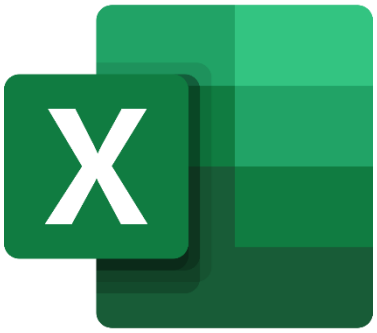
Web scraping en VBA



Contents

Introduction

Comme toute programmation de macro, la majeure partie du temps doit être consacrée à la réflexion sur l'algorithme. L'idée est toujours la même ; récupérer toutes les informations souhaitées en chargeant le minimum de pages internet.

Puisqu'il s'agit de pages internet, il faut connaître l'arborescence du site internet. Il existe presque toujours plusieurs chemins permettant d'accéder à une même information, cette étape est donc probablement la plus importante.

Il existe deux principales techniques permettant à VBA de récupérer du contenu internet :

Instance Internet Explorer

Il s'agit de générer des instances d'Internet Explorer depuis VBA. Le processus est similaire à une navigation "manuelle", nous lançons une page internet et naviguons dessus depuis Excel.

Avantages	Inconvénients
Possibilité de cliquer sur des éléments / d'interagir avec la page internet	Certains sites ne sont pas compatibles Internet Explorer
Possibilité de simuler une navigation "humaine" et déjouer les bots	Il faut toujours attendre que la page ait intégralement chargé pour commencer à exploiter son code.

Instance XML

La deuxième consiste en une instance XML qui va nous permettre de récupérer et exploiter le code HTML de la page. Il s'agit donc de récupérer le code HTML d'une page, et de l'analyser ensuite depuis Excel

Avantages	Inconvénients
Rapidité d'exécution car nous n'avons (presque) pas besoin d'attendre le chargement de la page	Peut être considéré comme une attaque DDOS par certains sites.
Permet parfois de contourner les sites ne permettant pas d'afficher le contenu sous Internet Explorer	Ne permet pas d'interagir directement avec la page web (i.e. cliquer sur un élément, remplir une barre de recherche…)

Le choix de l'une ou l'autre des techniques et parfois imposé, mais lorsque cela n'est pas le cas nous pourrons choisir la technique correspondant le mieux à notre besoin. La bonne nouvelle ; ces deux techniques diffèrent jusqu'à l'obtention du code HTML. Le reste de la macro est généralement identique pour ces deux approches.

Scraping

RECUPERATION DU CODE HTML

Instance Internet Explorer

Cette technique nécessite d'activer 2 références :

- **Microsoft Internet Controls**
- **Microsoft HTML Object Library**

Nous partons du principe que nous utiliserons souvent cette initialisation, nous allons donc créer une procédure dédiée. Pour être plus précis, nous allons créer 2 procédures, l'une appelant l'autre. La première est chargée de créer une instance internet et de faire une requête. La deuxième est chargée d'attendre que la page ait chargé et de renvoyer son code HTML.

Pourquoi 2 procédures ? Parce que nous pouvons être amené à attendre le chargement d'une page web sans pour autant avoir à créer une nouvelle instance.

Nous choisissons ici 2 variables public qui seront donc accessible dans l'ensemble du document Excel.

```
Public IE As InternetExplorer 'Variable pour créer l'instance Internet Explorer
Public webPage As HTMLDocument 'Variable du code HTML de l'instance


'/////////////////////////////////////
Sub browseIE(myURL As String, isVisible As Boolean)

    Set IE = New InternetExplorer 'Création d'une nouvelle instance "Internet Explorer"

    With IE
        .Visible = isVisible 'si "true" le navigateur est visible, sinon il est masqué
        .navigate(myURL) 'Navigation vers l'URL choisie
    End With

    Call loadingIE() 'On appelle une procédure qui attend le chargement de la page

End Sub


'/////////////////////////////////////
Sub loadingIE()

    'Tant qu'Internet n'a pas renvoyé un signal "complete", on attend
    While IE.readyState <> READYSTATE_COMPLETE
        DoEvents
    Wend
    'Tant qu'internet est occupé, on attend
    While IE.Busy
        DoEvents
    Wend
    Set webPage = IE.document 'On affecte le code HTML a notre variable

End Sub
```

Nous avons ainsi créé une procédure qui va ouvrir une page internet et nous donner accès au code HTML. Cette procédure prend 2 paramètres :

- myURL ; quel lien charger ?
- isVisible ; faut-il afficher le navigateur ?

