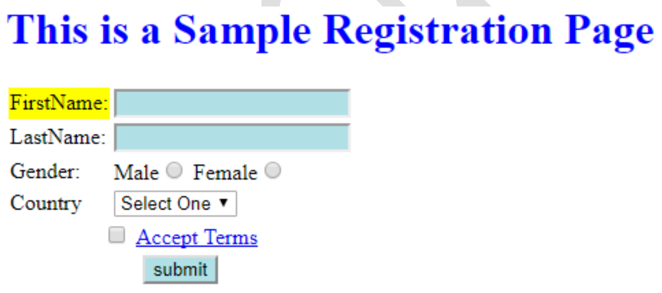
HTML Basics

- The best site to learn html is https://www.w3schools.com/html/
- In every html page there will be head and body
- Whatever we write in body will be displayed to the user
- We write html code using tags
- We specify open tag like this <tagname> and close tags like </tagname>
- We can also close the tag in open tag by using <tagname/>
- There is a meaning for every tag name
- Body tag creates body for html page
- Input tag used to create user input related elements like textbox, checkbox, radio…etc
- Every tag will create some UI in html page
- We also specify attributes for the tags
- These attributes will be specified in open tag of every element
- To create text box we use <input type="text" /> here input is tag name, type attribute name
- Some attributes create functionality for element and some are identification attributes given by by developer to work with those elements in future
- For example login functionality requires user name and password to be entered. When user enter the username and password, the developer should be able to read the data from username and password fields. For that developer will be assigning some identification attributes to those elements.
- Create a sample Html page for registration form
- Copy below html code to a text file and save that text file as .html

```
<!DOCTYPE html>
<html>
<head>
<title>SamplePage</title>

    <style>
    h1   {color: blue;}
    input {background-color: powderblue;}
    .demo {background-color: yellow;}
    #lname{color: red;}
    </style>

</head>
<body>

<h1>This is a Sample Registration Page</h1>

<table>

    <tr>
        <td class="demo">FirstName:</td>
        <td><input type="text" id="fname" /></td>
    </tr>

    <tr>
        <td>LastName:</td>
        <td><input type="text" id="lname" /></td>
    </tr>
    <tr>
        <td>Gender:</td>
        <td>Male<input type="radio" name="g"> Female<input type="radio" name="g"></td>
    </tr>

    <tr>
        <td>Country</td>
        <td><select id="country">
        <option>Select One</option>
        <option>India</option>
        <option>USA</option>
        <option>China</option>
        </select></td>
    </tr>

    <tr>
        <td colspan=2 align="center"><input type="checkbox"> <a href="google.com">Accept T
erms</a></td>
    </tr>

    <tr>
        <td colspan=2 align="center"><input type="button" value="submit" onclick="myFuncti
on()"/></td>
    </tr>
</table>

<script>
function myFunction() {
    var firstname = document.getElementById("fname").value;
    alert(firstname);
}
</script>

</body>
</html>
```

- Open Html Page
- You find below UI



- We can create
    - text boxes using <input type=”text” />
    - Radio using <input type=”radio” />
    - Checkbox using <input type=”checkbox” />
    - Links using <a href=”url”>linkname</a>
    - Buttons using <input type=”button” value=”Submit”/> or <button>Submit</button>
    - display text : for headers h1, span, p, label, li
- If we keep these controls in html table we can segregate them as rows and columns

- A webpage is combination of html + css + javascript
- Developers use css to style html pages
- For styling a single element they will use style attribute <h1 style="color:blue;">This is a header</h1>
- For styling elements that are having same tag then they use style tag and access the element tags in that for styling
    - In above html code I have used style tag and accessing h1 and input tags for giving styling
- For styling different elements then they create a css class under style tag
    - In above html I have used style tag and specified some styling using .demo{ }
    - The css class is used to store some styling
    - Where ever we want styling we specify the class name in open tag of the element
    - `<td class="demo">FirstName:</td>` is displayed in yellow color in presentation

Attributes and use of them for Automation purpose

- There are identification attributes and functional attributes
- Using identification attributes developer/Automation tester access the element to perform operations or to get data from that element
- Functional Attributes provides the functionality for elements
- Identification attributes used to provide identification
- id : It's an identification attribute
    - An element with meaningful id attribute value can be easily find by webdriver.
- name: it's an identification attribute
    - For radio buttons name will return multiple elements because we always have radio group. The rule for radio button in html is the name attribute must be same for all radio buttons in its group. If the name attribute is same then the selection and deselection will happen.
    - Gender: Male: <input type="radio" value="m" name="gen"/> Female: <input type="radio" value="f" name="gen" />
    - If there is id then find radio with id
    - If you want to use name to find radio button then use value attribute also
    - If value attribute is not there then use findelements method and go for index
- value: this attribute will work as identification for radio, buttons and list options
- For textbox elements value attribute is the text what we enter
- class: this attribute is used to specify css style class names for elements

