Interaction avec Internet Explorer via VBA Excel

Par Qwazerty

Date de publication : 31 août 2012

Dernière mise à jour : 9 septembre 2012

On se propose dans ce document, de prendre le contrôle d'une instance d'Internet Explorer et d'interagir avec celle-ci au travers d'un code VBA. Plusieurs méthodes et exemples seront développés afin d'y parvenir.

Pour toutes remarques, merci d'utiliser la discussion suivante .

I - Introduction

« Je suis enseignant, l'Académie nous fournit une application Web nous permettant de saisir les notes et les appréciations de mes élèves. Or il n'est pas possible de synthétiser les notes comme je le souhaite et de sortir des graphiques représentant l'évolution de mes classes. Je gère donc en parallèle un fichier Excel qui me permet de réaliser tout ceci. J'en ai assez de faire une double saisie et ainsi je voudrais à partir d'Excel aller chercher et saisir des renseignements directement sur le site Web. »

Suite à plusieurs questions sur le forum de DVP, voici un petit tutoriel, qui vous permettra, je l'espère, de vous connecter à Internet Explorer (IE dans la suite du document), via VBA et de manipuler les pages Web à votre guise. Je traiterai ici uniquement des pages basiques avec un code source apparent, les parties de code en Flash ne seront donc pas abordées.

Il ne s'agit pas ici de donner une méthode miracle qui fonctionne avec tout type de page, je vais vous donner des outils, qui vous permettront de vous repérer dans l'arborescence d'un site ainsi que dans celle des objets VBA qui lui sont liés. Bien sûr, il vous restera à fournir un gros travail de recherche afin de cibler les différents renseignements et les différentes actions que vous souhaitez effectuer sur la page et, en tenant compte de la structure même du code source, d'appliquer les bonnes méthodes pour y parvenir.

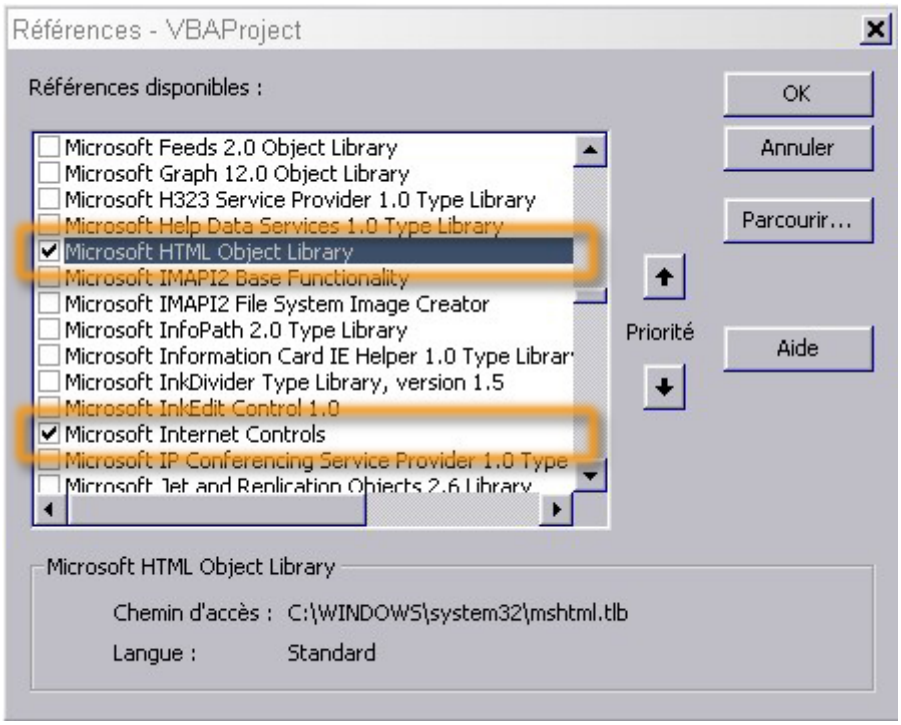
Vous vous direz sans doute parfois en lisant le document : « Mais pourquoi n'a-t-il pas tout simplement fait comme ci ou comme ça ?». Plusieurs raisons, déjà peut-être parce que le jour où j'ai écrit cette partie, eh bien je n'avais pas vu plus simple. Il y a des jours avec et des jours sans. La deuxième raison, c'est que ce document, comme je l'ai dit au-dessus, est fait pour vous montrer un maximum de méthodes différentes (mais non exhaustives), afin que vous puissiez y trouver votre bonheur, dans la majorité des cas.

