LOCATING WEB ELEMENTS AND ACTIONS ON WEB ELEMENTS:-

Selenium defines two methods for identifying web elements:

1. findElement: A command to uniquely identify a web element within the web page.
2. findElements: A command to identify a list of web elements within the web page.

There are 8 locators strategies included in Selenium:

- Identifier.
- Id.
- Name.
- Link.
- DOM.
- XPath.
- CSS.
- UI-element.

A Web element locator is an object that finds and returns Web elements on a page using a given query. In short, locators find elements.

Why are locators needed?

As human users, we interact with Web pages visually: We look, scroll, click, and type through a browser.

play a major role while testing an application. The first thing to do is to locate these elements on the web page. I'll be covering various options on how to find elements in Selenium that help in automation testing and data loading activities.

Why do we need to Find Element or FindElements?

Selenium is used for automatic data loading and regression testing of a website. As part of this automation feature interaction with a web page requires the driver to locate the web element and either trigger a *JavaScript* event like-click, enter, select, etc or type in the field value.

Find Element command is used to uniquely identify a (one) web element within the web page. Whereas, Find Elements command is used to uniquely identify the list of web elements within the web page.

Difference between "FindElement" and "FindElements"

Find Element	Find Elements
Returns the first matching web element if multiple web elements are discovered by the locator	Returns a list of matching web elements
Throws NoSuchElementException if the element is not found	Returns an empty list if no matching element found
This method is used only to detect a unique web element	This method is used to return a collection of matching elements.

There are multiple ways to uniquely identify a web element/elements within the web page such as ID, Name, Class Name, Link Text, Partial Link Text, Tag Name and XPATH.

Locator Strategy/ Types of locators

Locator Strategy can be one of the following types to find an element or FindElements –

- ID
- Name
- ClassName
- TagName
- Link Text/Partial Link Text
- CSS Selector
- XPATH Selector

Let us now try to see how each of these strategies can be used to find an element or find elements. First, we'll see about finding the

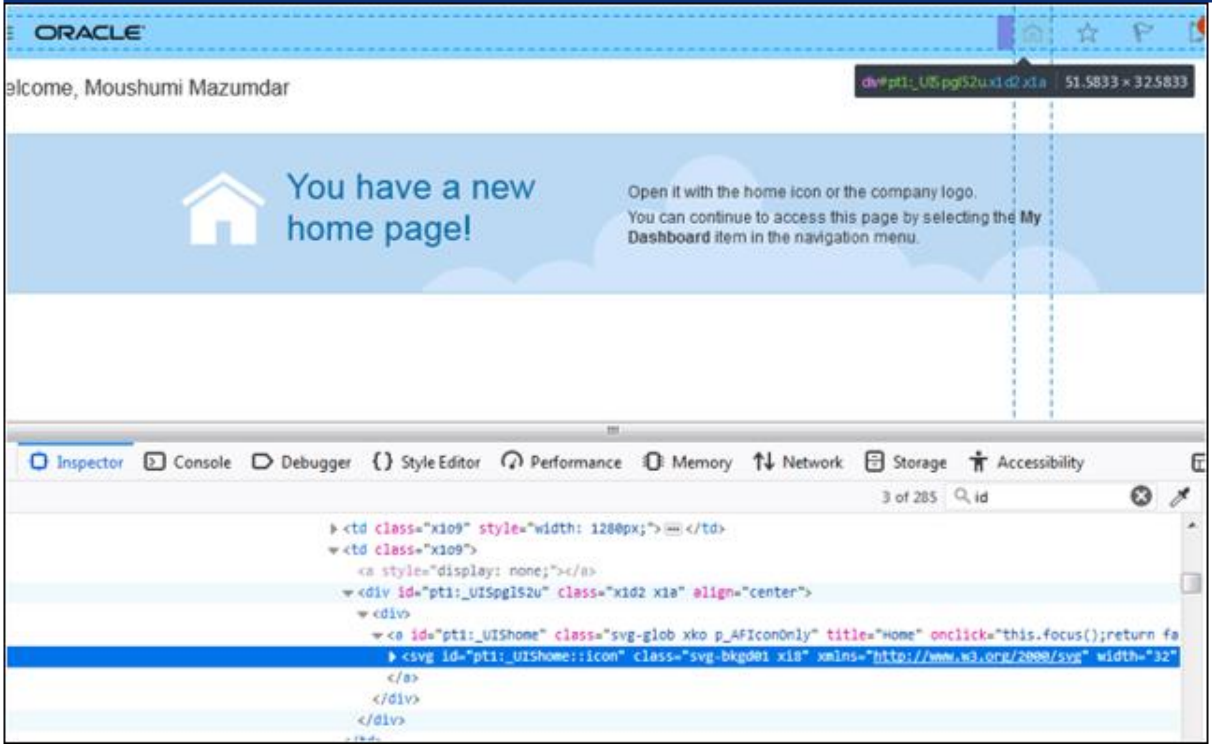
Find by ID

ID's are unique for each element so it is a common way to locate elements using ID Locator. It is the most common fastest and safest way to detect an element. It is recommended for website developers to avoid using non-unique Ids or dynamically generated Ids however some MVC frameworks like – ADF can lead to pages with dynamically generated ids.

If any website has non-unique Ids or has dynamically generated ids then this strategy can't be used to uniquely find an element, instead, it will return the first web element which matches the locator. How we can overcome such situations, will be explained in the XPATH/CSS selector strategy.