Which attribute to be used for which control

- textbox: id, name, class
- button: id, name, value, class
- radio: id, name, value, class
- checkbox: id, name, class
- listbox: id, name, class
- listOption: text, value

- links: id, name, text, class
- table: id, name, class
- display text elements: id, name, text, class

Webdriver is providing id, name and classname attributes as locators in findelement method. To use any other attribute we have to use xpath or cssSelector locators

WebDriver Sample Program using Locators

```
public static void main(String[] args) {

	//specify browser driver
	System.setProperty("webdriver.gecko.driver", "F:\\geckodriver.exe");

	//open new firefox window
	WebDriver driver = new FirefoxDriver();

	//Find User Name Textbox and enter admin text
	driver.findElement(By.id("txtUsername")).sendKeys("admin");

	//Find Password Textbox and enter admin text
	driver.findElement(By.id("txtPassword")).sendKeys("admin");

	//Find login button and click on it
	driver.findElement(By.id("btnLogin")).click();

}
```

- In above program we have used selenium provided classes and method to open browser, Finding elements and handling them
- FindElement will find the element in application using locators
- Webdriver is providing id, name and classname attributes as locators in findelement method
- If we want to use any other attribute for finding element then we have to use xpath or cssSelector locator
- There are 8 locators in webdriver
    - Id
    - Name
    - className
    - tagName
    - linkText
    - partialLinkText
    - Xpath
    - Css

Syntax for using each locator

- id: the id attribute of an element
    - <input type="text" id="uname" />
    - driver.findElement(By.id("uname")).sendKeys("abcd");

- name: the name attribute of an element
    - <input type="text" name="uname" />
    - driver.findElement(By.name("uname")).sendKeys("abcd");
- className: the css class attribute of an element
    - <input type="text" class="demo" />
    - driver.findElement(By.className("demo")).sendKeys("abcd");
- tagName: the tag name of the element
    - <input type="text" class="demo" />
    - driver.findElement(By.tagName("input").className("demo")).sendKeys("abcd");
    - className and tagName may not be unique
    - We can get list of elements with class name or tagname locators
- linkText: the text between open tag and closed tag of a link
    - <a href="http://seleniumhq.org"> Selenium </a>
    - driver.findElement(By.linkText("Selenium")).click();
- partialLinkText: the partial text between open tag and closed tag of a link
    - <a href="http://seleniumhq.org"> Selenium </a>
    - driver.findElement(By.partialLinkText("ium")).click();
- Xpath: XPath can be used to navigate through elements and attributes in an XML/html documents.
- cssSelector:
    - We can use the same technique the way how developers access elements for styling

Sample Program Using Locators

```
public static void main(String[] args) {

    //opening browser
    WebDriver driver = DriverFactory.getDriver("edge");

    //open application
    DriverFactory.openApplication(driver,
"http://opensource.demo.orangehrmlive.com");

    //enter text on username
    driver.findElement(By.id("txtUsername")).sendKeys("admin");

    //enter text on password
    driver.findElement(By.name("txtPassword")).sendKeys("admin");

    //click on login button
    driver.findElement(By.id("btnLogin")).click();

    //click on pim link
    driver.findElement(By.linkText("PIM")).click();

    //click on AddEmployee Link
    driver.findElement(By.partialLinkText("Add Emp")).click();
```

```
        //enter text on firstname
        driver.findElement(By.id("firstName")).sendKeys("selenium");

        //enter text on last name
        driver.findElement(By.id("lastName")).sendKeys("hq");

        //click on save button
        driver.findElement(By.id("btnSave")).click();

        //close browser
        driver.quit();
    }
```