II - Accès aux Objets d'Internet Explorer

II-A - Les références à activer

Comme souvent, il existe des références qui peuvent être activées au niveau de VBA et qui vous permettent d'accéder par la suite à l'aide (si elle existe), à l'autocomplétion du code et à l'arborescence des objets via l'Explorateur d'objets VBA.

Pour manipuler IE, il nous faudra activer deux références : « Microsoft Internet Controls » et « Microsoft HTML Object Library ». Pour accéder aux références dans VBA, menu *Outils -> Références.*



Si toutefois les deux références n'étaient pas présentes dans la liste, dans un premier temps, faites une recherche sur votre disque dur System et recherchez le chemin des deux fichiers suivants :

- Mshtml.tlb : correspond comme on peut le voir sur la capture d'écran à la référence « Microsoft HTML Object Library » ;
- IEFrame.Dll : qui représente bien sûr la référence « Microsoft HTML Object Library ».

Une fois les deux chemins notés, utilisez le bouton *Parcourir...* de la boite Références et ajoutez les deux fichiers.

*Pour les utilisateurs d'une version d'Internet Explorer inférieure à la version 7, le fichier **IEFrame.Dll** est à remplacer par le fichier **shdocvw.dll***

II-B - Déclaration des variables

Il existe deux méthodes de déclaration d'objets, le Late Binding et les Early Binding. Je vous les présente ici succinctement, si vous souhaitez plus d'information concernant ces méthodes, je vous invite à consulter le tutoriel de **Maxence Hubiche** sur **le Late et Early Binding**.

II-B-1 - Late Binding

Si vous choisissez de ne pas utiliser le référencement, voici le type de code que vous utiliserez

```
Sub PremierIE()
'Déclaration des variables
Dim IE As Object

   'Initialisation des variables
   Set IE = CreateObject("InternetExplorer.Application")

   'Suite du code
End Sub
```

Une telle utilisation ne vous permettra pas d'avoir l'autocomplétion, ni une aide concernant l'objet *IE*.

II-B-2 - Early Binding

Personnellement, je préfère cette méthode de déclaration. Certes elle impose certaines contraintes lors du déploiement de votre fichier sur d'autres postes, liées au référencement mais elle permet d'avoir accès à l'aide (F1), à l'Explorateur d'objets (F2) et permet d'utiliser l'autocomplétion du code.

Si vous utilisez le référencement voici le code que vous pouvez produire pour avoir accès à un objet *IE*

VBA
```
Sub PremierIE()
'Déclaration des variables
Dim IE As InternetExplorer

   'Initialisation des variables
   Set IE = CreateObject("InternetExplorer.Application")

   '[...] Suite du code
End Sub
```

Il existe une alternative, en déclarant IE avec ***As New*** au lieu de ***As***, il n'est alors plus nécessaire d'initialiser la variable ***IE*** avec ***CreateObject***.

VBA
```
Sub PremierIE()
'Déclaration des variables
Dim IE As New InternetExplorer

   '[...] Suite du code
End Sub
```

Comme je le disais plus haut, l'inconvénient du référencement apparaîtra au moment du déploiement de votre fichier sur un autre micro, si le poste n'a pas les fichiers ou si les versions sont différentes, VBA ne reconnaîtra pas le vocabulaire spécifique, tel que, par exemple, ***InternetExplorer***.

II-C - Lancer une navigation

Dans la suite de ce document j'utiliserai le référencement, l'écriture du code s'en trouve simplifiée et je pense que s'agissant des référencements Internet Explorer, il y a peu de risques que ceux-ci ne soient pas présents sur un poste. De plus il ne figure pas de numéro de version dans le titre des références, pouvant varier d'un poste à l'autre.