Syntax:

```
public class LocateByID
{
public static void main (String [] args)
{
// Open browser
WebDriver driver = new FirefoxDriver();
//instance of Chrome | Firefox | IE driver
driver.get(<url>);
// Open Application
WebElement elm = driver.findElement(By.id("pt1:_UIShome::icon"));
// will raise NoSuchElementException if not found
elm.click()
//e.g- click the element
}
}
```


ORACLE
elcome, Moushumi Mazumdar
div#pt1:_UISpgl52u.x1d2.x1a 51.5833 × 32.5833
You have a new home page!
Open it with the home icon or the company logo.
You can continue to access this page by selecting the My Dashboard item in the navigation menu.
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Accessibility
3 of 285
id

```
<td class="x1o9" style="width: 1280px;">…</td>
<td class="x1o9">
  <a style="display: none;"></a>
  <div id="pt1:_UISpgl52u" class="x1d2 x1a" align="center">
    <div>
      <a id="pt1:_UIShome" class="svg-glob xko p_AFIconOnly" title="Home" onclick="this.focus();return fa
        <svg id="pt1:_UIShome::icon" class="svg-bkgd01 x18" xmlns="http://www.w3.org/2000/svg" width="32"
      </a>
    </div>
  </div>
```

Now let’s understand how to find an element using a name.

Find by Name

This method is similar to Find By Id except the driver will try to locate an element by “name” attribute instead of “id” attribute.

Syntax:

```
public class LocateByName
{
public static void main (String [] args)
{
// Open browser
WebDriver driver = new FirefoxDriver();
//instance of Chrome | Firefox | IE driver
driver.get(<url>);
// Open Application
WebElement elm = driver.findElement(By.name("name"));
```

```
public class LocateByClass {

public static void main (String [] args){

// Open browser

WebDriver driver = new FirefoxDriver();//instance of Chrome | Fire

driver.get(<url>);// Open Application

List<WebElement> links = driver.findElements(By.className("svg-bkgd01 xi8"));//return
// loop over the list and perform the logic of a single e
}
}
```

```
// will raise NoSuchElementException if not found
elm.sendKeys("Hi");
//e.g - type Hi in the detected field
}
}
```

Now let’s move ahead and understand how to find elements in Selenium using the className.

Find by ClassName

This method finds elements based on the value of the *CLASS* attribute. More applicable for locating multiple elements which has a similar css class defined against them.

Syntax:

driver.findElements(By.**className**(<locator_value>)) ;//for list of elements

or

driver.findElement(By.**className**(<locator_value>)) ;//single web element

Now let’s understand how to find elements in Selenium using TagName.

Find by Tag Name

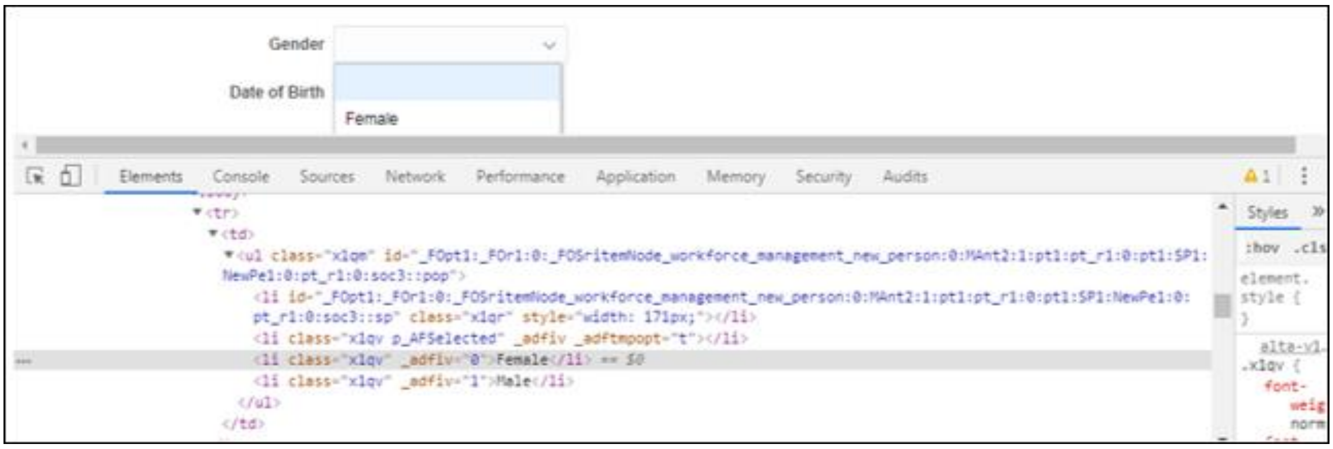
This method finds elements based on the HTML tag name of the element. This is not widely used and used as the last resort if the particular web element can’t be detected by Id/name/link/className/XPATH/CSS.

Syntax:

driver.findElement(By.**tagName**(<locator_value>)) ;//single web element or

driver.findElements(By.**tagName**(<locator_value>)) ;//for list of elements

```
public class LocateByTagName{

public static void main (String [] args){

// Open browser

WebDriver driver = new FirefoxDriver();//instance of Chrome | Firefox | IE
driver

driver.get(<url>);// Open Application

WebElement ul = driver.findElement(By.id(<id>));

List<WebElement> links = ul.findElements(By.tagName("li"));

...

}

}
```



```
▼<tr>
  ▼<td>
    ▼<ul class="x1qm" id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:
    NewPe1:0:pt_r1:0:soc3::pop">
        <li id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:
        pt_r1:0:soc3::sp" class="x1qr" style="width: 171px;"></li>
        <li class="x1qv p_AFSelected" _adfiv _adftmpopt="t"></li>
        <li class="x1qv" _adfiv="0">Female</li> == $0
        <li class="x1qv" _adfiv="1">Male</li>
      </ul>
    </td>
```

This is about how to find an element using TagName. Let’s move ahead and take a look at how to find elements using LinkText

TextFind by Link Text/Partial Link

With this method, one can find elements of *“a” tags (*Link*)* with the link names or having matching partial link names. This strategy is only applicable in finding element(s) of type anchor tags which contain a text value.

Syntax

driver.findElement(By.**linkText**(<link_text>)) ;//single web element or

driver.findElements(By.**linkText**(<link_text>)) ;//for list of elements

driver.findElement(By.**partialLinkText**(<link_text>)) ;//single web element

or

driver.findElements(By.**partialLinkText**(<link_text>)) ;//for list of elements

This is about how to find elements in Selenium using LinkText. Now let’s understand how to find elements in Selenium using CSS Selector.