XPath Locator

- Xpath is used to navigate between the elements in xml or html documents
- Using this we can access the elements using any attribute
- We have to prepare correct xpath of the element
- there are two types in xpath
    - Absolute XPath:
        - The complete hierarchy of an element
        - html/body/table/tbody/tr[1]/td[2]/input
        - / : we must specify immediate child
    - Relative Xpath:
        - We create xpath using the elements attributes
        - we use // for directly accessing any child from that page
- //tagName: //input : returns all input tag elements from that page
- @ : is used to specify attribute
- Syntax for creating relative XPath //tagName[@attributeName='AttributeValue']
- //input[@id='uname'] : selects an input element which has uname as id
    - driver.findElement(By.xpath("//input[@id='uname']")).sendKeys("abcd");
- text() : is used specify innertext. if it is not working use "."
    - //a[text()='gmail'] : selects a link with gmail text
    - <a href='http://gmail.com' > <b>Gmail</b> </a>
- contains : used to specify partial text
    - //a[contains(text(),'Add em')] : This will select a link which has text Add em
    - //*[contains(@id,'una')] : This will select an element which id has una
- Indexing:
    - //input[1] : select first input element in its parent
    - //td[2] : selects second td in its parent
    - //td[last()] : selects last td in its parent
- Using starts with:
    - //input[starts-with(@id,"txt")] : This will select an element which id starts with txt

  - ends-with will not work for now as it introduced in xpath 2.0. all browsers are supporting xpath 1.0

```
public static void main(String[] args) {

    WebDriver driver = WebUtils.getDriver("chrome");
    WebUtils.openApplication(driver, "http://opensource.demo.orangehrmlive.com/");

    driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("admin");
    driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin");
    driver.findElement(By.xpath("//input[@value='LOGIN']")).click();

    driver.findElement(By.xpath("//a[.='PIM']")).click();
    driver.findElement(By.xpath("//a[contains(text(),'Add Emp')]")).click();

    driver.findElement(By.xpath("//input[@id='firstName']")).sendKeys("selenium");
    driver.findElement(By.xpath("//input[@id='lastName']")).sendKeys("hq");
    driver.findElement(By.xpath("//input[@id='btnSave']")).click();
}
```

CSS Selector

- CSS Selector is a technique to access elements by developer for styling purpose
- To access an element by id then use #
- To access an element by classname then use .
- in xpath : //input[@id='uname']
- In css : **#uname** or **input[id='uname']** or **input#uname**
- Using attributes in CSS Selector
  - tagname[attributename='attributevalue']
  - input[id='uname']
- With respect to element identification in selenium xpath is having more features, css is having good performance
- Which technique to be used in real time:
  - id -> name -> css -> xpath

```
public static void main(String[] args) {

    WebDriver driver = WebUtils.getDriver("chrome");
    WebUtils.openApplication(driver, "http://opensource.demo.orangehrmlive.com/");

    driver.findElement(By.cssSelector("input[id='txtUsername']")).sendKeys("admin"
);
    driver.findElement(By.cssSelector("input[id='txtPassword']")).sendKeys("admin"
);
    driver.findElement(By.cssSelector("input[value='LOGIN']")).click();

    driver.findElement(By.linkText("PIM")).click();
    driver.findElement(By.xpath("//a[contains(text(),'Add Emp')]")).click();
    driver.findElement(By.cssSelector("input[id='firstName']")).sendKeys("selenium
");
    driver.findElement(By.cssSelector("input[id='lastName']")).sendKeys("hq");
    driver.findElement(By.cssSelector("input[id='btnSave']")).click();
}
```

XPath VS CSS Syntax

FindElement	Xpath	CSS	Comments
Using Id Attrubute	//input[@id='txtUsername']	input#txtUsername input[id='txtUsername']	
Using Class Attribute	//div[@class='classname']	div.classname div[class='classname']	
Using multiple classes	//div[@class='Class1 Class2']	div.Class1.Class2 div[class='Class1 Class2']	
Using Name Attribute	//div[@name='value']	div[name='value']	
Select immediate child in parent	//td/input	td > input	
Select second element inside a parent	//td/input[2]	td input:nth-of-type(2)	
Goto the parent of an element	//input[@id='txtUsername']/..	input#txtUsername:parent	It may not work for CSS
	//div[@title="Professional Edition"]/parent::*		
Goto Any parent in element hierarchy	//div[@title='Professional Edition']/ancestor::*		
Goto next sibling	//div[@title="Professional Edition"]/following-sibling::div[1]	li.blue + li	
goto Previous sibling	//div[@title="Professional Edition"]/preceding-sibling::div[1]		
Using contains	//a[contains(text(),'Sign in')]	a:contains('Sign in')	It may not work for CSS
	//a[contains(string(),'Sign in')]		
Using starts with	//input[starts-with(@id,"txt")]	input[id^='txt']	
Using ends with	//input[ends-with(@id,"suffixString")]	input[id$='suffixString']	ends-with is introduced xpath 2.0 It depends xpath