Par la suite, nous pouvons appeler facilement cette procédure depuis n'importe quel fonction ou procédure avec une simple ligne de code :

```
Call browseIE("amazon.fr", False)
```

Instance XML

Pour cette technique, les références à ajouter sont :

- **Microsoft HTML Object Library**
- **Microsoft XML, vX.X**

Dans ce cas, nous ne créons qu'une seule variable publique "webPage" :

```
Public webPage As HTMLDocument 'Variable du code HTML de l'instance

Public Sub myXML(mySearch As String)

    Dim xmlPage As MSXML2.XMLHTTP60 'Declaration variable XML
    Set xmlPage = New MSXML2.XMLHTTP60 'Creation d'une nouvelle instance XML
    xmlPage.Open "GET", mySearch, False 'Requete XML ; "False" indique que nous attendons la fin
     du chargement avant la reprise
    xmlPage.send 'Procédure classique, les méthodes "Open " et "Send" sont complémentaires
    webPage.body.innerHTML = xmlPage.responseText 'la variable webPage correspond au texte de
     notre requête

End Sub
```

Contrairement à la procédure générant une instance Internet Explorer, il est indispensable de renseigner l'URL complète ("https://www.blablabla.com"). Cette procédure s'appelle ensuite très simplement :

```
Call myXML("https://www.amazon.fr")
```

RECHERCHE D'UN ENDROIT PRECIS DANS LA PAGE

Pour cette partie, les deux techniques sont identiques. Excel sera cependant plus flexible via l'instance Internet Explorer par sa connexion directe avec le site, ce qui permettra de "bricoler" plus facilement que via la requête HTTP.

Les variables

Dans l'immense majorité des cas, seulement 2 types de variables sont nécessaires :

- IHTMLElement -> correspond à un élément unique dans le code
- IHTMLElementCollection -> correspond à un élément multiple dans le code (i.e. les liens hyperlinks)

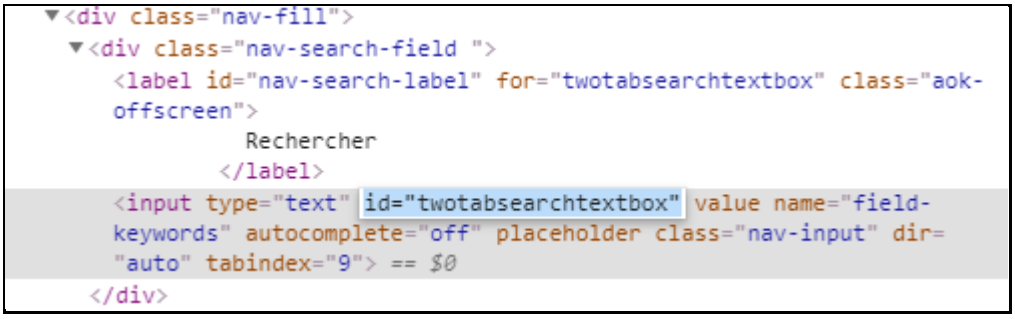
Il existe une multitude de déclaration possible, mais l'utilisation de ces 2 permet de tout faire. (i.e. IHTMLAnchorElement correspond à un lien hyperlink, mais est avant tout un élément HTML. Nous pouvons facilement le remplacer par ITHMLElement et pourrons toujours accéder à ses propriétés (mais sans IntelliSense))

Attention, il s'agit toujours d'objets, il faudra donc toujours utiliser "Set" pour affecter une variable à l'un ou l'autre de ces objets.