Find by CSS Selector

For websites generating dynamic Ids like ADF based applications or websites which are built on latest javascript frameworks like – React js which may not generate any Ids or names can’t use locator by Id/Name strategy to find elements. Instead, we have to use either CSS selector or XPath selectors.

Choosing a *CSS* selector over *XPath* selector for the sake of performance is a myth now. One can choose a hybrid approach. For simple screens CSS selectors(forward only) is preferred over XPATH, however, for complex traversal (forward/backward and complex search conditions) XPATH is the only choice.

CSS selectors have native browser support, so on occasion basis, it can turn out to be faster than XPATH selector.

XPATHSelector

XPATH is more readable and the learning curve is less steep since it uses standard XML query syntaxes, however, CSS selectors though have simpler syntax support but are not standard like XPATH and other documentation support, unlike XPATH.

Following are some of the mainly used formats of CSS Selectors –

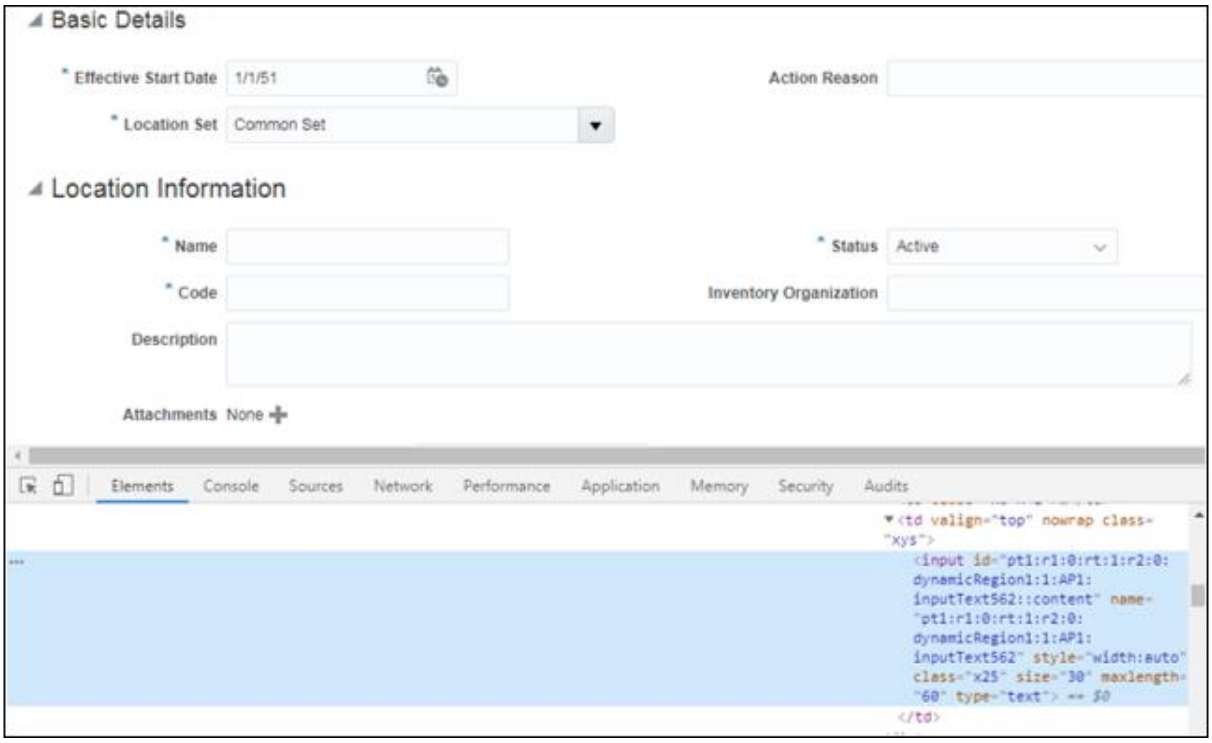
- Tag and
- ID
- Tag and Class
- Tag and Attribute
- Tag, Class, and Attribute
- Sub-String Matches
    - Starts With (^)
    - Ends With ($)
    - Contains (*)
- Child Elements
    - Direct Child
    - Sub-child
    - nth-child

Refer below screenshot –

Tag with ID

css= tag # id

```
public class LocateByCSSSelector
{
public static void main (String [] args)
{
WebDriver driver = new FirefoxDriver()
//instance of Chrome | Firefox | IE driver
driver.get(<url>);
// Open Application
WebElement el = driver.findElement(By.cssSelector("input#pt1:r1:0:rt:1:r2:0:dynamicReg
el.sendKeys("Location1");
}
}
```



1/1/51

Common Set

Active

None

```
▼<td valign="top" nowrap class=
"xys">
  <input id="pt1:r1:0:rt:1:r2:0:
  dynamicRegion1:1:AP1:
  inputText562::content" name=
  "pt1:r1:0:rt:1:r2:0:
  dynamicRegion1:1:AP1:
  inputText562" style="width:auto"
  class="x25" size="30" maxlength=
  "60" type="text"> == $0
</td>
```

Tag and class

```
css   = tag.class

public static void main  (String  []  args)
{

WebDriver  driver  =  new FirefoxDriver();
//instance  of  Chrome  |  Firefox  |  IE  driver

driver.get(<url>);
//  Open  Application

WebElement  el  =  driver.findElement(By.cssSelector("input.x25"));

el.sendKeys("Location1");

}

}
```

Tag and attribute

```
css  =  tag[attribute=value]

public class LocateByCSSSelector{
public static void main  (String  []  args){
WebDriver  driver  =  new FirefoxDriver();//instance  of  Chrome
driver.get(<url>);//  Open  Application
WebElement  el  =  driver.findElement(By.cssSelector("input[name='pt1:r1:0:rt:1:r2:0:dyna
el.sendKeys("Location1");
}
}
```

