Sub UpdateIE(ByVal userPass As String)

Dim URL As String

Dim userName As String

Dim Tag As Object

Dim tagx As Object

Dim IE As Object

Set IE = New InternetExplorerMedium

URL = "https://11.15.7.215/login/plrepo/plrepo\_rptmain.jsp"

IE.Navigate URL

IE.Visible = True

While IE.Busy

DoEvents

Wend

Set Tag = IE.Document.getelementsbytagname("a")

For Each tagx In Tag

If tagx.ID = "overridelink" Then

tagx.Click

End If

Next

While IE.Busy

DoEvents

Wend

userName = wsMain.Range("MARKET\_WEB\_ID").Value

Set Tag = IE.Document.getelementsbytagname("input")

For Each tagx In Tag

If tagx.Name = "USER\_ID" Then

tagx.Value = userName

End If

Next

Set Tag = IE.Document.getelementsbytagname("input")

For Each tagx In Tag

If tagx.Name = "PASSWORD" Then

tagx.Value = userPass

End If

Next

Set Tag = IE.Document.getelementsbytagname("input")

For Each tagx In Tag

If tagx.Type = "submit" Then

tagx.Click

End If

Next

While IE.Busy

DoEvents

Wend

URL = "https://11.15.7.215/login/plrepo/PlrepoServlet?deptteam=FTD2374C&mode=form"

IE.Navigate URL

While IE.Busy

DoEvents

Wend

IE.Visible = True

delay (5)

Set Tag = IE.Document.getelementsbytagname("input")

' With Sheets("Sheet1")

With WsTrading

For Each tagx In Tag

Select Case tagx.Name

'update P/L

Case "PL\_ON0"

tagx.Value = .Range("D4").Value

Case "PL\_ON1"

tagx.Value = .Range("D5").Value

Case "PL\_FY0"

tagx.Value = .Range("E4").Value

Case "PL\_FY1"

tagx.Value = .Range("E5").Value

Case "PL\_FYPREVMONTH0"

tagx.Value = .Range("F4").Value

Case "PL\_FYPREVMONTH1"

tagx.Value = .Range("F5").Value

Case "AS\_OF\_DATE\_PL0"

tagx.Value = .Range("I4").Value

Case "AS\_OF\_DATE\_PL1"

tagx.Value = .Range("I5").Value

'Update Position

Case "POS0"

tagx.Value = .Range("E8").Value

Case "POS1"

tagx.Value = .Range("E9").Value

Case "POS2"

tagx.Value = .Range("E10").Value

Case "POS3"

tagx.Value = .Range("E11").Value

Case "POS4"

tagx.Value = .Range("E12").Value

Case "AS\_OF\_DATE\_POS0"

tagx.Value = .Range("G8").Value

Case "AS\_OF\_DATE\_POS1"

tagx.Value = .Range("G9").Value

Case "AS\_OF\_DATE\_POS2"

tagx.Value = .Range("G10").Value

Case "AS\_OF\_DATE\_POS3"

tagx.Value = .Range("G11").Value

Case "AS\_OF\_DATE\_POS4"

tagx.Value = .Range("G12").Value

End Select

Next

End With

End Sub

**Often-used VBA HTML object Properties**

**Properties – HTML example 1**

Consider the following snippet of HTML (from this site’s footer):

|  |
| --- |
| <**div** class="site-info">  <**span** class="site-title"><**a** rel="home" href="http://automatetheweb.net/">Automate the Web</**a**></**span**>  <**a** href="https://wordpress.org/">Proudly powered by WordPress</**a**>  </**div**> |

We can have VBA snag the **div** element like this (*ele* is just a variable name I’m making up):  
**Set ele = objIE.document.getElementsByClassName("site-info")(0)**

Read it like this: *"for all elements on the web page that have a class of ‘site-info’ get me the zero-ith (first) one, and attach it to the object variable ele"*.

Remember, once VBA runs that line of code it finds a **div** element with the class name *site-info*… then it tucks that div into the variable ele for later use…

… so then we can look at some of that div’s properties in the debug console like this…

Debug.Print ele.ClassName would return:

site-info

Debug.Print ele.TagName

div

Debug.Print ele.Id

(nothing … this element has no **id=””** attached to it)

Debug.Print ele.href

ERROR: Object does not support this method or property

(silly rabbit, div’s don’t have href’s!)

