on Submit button

```
static void RelativeXPathUsingTagNameWithMultipleAttributeNameAndValueWithAndOperator()
{
    oBrowser.FindElement(By.XPath("//input[@value='Submit' and
@name='submit1btn1']")).Click();
}
```

Case 7: Identify the Element based on Tag Name with Multiple Attribute Names and Values combination with or Operator

Syntax:

//<tagname>[@attributename1='attribute value1' or @attributename2='attribute value2']

The below Method click on Submit button

```
static void RelativeXPathUsingTagNameWithMultipleAttributeNameAndValueWithOROperator()
{
    oBrowser.FindElement(By.XPath("//input[@value='Submit' or
@name='submit1btn1']")).Click();
}
```

Case 8: Identify the Element based on Tag Name with Attribute Value Alone
Syntax:

//<tagname>[@* =’attribute value’]

The below Method click on Submit button

```
static void RelativeXPathUsingTagNameWithAttributeValueAlone()
{
    oBrowser.FindElement(By.XPath("//input[@*='Submit']")).Click();
}
```

Case 9: Identify the Elements based on Tag Name and Attribute Name alone
Syntax:

//<tagname>[@attributename]

The below Method click on SeleniumHQ link

```
static void RelativeXPathWithTagNameAndAttributeNameAlone()
{
    try
      {
          IList<IWebElement> oLinks = oBrowser.FindElements(By.XPath("//a[@href]"));
          Console.WriteLine("# of Links in the Application:" + oLinks.Count);
          for (int i = 0; i < oLinks.Count; i++)
          {
               if (oLinks[i].Text.StartsWith("Selenium"))
               {
                    oLinks[i].Click();
                    Thread.Sleep(4000);
                    oBrowser.Navigate().Back();
               }
          }
       }catch(Exception e)
        {
            Console.WriteLine(e.ToString());
        }
  }
```

Case 10: Identify the Elements based on Partial matching of Attribute value
Syntax:

//<tagname>[starts-with(@attributename,’partial attribute value’)]

//<tagname>[ends-with(@attributename,’partial attribute value’)]

//<tagname>[contains (@attributename,’partial attribute value’)]

The below Method click on Submit button

```
static void RelativeXPathUsingPartialMatchingOfAttributeValue()
{
    oBrowser.FindElement(By.XPath("//input[contains(@id,'submit1')]")).Click();
}
```

Case 11: Identify the Element based on the Text Content
Syntax:

//<tagname>[text() = 'text Content']

The below Method click on Gmail Link

```
static void RelativeXpathUsingExactMatchingOfTextContent()
{
    oBrowser.FindElement(By.XPath("//a[text()='Gmail']")).Click();
}
```

Case 12: Identify the Element based on the Partial Matching Text Content

Syntax:

//<tagname>[starts-with(text(),'partial attribute value')]

//<tagname>[ends-with(text(),,'partial attribute value')]

//<tagname>[contains (text(),,'partial attribute value')]

The below Method click on SeleniumHQ Link

```
static void RelativeXPathUsingPartialMatchingOfTextContent()
{
    oBrowser.FindElement(By.XPath("//a[starts-with(text(),'Selenium')]")).Click();
}
```

CSS Tutorial:

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language such as HTML or XML.

Selenium WebDriver uses principles of CSS selectors to locate elements in DOM. This is a much faster and more reliable way to locate the elements when compared with XPaths.

Types of CSS:

1. **Absolute CSS**
2. **Relative CSS**

Absolute CSS:

If any CSS Path starts with roots element or tag name <html> that indicates Absolute CSS

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using System.Collections;

namespace SeleniumAutomation
{
    class Program
    {
        public static IWebDriver oBrowser = null;
        static void Main(string[] args)
        {
            LaunchBrowser();
            Navigate();
            AbsoluteCSSPath();

        }

        static void LaunchBrowser()
        {
            oBrowser = new ChromeDriver();
            oBrowser.Manage().Window.Maximize();
            Thread.Sleep(4000);
        }

        static void Navigate()
        {
            oBrowser.Navigate().GoToUrl("file:///E:/EXAMPLE/Sample.html");
            Thread.Sleep(4000);
        }
```

```
        static void AbsoluteCSSPath()
        {
            oBrowser.FindElement(By.CssSelector("html body div form input"))
            .SendKeys("DemoUser1");
        }
    }
}
```

Relative CSS:

With relative path, we can locate an element directly irrespective of its location in the DOM.

Case 1: Identify Element based on tag name alone

Syntax:

<tagname>

The below Method enters the value in User Name text field