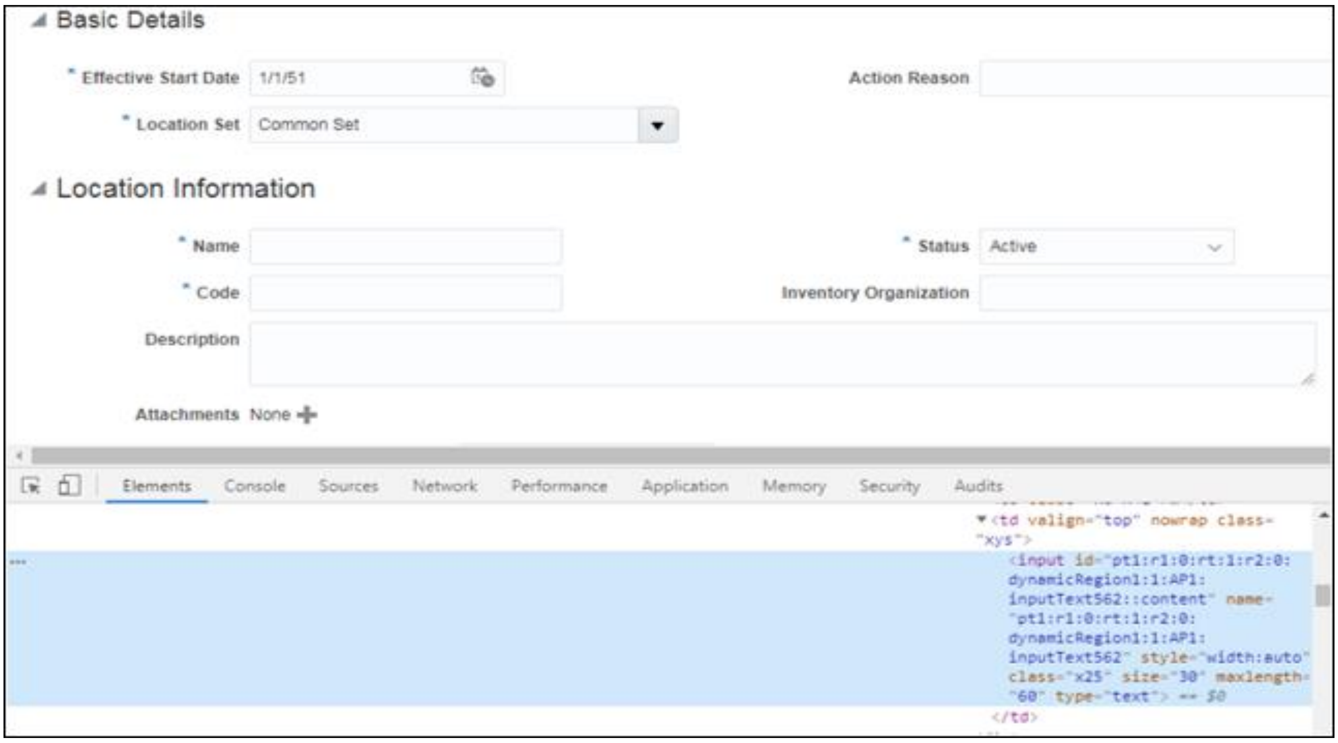
Tag, class, and attribute

```
css = tag.class[attribute=value]

public class LocateByCSSSelector
{

public static void main (String [] args)
{

WebDriver driver = new FirefoxDriver
//instance of Chrome | Firefox | IE driver
driver.get(<url>);
// Open Application

WebElement el = driver.findElement(By.cssSelector("input.x25[name='pt1:r1:0:rt:1:r2:0
el.sendKeys("Location1");
}

}
```

Substring matches

Starts with –

```
public class LocateByCSSSelector
{
public static void main (String [] args)
{
WebDriver driver = new FirefoxDriver(); //instance of Chrome | Firefox | IE driver
driver.get(<url>); // Open Application
WebElement el = driver.findElement(By.cssSelector("input[name^='pt1:r1:0:rt']"));
el.sendKeys("Location1");
}
}
```



```
▼<td valign="top" nowrap class=
"xys">
  <input id="pt1:r1:0:rt:1:r2:0:
  dynamicRegion1:1:AP1:
  inputText562::content" name=
  "pt1:r1:0:rt:1:r2:0:
  dynamicRegion1:1:AP1:
  inputText562" style="width:auto"
  class="x25" size="30" maxlength=
  "60" type="text"> == $0
</td>
```

Ends with –

```
<span style="font-weight: 400">[attribute^=prefix of the string]</span>
public class LocateByCSSSelector
{

public static void main (String [] args)
{

WebDriver driver = new FirefoxDriver();
//instance of Chrome | Firefox | IE driver
driver.get(<url>);
// Open Application

WebElement el =
driver.findElement(By.cssSelector("input[name$='1:AP1:inputText562']"));
el.sendKeys("Location1");

}

}
```

Refer same example screenshot above.