Debug.Print ele.innerText

Automate the Web

Proudly powered by WordPress

(this can also be written as Debug.Print ele.textContent)

Debug.Print ele.innerHTML

<span class="site-title"><a rel="home" href="http://automatetheweb.net/">Automate the Web</a></span>

<a href="https://wordpress.org/">Proudly powered by WordPress</a>

Debug.Print ele.outerHTML

<div class="site-info">

<span class="site-title"><a rel="home" href="http://automatetheweb.net/">Automate the Web</a></span>

<a href="https://wordpress.org/">Proudly powered by WordPress</a>

</div>

**Properties – HTML example 2**

Now let’s look at some **Properties** of a different element. In the upper-right of this page there’s a blue email *SUBSCRIBE* button. If we right-click it, then click **Inspect**, we see this HTML:

|  |
| --- |
| <**input** tabindex="923" type="submit" class="button btn btn-primary" name="submit"  value="subscribe" id="newsletters-2-button"> |

So let’s tell VBA to scrape that **input** element using the following line of code (*iEle* is just a made up variable name) …  
**Set iEle = objIE.document.getElementId("newsletters-2-button")**

Read like this: *"for the lone element on the page that has an****id****of ‘newsletters-2-button’, attach it to the object variable iEle"*.

When VBA runs that line of code it discovers that particular **id** belongs to an **input** element, and it holds this element in memory in the variable iEle.

So executing Debug.Print iEle.Id would produce:

newsletters-2-button

Debug.Print iEle.Type

submit

Debug.Print iEle.Name

submit

Debug.Print iEle.Value

subscribe

*.Value* is an example of a property that we can ***set*** as well as ***get***, meaning it’s not a *read-only* property. We could assign it a value (*write* to it) as well, using code like the following, which enters a string of text into the input box:  
objIE.document.getElementId("newsletters-2-button").Value = "hi@automatetheweb.net"

**Properties – HTML example 3**

|  |
| --- |
| <**div** class="header-image">  <**a** href="http://automatetheweb.net/" rel="home">  <**img** src="http://automatetheweb.net/wp-content/uploads/login-logo.jpg" width="350" height="40" alt="Automate the Web">  </**a**>  </**div**> |

This time let’s just access some elements directly — we won’t use a variable…

Debug.Print objIE.document.getElementsByTagName("a")(7).href  
(assumes it’s the 8th **<a>** tag on the page we’re after… remember 1 is 0 in variable land, 2 is 1, 3 is 2, and so on)

http://automatetheweb.net/

Debug.Print objIE.document.getElementsByTagName("img")(3).src  
(assumes it’s the 4th **<img>** tag on the page we’re after)

http://automatetheweb.net/wp-content/uploads/login-logo.jpg

**Often-used VBA HTML object Methods**

In addition to those amazing and exciting object **Properties** (think *attributes*, or *nouns*), VBA objects can have **Methods** (think *actions*, or *verbs*), too:

We use **Methods** like these to interact withn **HTML** objects:

objIE.document.getElementsByTagName("button")(0).click  
clicks the first button element on the page.

objIE.document.getElementsByTagName("form")(0).submit  
submits the first form on the page.

And we use **Methods** with the **InternetExplorer** object like these:

Assume we have  
Dim objIE As InternetExplorer  
Set objIE = New InternetExplorer

Then…

objIE.navigate "http://google.com"  
navigates the browswer

objIE.stop  
manually stops browser navigation.

objIE.refresh  
performs a browser refresh.

objIE.quit  
closes the IE browser.

But the real web-scraping fun happens when using the following *powerhouse* **Methods** with **InternetExplorer** to hunt down (sometimes-illusive) web page elements. These VBA/IE **Methods** are powerful web-scraping allies that don’t just *walk the DOM*, they stomp all over it! …

getElementById   [video with example use](http://automatetheweb.net/)  
getElementsByName  
getElementsByTagName   [video with example use](http://automatetheweb.net/vba-getelementsbytagname-method/)  
getElementsByClassName   [video with example use](http://automatetheweb.net/reddit-how-to-extract-data-from-html-into-excel/)

And to work alongside those go-**get**ters, we employ these sophisticated, relationship-bound DOM-walkers:

Children  
firstElementChild  
nextElementChild  
lastElementChild  
nextElementSibling  
lastElementSibling  
previousElementSibling  
parentElement

These *family-oriented* methods rely on the predictable, cohesive and hierarchical structure of the DOM to pinpoint and hook onto any element(s) on a web page.