Les éléments du code HTML

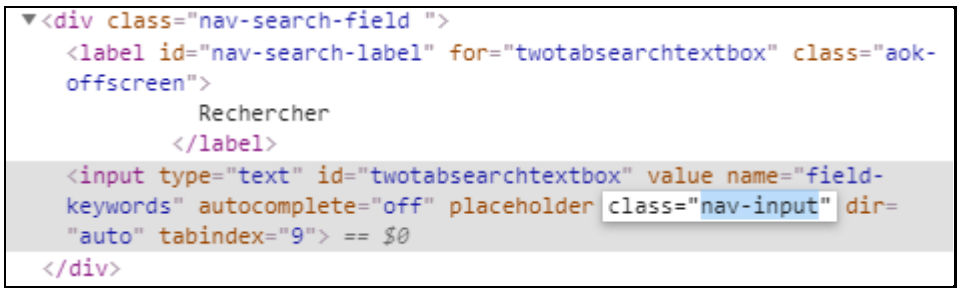
Les méthodes utilisées pour les instances Internet ou pour les requêtes sont également peu nombreuses :

- getElementByID -> correspond à l'ID d'un élément web, et est donc forcément affecté à une variable "Element" (et non "ElementCollection")



- getElementsByTagName -> correspond au TagName (i.e. "a", "div", , "input", "body", "table"...) de la page internet

- getElementsByClassName -> correspond au ClassName(i.e. “searchBar”, “nav-input”...)



Pour ces deux éléments getElementsByTagName ou getElementsByClassName, il s’agit par défaut de **collection** : le code va récupérer toutes les variables de cette collection, on est alors dans une variable “IHTMLElementCollection”.

On peut cependant choisir de n’accéder qu’à un élément de cette collection, on va alors utiliser son **index**. Dans ce cas, il s’agit d’une variable “IHTMLElement ”.

Par exemple :

```
'première URL de la page
Dim myLink As IHTMLElement
Set myLink = webPage.getElementsByTagName("a")(0) 'première URL de la page

'dernière URL de la page
Dim myLink As IHTMLElement
Set myLink = webPage.getElementsByTagName("a")(webpage.getElementsByTagName("a").length -1)

'toutes les URL de la page
Dim myLinks As IHTMLElementCollection
Set myLinks = webPage.getElementsByTagName("a")
```

Nous allons également introduire deux autres notions, moins utilisées mais qui peuvent être très pratique lorsque nous avons des difficultés pour atteindre un élément HTML précis de la page web : Children et ParentElement. Ceux-ci sont plutôt explicites :

- “Children” nous permet d’atteindre un élément HTML depuis son bloc “parent”
- “ParentElement” nous permet d’atteindre un élément HTML depuis son bloc “enfant”.

Ces instructions sont particulièrement utiles lorsque nous savons que l'élément HTML que nous cherchons est juste avant ou juste après un bloc HTML ayant un ID. On accèdera alors très facilement à cet élément, même sans connaître son className ou autre.

Par exemple :

```
Dim maVariable As IHTMLElement
Set myVariable = webPage.getElementById("myID").Children
```

Les URL

Comme mentionné plus haut, une URL sera accessible via IHTMLAnchorElement. Il existe trois principales méthodes/propriétés que nous appliquons à cet objet ;

La méthode « click »

```
Dim myLink As IHTMLElement
Set myLink = xxx
myLink.Click
```

La propriété « innerText »

```
Dim myLinkTitle As String
Dim myLink As IHTMLElement
Set myLink = xxx
myLinkTitle = myLink.innerText
```

Note: il existe également la propriété xxx.innerHTML qui permet de récupérer le code HTML d'un élément précis. Cette propriété est accessible pour tous les IHTMLElement, donc par extension tous les IHTMLAnchorElement

La propriété « href »

```
Dim myLinkUrl As String
Dim myLink As IHTMLElement
Set myLink = xxx
myLinkUrl = myLink.href
```

Exemples

Récupération de la barre de recherche sur Amazon :

```
Sub amazon1()

     Dim mySearchBar As IHTMLElement
     Call browseIE("amazon.fr", False)

     Set mySearchBar = webPage.getElementById("twotabsearchtextbox")
     While mySearchBar Is Nothing
         Application.Wait(Now() + TimeValue("00:00:01"))
         Set mySearchBar = webPage.getElementById("twotabsearchtextbox")
     Wend

     With mySearchBar
         'Code here
     End With

End Sub
```

Note sur la boucle "*While mySearchBar is nothing*". Sans cette boucle, "mySearchBar" ne prend régulièrement aucune valeur. Cette erreur est fréquente pour la première affectation d'un objet après avoir récupéré la variable HTMLDocument "webPage".