Contains

```
public class LocateByCSSSelector
{

public static void main (String [] args)
{
WebDriver driver = new FirefoxDriver();
//instance of Chrome | Firefox | IE driver
driver.get(<url>);// Open Application
WebElement el = driver.findElement(By.cssSelector("input[name*='AP1']"));
el.sendKeys("Location1");
}
}
```

Alternately the above syntax can be written as below –

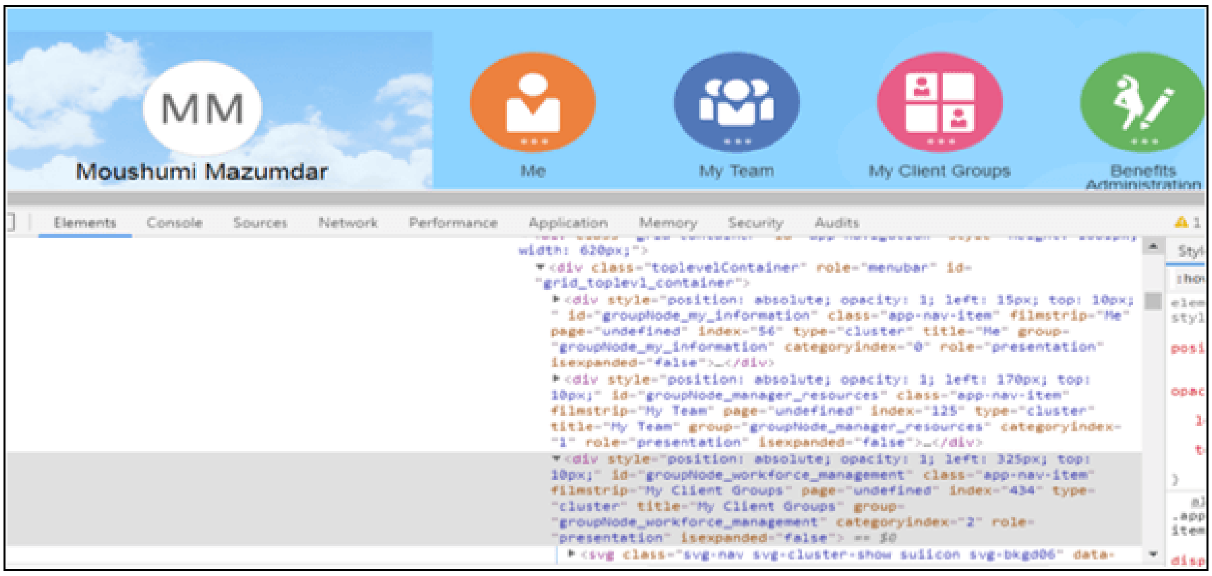
```
public class LocateByCSSSelector
{
public static void main (String [] args)
{
//instance of Chrome | Firefox | IE driver
WebDriver driver = new FirefoxDriver();
driver.get(<url>);// Open Application
WebElement el = driver.findElement(By.cssSelector("input:contains('AP1')]"));
el.sendKeys("Location1");
}
}
```

Locating child elements(direct child/sub child)

Syntax:

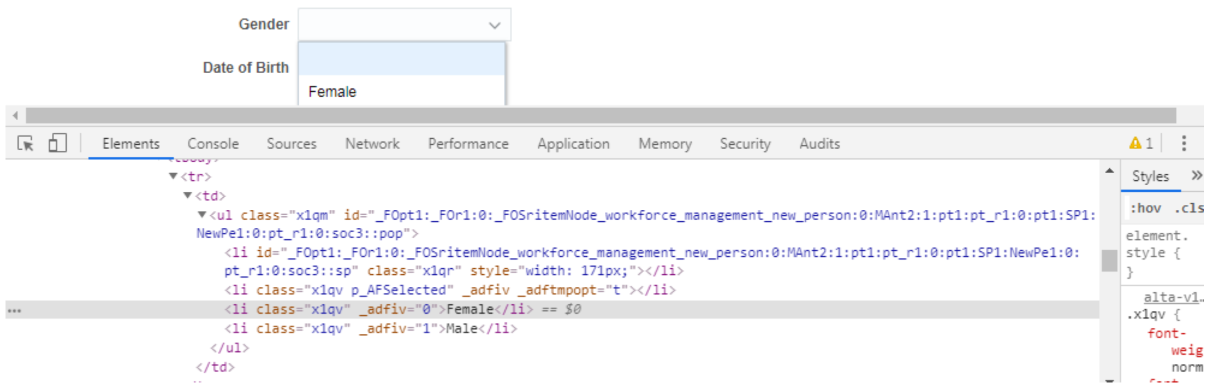
parentLocator>childLocator

```
public class LocateByCSSSelector
{
public static void main (String [] args)
{
WebDriver driver = new FirefoxDriver();//instance of Chrome | Firefox | IE driver
driver.get(<url>);// Open Application
WebElement el = driver.findElement(By.cssSelector(“div#grid_toplevl_container >
div#groupNode_workforce_management”));
el.click();
}
}[/java]
```



```
width: 620px;">
  ▼<div class="toplevelContainer" role="menubar" id=
  "grid_toplevl_container">
    ▶<div style="position: absolute; opacity: 1; left: 15px; top: 10px;
    " id="groupNode_my_information" class="app-nav-item" filmstrip="Me"
    page="undefined" index="56" type="cluster" title="Me" group=
    "groupNode_my_information" categoryindex="0" role="presentation"
    isexpanded="false">…</div>
    ▶<div style="position: absolute; opacity: 1; left: 170px; top:
    10px;" id="groupNode_manager_resources" class="app-nav-item"
    filmstrip="My Team" page="undefined" index="125" type="cluster"
    title="My Team" group="groupNode_manager_resources" categoryindex=
    "1" role="presentation" isexpanded="false">…</div>
    ▼<div style="position: absolute; opacity: 1; left: 325px; top:
    10px;" id="groupNode_workforce_management" class="app-nav-item"
    filmstrip="My Client Groups" page="undefined" index="434" type=
    "cluster" title="My Client Groups" group=
    "groupNode_workforce_management" categoryindex="2" role=
    "presentation" isexpanded="false"> == $0
      ▶<svg class="svg-nav svg-cluster-show suiicon svg-bkgd06" data-
```

- Sub child
- Same as before only the locator can be a direct child/sub child
- Nth child
- Using nth-of-type



```
▼<tr>
  ▼<td>
    ▼<ul class="x1qm" id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:
    NewPe1:0:pt_r1:0:soc3::pop">
        <li id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:
        pt_r1:0:soc3::sp" class="x1qr" style="width: 171px;"></li>
        <li class="x1qv p_AFSelected" _adfiv _adftmpopt="t"></li>
        <li class="x1qv" _adfiv="0">Female</li> == $0
        <li class="x1qv" _adfiv="1">Male</li>
      </ul>
    </td>
```

For detecting "Female" from the above li dropdown

```
public class LocateByCSSSelector
{

public static void main (String [] args)
{

WebDriver driver = new FirefoxDriver();//instance of Chrome |
Firefox | IE driver
driver.get(<url>);// Open Application

WebElement el = driver.findElement(By.cssSelector("ul#_FO... li:nth-of-
type(2)"));
el.click();
}

}
```

Alternatively, you can check out the Automation Engineer Course by Edureka and get certified!