**Chaining multiple Properties and Methods to walk the DOM**

So what is the DOM? Think of the [**Document Object Model**](http://stackoverflow.com/questions/4976344/what-is-dom-summary-and-importance) as a complete and perfect snapshot of the entire loaded web page, stored in VBA as a tree-like structure. A page’s DOM is completely known and understood by VBA once a page has been successfully loaded. Study the image below, replacing the word *jQuery* with *VBA* …

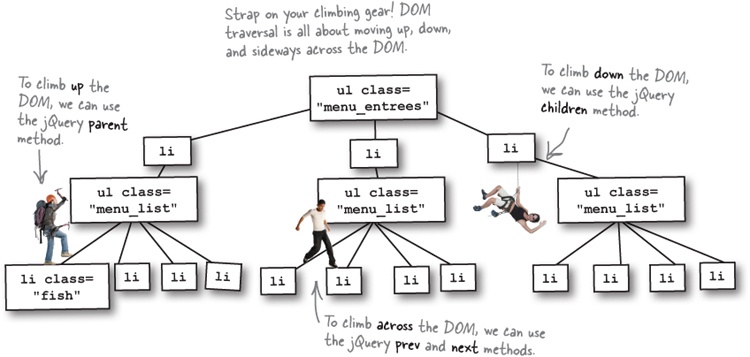
VBA and the Document Object Model of a web page

image source: [this awesome site](https://www.safaribooksonline.com/library/view/head-first-jquery/9781449311988/ch04.html)

In order to scrape web page data, we just need to direct VBA how and where within the DOM to access it. This is sometimes easier said then done, and questions with *"how do I get to this data…"* are the most common ones I hear. But accessing and interacting with a specific element or elements on a web page with VBA is always possible, and becomes increasingly easy to do with practice and learning to read the DOM (see video links above).

**The most common technique when using VBA to find and interact with web page elements is using a method I call *chaining*** (well, I probably read it somewhere) where we employ a combination of one or more of the afformentioned ***get****ters* with one or more of the *family-oriented* methods.

**Chaining VBA Properties and Methods – example 1**

Considering the image above, let’s figure out how to get the text within the li element with the class name "fish". It’s important to note there are many ways to get there, just as there are different ways to climb a tree and still get to the top. Some of these ways are more efficient than others.

Here’s the most direct way that I see: objIE.document.getElementsByClassName("fish")(0).innerText.

And that route would work even if all those bottom-row li‘s had the class "fish", since *(0)* represents the first one.

But what if that li had no class? Then we could use: objIE.document.getElementsByClassName("menu\_list")(0).Children(0).innerText

Here you can see we’ve used dots (.) to **chain** together the innerText property of the Children method of the getElementsByClassName method of the document method of the InternetExplorer object.

**Chaining VBA Properties and Methods – example 2**

Let’s consider another example, using a previous code sample, of chaining/connecting together multiple **InternetExplorer** and **HTML** methods and properties.

|  |
| --- |
| <**div** id="hdimg" class="header-image">  <**a** href="http://automatetheweb.net/" rel="home">  <**img** src="http://automatetheweb.net/wp-content/uploads/login-logo.jpg" width="350" height="40" alt="Automate the Web">  <**div**><**span** style="font-size: 90%;">a div of text I added, just for giggles</**span**></**div**>  </**a**>  </**div**> |

Let’s assume  
Dim objIE As InternetExplorer  
Dim ele as Object  
Set objIE = New InternetExplorer

Now let’s come up with a variety of ways to scrape out the line of text with the word "giggles" and print it in the debug console. Each of the following VBA examples prints *"a div of text I added, just for giggles"*.

* Debug.Print objIE.document.getElementById("hdimg").innerText
* Debug.Print objIE.document.getElementsByClassName("header-image")(0).innerText
* Debug.Print objIE.document.getElementsByClassName("header-image")(0).getElementsByTagName("span")(0).innerText
* Debug.Print objIE.document.getElementsByClassName("header-image")(0).getElementsByTagName("img")(0).nextElementSibling.innerText
* Debug.Print objIE.document.getElementsByClassName("header-image")(0).getElementsByTagName("img")(0).nextElementSibling.Children(0).innerText
* Debug.Print objIE.document.getElementsByClassName("header-image")(0).getElementsByTagName("img")(0).nextElementSibling.getElementsByTagName("span").innerText
* For each ele in objIE.document.getElementsByTagName("span")  
     If instr(ele.style, "ize: 9") > 0 then  
        Debug.print ele.innerText: Exit For  
     End If  
  Next

While all of these VBA code examples accomplish the same thing, clearly the first is the most obvious and practical, while the remainder are merely illustrative and increasingly obscure (even a little ridiculous).