Récupération de tous les liens sur la page Amazon :

```
Sub amazon1_1()

     Dim myLinks As IHTMLElementCollection
     Dim myLink As IHTMLElement

     Call browseIE("amazon.fr", False)

     Set myLinks = webPage.getElementBytagname("a")
     While myLinks Is Nothing
         Application.Wait(Now() + TimeValue("00:00:01"))
         Set myLinks = webPage.getElementBytagname("a")
     Wend

     For Each myLink In myLinks
         'Code here
     Next myLink

End Sub
```

INTERACTIONS AVEC INTERNET EXPLORER

Ce paragraphe n'est applicable qu'à la technique passant par une instance d'Internet Explorer. Pour rappel, l'autre technique absorbe le contenu HTML d'une page et le renvoi dans une variable donc sans interaction possible.

Click & Value

Reprenons notre exemple "searchBar Amazon" : nous allons chercher l'emplacement de la "searchBar", y placer une valeur et lancer la recherche

```
Sub amazon2()

     Dim mySearchBar As IHTMLElement
     Dim myRequest As String
     Dim myLinks As IHTMLElementCollection

     myRequest = "apple iPhone XR"
     Call browseIE("amazon.fr", True)
     Set mySearchBar = webPage.getElementById("twotabsearchtextbox")
     While mySearchBar Is Nothing
         Application.Wait(Now() + TimeValue("00:00:01"))
         Set mySearchBar = webPage.getElementById("twotabsearchtextbox")
     Wend
     mySearchBar.Value = myRequest

     Set myButton = webPage.getElementsByClassName("nav-input")(0)
     myButton.Click
     Call loadingIE

End Sub
```

Pour la valeur de la searchBox, nous utilisons simplement la méthode ".value"

L'instruction "click" d'un élément HTML permet de cliquer sur le dit objet. En cliquant sur l'objet, nous avons chargé une nouvelle page web. Il faut donc à nouveau attendre le chargement intégral avant de continuer.

Seulement, un rapide coup d'œil à l'URL permet de voir que nous aurions pu directement atteindre cette page... Le code ci-dessous obtient le même résultat, mais bien plus rapidement... Il est également plus lisible et plus simple à entretenir.

```
Sub amazon3()

     Dim myRequest As String
     myRequest = "apple iPhone XR"
     Call browseIE("amazon.fr/s?k=" & Replace(myRequest, " ", "+"), False)

 End Sub
```

Javascript :

La majorité des pages web utilisent des scripts Javascript, que ce soit pour des animations, des autocomplétions, etc…Dans certains cas, il est indispensable de déclencher un script JS de la page directement depuis Visual Basic.

Technique encore inconnue au moment de l'écriture, impossible à couvrir pour l'instant.

SendKeys :

Dans certains cas, il est tout à fait possible de simuler l'appui d'une touche du clavier : il s'agit des instructions SendKeys. L'instruction permettant d'appuyer sur la touche "Entrée" est SendKeys ("{ENTER}")

RENVOI DES DONNEES DANS EXCEL

Dans l'hypothèse où nous ne voudrions récupérer que quelques éléments isolés, il n'y a pas vraiment de question à se poser. Dans le cas de large volume, il est souvent efficace de passer par des tableaux multidimensionnels dynamiques.

Nous reprenons notre cas Amazon, et allons récupérer toutes les URL de notre page de recherche. Il faut donc identifier tous ces liens, et venir alimenter un tableau "Array" qui contiendra 2 colonnes : le nom du lien et son URL

```
Sub amazon4()

    Dim myRequest As String
    Dim myArr() As String
    Dim myLinks As IHTMLElementCollection
    Dim myLink As IHTMLElement
    Dim myCount As Integer

    'Access website
    myRequest = "apple iPhone XR"
    Call browseIE("https://www.amazon.fr/s?k=" & Replace(myRequest, " ", "+"), True)

    'On récupère la liste des url de la page
    Set myLinks = webPage.getElementsByTagName("a")
    While myLinks Is Nothing
        Application.Wait(Now() + TimeValue("00:00:01"))
        Set myLinks = webPage.getElementsByTagName("a")
    Wend

    'On boucle sur chaque url
    For Each myLink In myLinks
        ReDim Preserve myArr(0 To 1, 0 To myCount) 'on redimensionne notre array
        myArr(0, myCount) = myLink.innerText 'texte du lien
        myArr(1, myCount) = myLink.href 'lien
        myCount = myCount + 1 'on incremente le compteur
    Next myLink

    'On renvoie notre tableau dans excel
    ThisWorkbook.ActiveSheet.Range("A1").Resize(myCount, 2) = Application.Transpose(myArr)

End Sub
```