Find by XPATH selector

In our test automation codes, we generally prefer to use id, name, class, etc. these kinds of locators. However, sometimes we can not find any of them in the DOM and also sometimes locators of some elements change dynamically in the DOM. In these kinds of situations, we need to use smart locators. These locators must be capable to locate complex and dynamically changing web elements.

Recently when I was working on automation of regression testing of Oracle Fusion SaaS screens, I was struggling to identify a methodology of locating web elements. The same version of SaaS instance across various environments was generating different Ids.XPATH selectors came to my rescue and I mostly used contains() option to locate the web elements.

There are also other tactics of writing XPATH selectors. These are briefly explained below –

Absolute and Relative XPath

Absolute	Relative
A direct way to locate an element	Starts from the middle of the DOM element
Brittle can break if the path of accessing the element changes due to the position	Relatively stable since the search is relative to DOM
Starts with “/” and from the root	Starts with “//” and it can start search anywhere in the DOM
Longer XPATH expressions	Shorter expressions

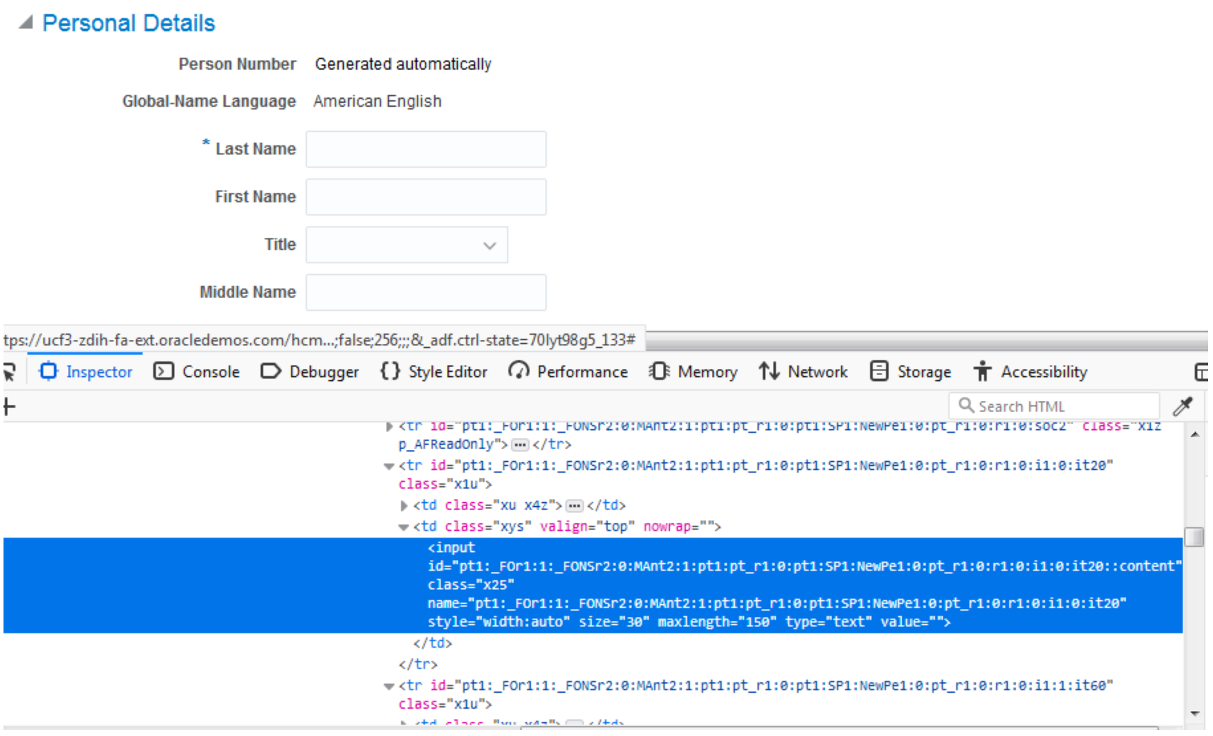
```
//tag[@attribute='value']

public class LocateByXPATHSel
{

public static void main (String [] args)
{

WebDriver driver = new FirefoxDriver();//instance of Chrome | Firefox | IE driver
driver.get(<url>);// Open Application

WebElement el = driver.findElement(By.xpath("xpath=//button[@id='pt1:r1:0:r0:1:AP1:APb']")); // trying to locate a buttton

el.click();

}
```

}

Using contains()