			engine what we are using

Testing Xpath Or CSS Selectors in Browsers

- We can test xpath/css in firefox and chrome using a built in facility called console
- In chrome we can also use inspect window find functionality to verify xpath and css
- In firefox we can use inspect window find functionality for only css
- In chrome/firefox console we can use the command $x("xpath"); to verify xpath and $$("css"); to verify css
- This command will return array of elements
- Using console:
    - inspect element -> click on console tab
    - Give the command -> press enter
    - it returns array of elements
    - mouse hover return element to highlight in application
    - click on element to see html code
- In chrome we have an extra facility to copy xpath/css
- inspect element -> right click on element html code -> copy -> copy xpath/selector
- it generates relative xpath if the element has id
- if element is not having id then with in its hierarchy if any parent of the element has id then the xpath starts from that parent
- If there is no id for any parent element then it gives absolute xpath.
- Note: In webdriver code or in console we must specify single quotes in side of xpath.
- In chrome inspect search it accepts double quotes as well.
    - //*[@id="uname"] this is accepted in chrome inspect element
    - //*[@id='uname'] this is accepted in chrome inspect element
    - $x("//*[@id="uname"]") this is not accepted in console
    - $x("//*[@id='uname']") this is accepted in console
- We can also use extensions or addons for testing xpath/css in firefox or chrome
- In chrome / firefox there are extension and addons given by third party people to test xpath and css
- In firefox there is firepath extension. To use this we must install firebug add on.
- install firebug:
    - Goto https://addons.mozilla.org/en-US/firefox/addon/firebug/
    - click on add to firefox
- install firepath:
    - Goto : https://addons.mozilla.org/en-US/firefox/addon/firepath/
    - add to firefox
- Using firepath: right click on element -> inspect in firepath
- For chrome we have some extensions like xpath helper. But no extension is better than in built functionality of chrome inspect search.

XPath Advanced

- Sometimes multiple elements will have same attribute values or no attributes defined in its html.
- In this case we need to work on creating unique xpath
- To find an element which is not having an identity, We find the related elements of that element and access that element using a relation between identified element
- Xpath using self attributes
    - //tagname[@attribute='attributevalue']
- If Parent is having unique identification
    - parentXpath/ActualelementXpath
    - parentXpath//ActualelementXpath
- If immediate child is having unique identification
    - childXpath/..
    - childXpath/parent::*
- If any child in parent is having unique identification
    - childXpath/ancestor::*
    - childXpath/ancestor-or-self::ActualelementXpath
    - childXpath/ancestor::*[1] it selects first parent
    - //label[text()='License Expiry Date']/..//img[@class='ui-datepicker-trigger']
- If the next element is having unique identification
    - nextElementXpath/preceding-sibling::*[1]
    - nextElementXpath/preceding-sibling::ActualelementXpath
- If the previous element is having unique identification
    - previousElementXpath/following-sibling::*[1]
    - previousElementXpath/following-sibling::ActualelementXpath
    - //label[text()='Date of Birth']/following-sibling::*[2]
    - //label[text()='Date of Birth']/following-sibling::img[@class='ui-datepicker-trigger']
    - //label[text()='License Expiry Date']/following-sibling::img[@class='ui-datepicker-trigger']
- Practice Site: http://demo.servicedeskplusmsp.com/
- http://opensource.demo.orangehrmlive.com/index.php/pim/viewPersonalDetails/empNumber/0005

XPath Visible Element

- **Scenario to get Hidden Duplicate Elements**
- Goto https://www.redbus.in/
- click on user icon
- click on sign in/sign up link
- try to find the close button element in the opened html popup
- You can find it using //div[@class="modalCloseSmall"]
- close the html popup and Repeat from step 2 again
- Now try to find it
- it will show two elements
- like that how many you close and opened those many elements will get found using xpath

- This is happening because developers hiding the element when you click on close
- when you click on sign in/sign up link the element is created again
- **Handling the Situation**
- To create xpath for only visible element we should first find what is attribute is used by developer to make that element hide
- **To find only visible elements**
- //div[@class="modalCloseSmall" and not(ancestor-or-self::*[contains(@class,'hide')])]
- if they use display none then
- //div[@class="modalCloseSmall" and not(ancestor-or-self::*[contains(@style,'display:none')])]
- **To find hidden elements:**
- //div[@class="modalCloseSmall" and ancestor-or-self::*[contains(@class,'hide')]]
- //div[@class="modalCloseSmall" and ancestor-or-self::*[contains(@style,'display:none')]]