Optimisations & Conseils

RAPPELS

ScreenUpdating : Désactiver / activer l'actualisation de l'écran pour accélérer l'exécution

```
Application.ScreenUpdating = False 'au début de la macro
Application.ScreenUpdating = True 'à la fin de la macro
```

Calculation : Les seuls calculs autorisés sont ceux demandés par l'utilisateur

```
Application.Calculation = xlCalculationManual 'au début de la macro
Application.Calculation = xlCalculationAutomatic 'à la fin de la macro
```

Boucles : préférer "For each" plutôt que "For i = xx to xx"

Select, activate, selection, activecell… sont à éviter autant que possible

Set object = Nothing : il faut ménager la mémoire de l'ordinateur. Il faut donc libérer les objets lorsque nous ne les utilisons plus, voire "manuellement" de manière régulière !

DoEvents cède le contrôle au système d'exploitation. Lorsque ce dernier a fini de traiter les événements de la file d'attente, le processeur reprend le contrôle.

CONSEILS

La propriété .length

Il arrive fréquemment que le seul moyen d'accéder à un élément spécifique soit de l'atteindre via son className ou son tagName, qui sont par défaut des collections. Cependant, ces className ou tagName sont souvent uniques.

Une technique classique consiste à déclarer en IHTMLElementCollection, de vérifier le nombre d'items. S'il n'y a qu'un seul item, on modifie alors la déclaration en IHTMLElement, et rajoutons l'index 0.

Par exemple :

```
Dim myTest As IHTMLElementCollection
Set myTest = xx
Debug.Print myTest.Length
```

Si Excel retourne la valeur 1, nous savons alors que cet élément est unique et pouvons alors changer notre déclaration :

```
Dim myTest As IHTMLElement
Set myTest = xx(0)
```

Option Explicit // Déclaration en Variant

Il peut arriver également de manipuler des types de données inhabituels et de ne pas savoir comment les déclarer. Dans ce cas, il existe 2 techniques

- **Déclarer l'objet en "variant"**, Excel affecte alors automatiquement un type de variable que l'on peut retrouver dans la fenêtre des variables locales.
- **Supprimer "Option Explicit"** en début de code, faire tourner la macro et vérifier dans la fenêtre des variables locales le type de variable qu'a utilisé Excel par défaut. Ne pas oublier de rajouter l'option lorsque l'information est trouvée.

Trim & LCase

La manipulation de chaînes de caractères a un rôle central dans le scraping ; la majeure partie des conditions reposent généralement sur ce type de variable. Seulement il arrive fréquemment de rencontrer des espaces en trop, ou d'avoir des majuscules mal placées. La combinaison de "trim" et "lcase" permet de simplifier ces manipulations.

```
'Plutot que d'écrire :
Dim myString As String
Dim myElement As IHTMLElement
Set myElement = xx
myString = myElement.innerText

'On préfèrera :
Dim myString As String
Dim myElement As IHTMLElement
Set myElement = xx
myString = LCase(Trim(myElement.innerText))
```

Chronomètre & compteur

Dans de nombreux cas, nous sommes confrontés à des limites, généralement le nombre de requête sur une période donnée. Il convient alors de chronométrer la macro et de compter ses requêtes.

```
'Exemple de Timer
Dim myStart As Single, timeElapsed As Single

myStart = Timer 'At the beginning
timeElapsed = Round(Timer - myStart, 2) 'Anytime you want
```

On incrémentera sa variable compteur à chaque requête, et avons ainsi nos deux conditions pour gérer les limites.

Selenium

Nous avons mentionné en introduction les limites de ces techniques en termes de navigateur et d'OS. Il existe cependant une librairie « Selenium » qui permet de contourner ces limites. https://codingislove.com/browser-automation-in-excel-selenium/

JSON

Il peut arriver de vouloir gérer des fichiers JSON, notamment dans le cas de requêtes API. Il existe également une librairie pour cet usage : https://github.com/VBA-tools/VBA-JSON