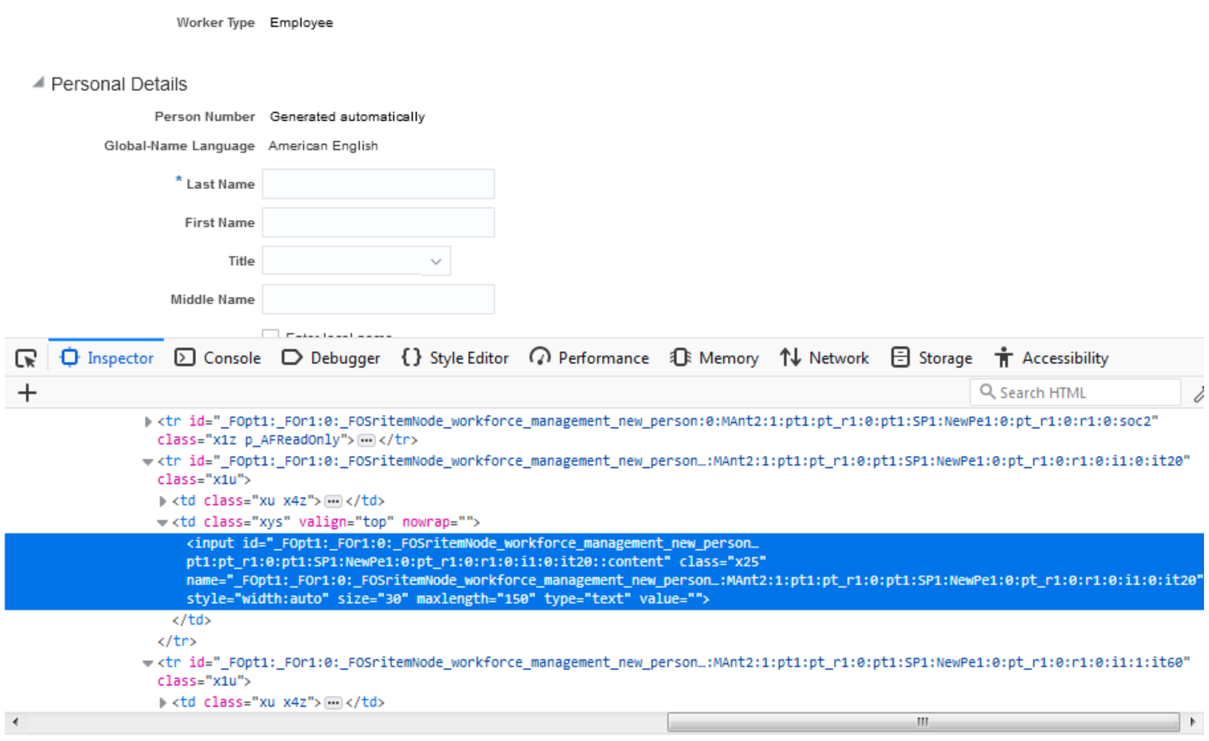
It is very handy XPath Selenium locator and sometimes it saves the life of a test automation engineer. When an attribute of an element is dynamic, then we can use contains() for the constant part of the web element but also you can use contains() in any condition when you need.

Fusion Instance#1



```
<tr id="pt1:_FOr1:1:_FONSr2:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:soc2" class="x1z
p_AFReadOnly">…</tr>
<tr id="pt1:_FOr1:1:_FONSr2:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20"
class="x1u">
  <td class="xu x4z">…</td>
  <td class="xys" valign="top" nowrap="">
    <input
    id="pt1:_FOr1:1:_FONSr2:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20::content"
    class="x25"
    name="pt1:_FOr1:1:_FONSr2:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20"
    style="width:auto" size="30" maxlength="150" type="text" value="">
  </td>
</tr>
<tr id="pt1:_FOr1:1:_FONSr2:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:1:it60"
class="x1u">
```

Fusion Instance#2



Worker Type Employee

Personal Details

Person Number Generated automatically

Global-Name Language American English

* Last Name

First Name

Title

Middle Name

```
<tr id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:soc2"
class="x1z p_AFReadOnly"> ··· </tr>
<tr id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person…:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20"
class="x1u">
  <td class="xu x4z"> ··· </td>
  <td class="xys" valign="top" nowrap="">
    <input id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person…
    pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20::content" class="x25"
    name="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person…:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20"
    style="width:auto" size="30" maxlength="150" type="text" value="">
  </td>
</tr>
<tr id="_FOpt1:_FOr1:0:_FOSritemNode_workforce_management_new_person…:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:1:it60"
class="x1u">
  <td class="xu x4z"> ··· </td>
```

If we compare the same field it has 2 dynamically generated Ids –

//input[@id='pt1:_FOr1:1:_FONSr2:0:MAnt2:1:pt1:pt_r1:0:pt1:SP1:NewPe1:0:pt_r1:0:r1:0:i1:0:it20::content']

Starts-with

This method checks the starting text of an attribute. It is very handy to use when the attribute value changes dynamically but also you can use this method for non-changing attribute values. This comes handy when the prefix part of the id of the dynamic web element is constant.

Syntax:

//tag[starts-with(@attribute, 'value')]

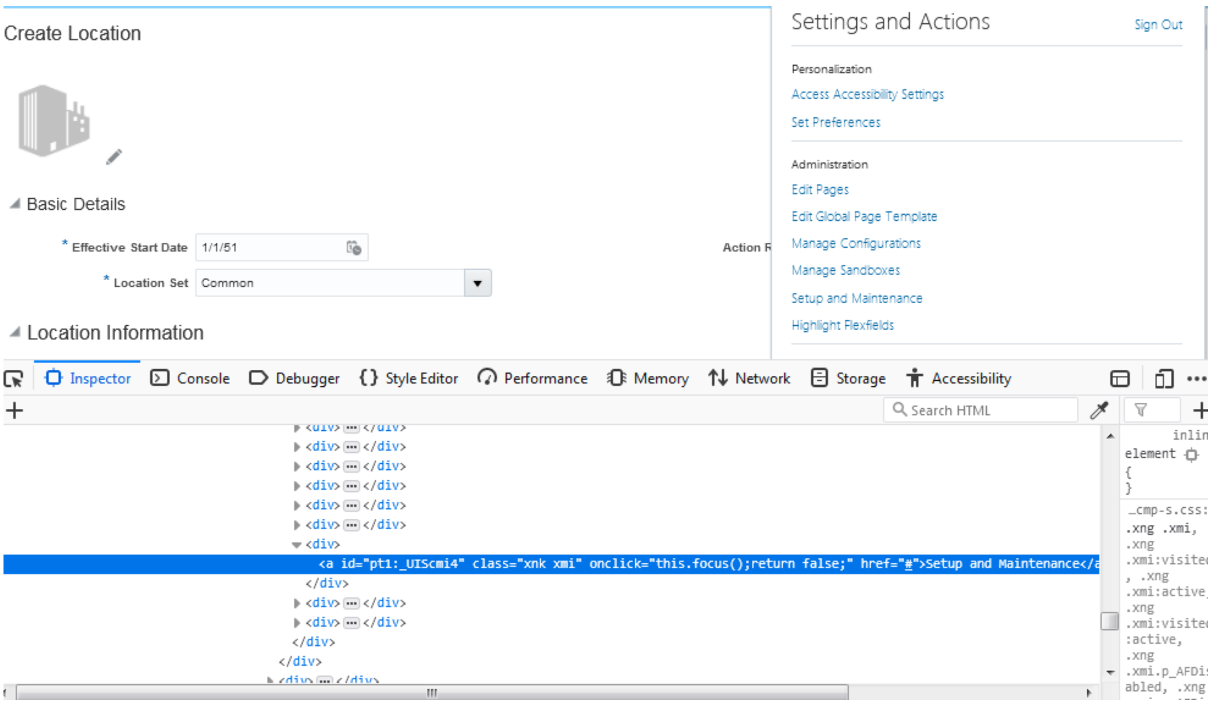
Example:

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

Chained Declarations

We can chain multiple relative XPath declarations with "//" double slash to find an element location as shown below.

xpath=//div[@id='pt1:_USSpgl5′]//a[@id='pt1:_UIScmi4′]



```
<div>…</div>
<div>…</div>
<div>…</div>
<div>…</div>
<div>…</div>
<div>
  <a id="pt1:_UIScmi4" class="xnk xmi" onclick="this.focus();return false;" href="#">Setup and Maintenance</a
</div>
<div>…</div>
<div>…</div>
</div>
</div>
```

Combining 'and' 'or' operators

Referring the same screenshot above we can write a condition as below –

xpath=//a[@id='pt1:_UIScmi4′ or @class='xnk xmi']

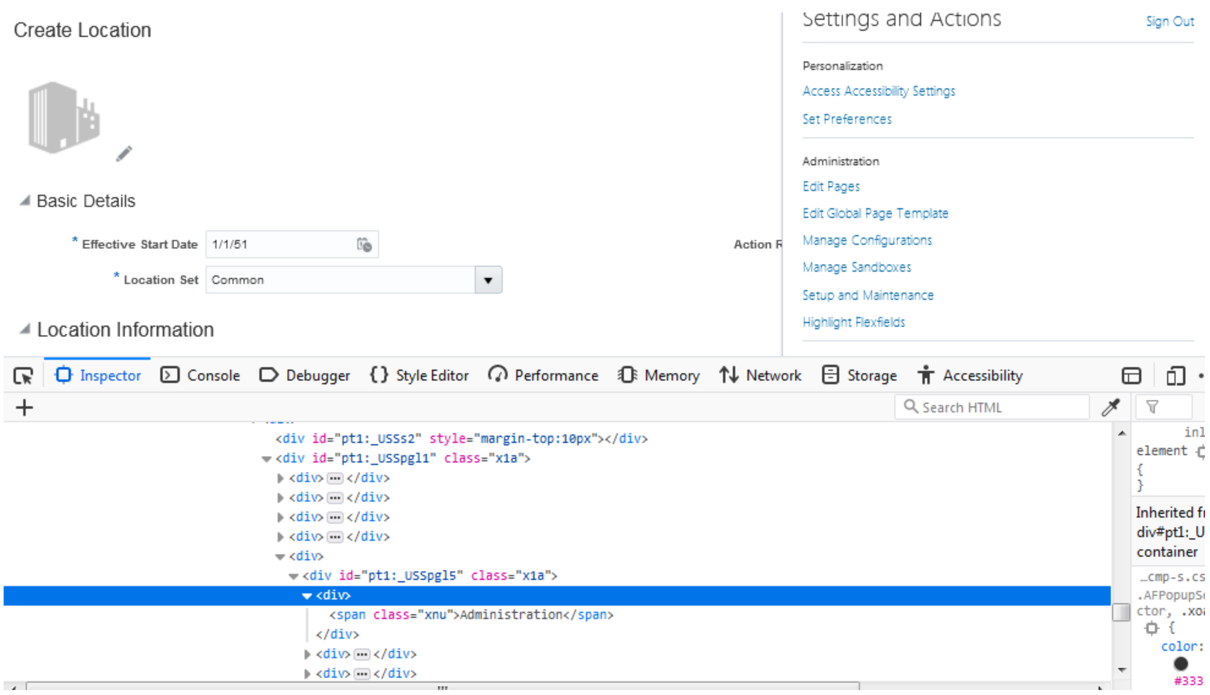
xpath=//a[@id='pt1:_UIScmi4′ and @class='xnk xmi']

Ancestor

We can use this option to find web elements with the help of the ancestor of a particular web element.

Following-sibling

Select the following siblings of the context node.



```
<div id="pt1:_USSs2" style="margin-top:10px"></div>
<div id="pt1:_USSpgl1" class="x1a">
  <div>…</div>
  <div>…</div>
  <div>…</div>
  <div>…</div>
  <div>
    <div id="pt1:_USSpgl5" class="x1a">
      <div>
        <span class="xnu">Administration</span>
      </div>
      <div>…</div>
      <div>…</div>
```

Example:

//span[@class='xnu']/ancestor::div[@id='pt1:_USSpgl5']/following-sibling::div

In the above example we are trying to access all menus under "Administration".

Following

Starts to locate elements after the given parent node. It finds the element before the following statement and set as the top node and then starts to find all elements after that node.

Syntax:

//tagName[@attribute=value]//following::tagName

Example:

//div[@id='xx']//following::input

So basically the search will start from div whose id='xx' and search all elements with tagname ='input' following the div tag.

Child

Selects all children elements of the current node.