```
static void RelativeCSSSelectorUisngTagNameAlone()
{
    oBrowser.FindElement(By.CssSelector("input")).SendKeys("DemoUser1");
}
```

Case 2: Identify the Element based on Tag Name with ID attribute value
Syntax:

<tagname>#<id attribute value>

The below Method enters the value in Password text field

```
static void RelativeCSSSelectorUisngTagNameWithIDAttributeValue()
{
    oBrowser.FindElement(By.CssSelector("input#pwd1pass1word1"))
    .SendKeys("DemoUser1");
}
```

Case 3: Identify the Element based on ID attribute value Alone

Syntax:

#<id attribute value>

The below Method enters the value in Password text field

```
static void RelativeCSSSelectorUisngIDAttributeValueAlone()
{
    oBrowser.FindElement(By.CssSelector("#pwd1pass1word1"))
    .SendKeys("DemoUser1");
}
```

Case 4: Identify the Element based on Tag Name with Class attribute value
Syntax:

<tagname>#<class attribute value>

The below Method enters the value in Password text field

```
static void RelativeCSSSelectorUisngTagNameWithClassAtributeValue()
{
    oBrowser.FindElement(By.CssSelector("input.pass1word1"))
    .SendKeys("DemoUser1");
}
```

Case 5: Identify the Element based on Class attribute value Alone

Syntax:

#<class attribute value>

The below Method enters the value in Password text field

```
static void RelativeCSSSelectorUisngClassAttributeValueAlone()
{
    oBrowser.FindElement(By.CssSelector(".pass1word1")).SendKeys("DemoUser1");
}
```

Case 6: Identify the Element based on Tag Name with Other Attribute Name and Value combination

Syntax:

<tagname>[attributename='attribute value']

The below Method click on Submit button

```
static void RelativeCSSSelectorUsingOtherAttributeNameAndValue()
{
    oBrowser.FindElement(By.CssSelector("input[value='Submit']")).Click();
}
```

Case 7: Identify the Element based on Tag Name with Multiple Other Attribute Names and Values combination

Syntax:

<tagname>[attributename1='attribute value1'] [attributename2='attribute value2']

The below Method click on Submit button

```
static void RelativeCSSSelectorUsingMultipleOtherAttributeNamesAndValues()
{
    oBrowser.FindElement
    (By.CssSelector("input[value='Submit'][name='submit1btn1']")).Click();
}
```

Case 8: Identify the Elements based on Tag Name and Attribute Name alone
Syntax:

//<tagname>[attributename]

The below Method click on Eclipse link

```
static void RelativeCSSSelectorUsingTagNameWithAttributeNameAlone()
{
    try
       {
            IList<IWebElement> oLinks = oBrowser.FindElements(By.CssSelector("a[href]"));
            Console.WriteLine("# of Links in the Application:" + oLinks.Count);
            for (int i = 0; i < oLinks.Count; i++)
            {
                    if (oLinks[i].Text.StartsWith("Selenium"))
                    {
                        oLinks[i].Click();
                        Thread.Sleep(4000);
                        oBrowser.Navigate().Back();
                    }
             }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }

  }
```

Case 9: Identify the Elements based on Partial Matching of Attribute Value

Syntax:

the Below Syntax for contains

//<tagname>[attributename *= 'Partial Attribute value]

the Below Syntax for starts-with

//<tagname>[attributename ^= 'Partial Attribute value]

the Below Syntax for ends-with

//<tagname>[attributename $= 'Partial Attribute value]

The below Method click on Submit button

```
static void RelativeCSSSelectorUsingPartialAttributeValue()
{
    oBrowser.FindElement(By.CssSelector("input[id *= 'submit1']")).Click();
}
```

Case 10: Identify the Child Element based on the Parent Element Reference

Syntax:
Parent Element CSS Path>Child Element CSS Path
CSS selectors provide way to locate child elements from parent elements.

The Below Method click on Windows Check Box

```
static void RelativeCSSSelectorUsingParentReferenceIdentifyChildElement()
{
    oBrowser.FindElement(By.CssSelector("div#d2>form#frm2 input")).Click();
}
```

Case 11: Identify the Element based on nth Child Concept

The Below Method enters text in 5th FirstName text field

```
static void RelativeCSSSelectorUsingNthChildConcept()
{
    oBrowser.FindElement(By.CssSelector("div#d3>form#frm3 :nth-child(5)"))
    .SendKeys("DemoUser1");
}
```

Case 12: Identify the Sibling Elements

With CSS selector, we can locate sibling elements using the + operator.

The Below Method enters text in 3rd FirstName text field

```
static void RelativeCSSSelectorUsingSiblingConcept()
{
    oBrowser.FindElement(By.CssSelector("div#d3>form#frm3 input + input + input"))
    .SendKeys("DemoUser1");
}
```