Une petite mise en pratique, avec le lancement d'une page Google, la fermeture de la page restant ici à la charge de l'utilisateur.

VBA
```vba
Sub PremierIE()
'Déclaration des variables
Dim IE As New InternetExplorer

   'Chargement d'une page web Google
   IE.Navigate "www.google.fr"

   'Affichage de la fenêtre IE
   IE.Visible = True

   'On libère la variable IE
   Set IE = Nothing

End Sub
```

III - Obtenir le code source d'une page HTML

III-A - Vocabulaire

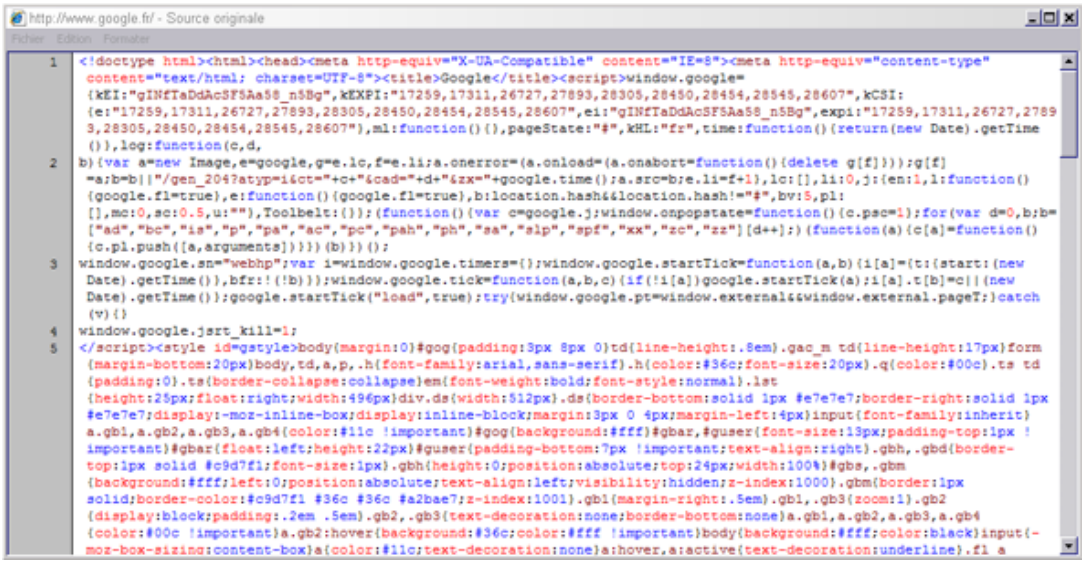
N'étant pas un développeur orienté Web, techniquement je ne suis d'ailleurs pas développeur, je préfère vous rediriger sur le tutoriel de **Josselin Willette** traitant du **code HTML**, vous y trouverez les informations nécessaires à la compréhension de l'organisation d'une page Web.

III-B - Retrouver les zones intéressantes sur une page

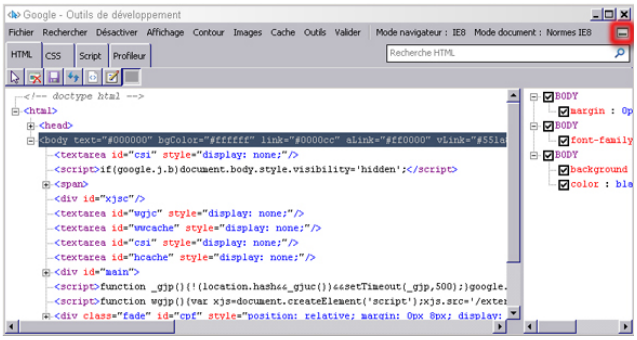
Pour interagir avec les éléments d'une page Web, il va bien falloir les trouver sur cette page et pour les trouver il faut les caractériser en trouvant un moyen de les identifier. Pour ce faire, voyons comment accéder au code qui compose la page.

III-B-1 - Afficher le code Source

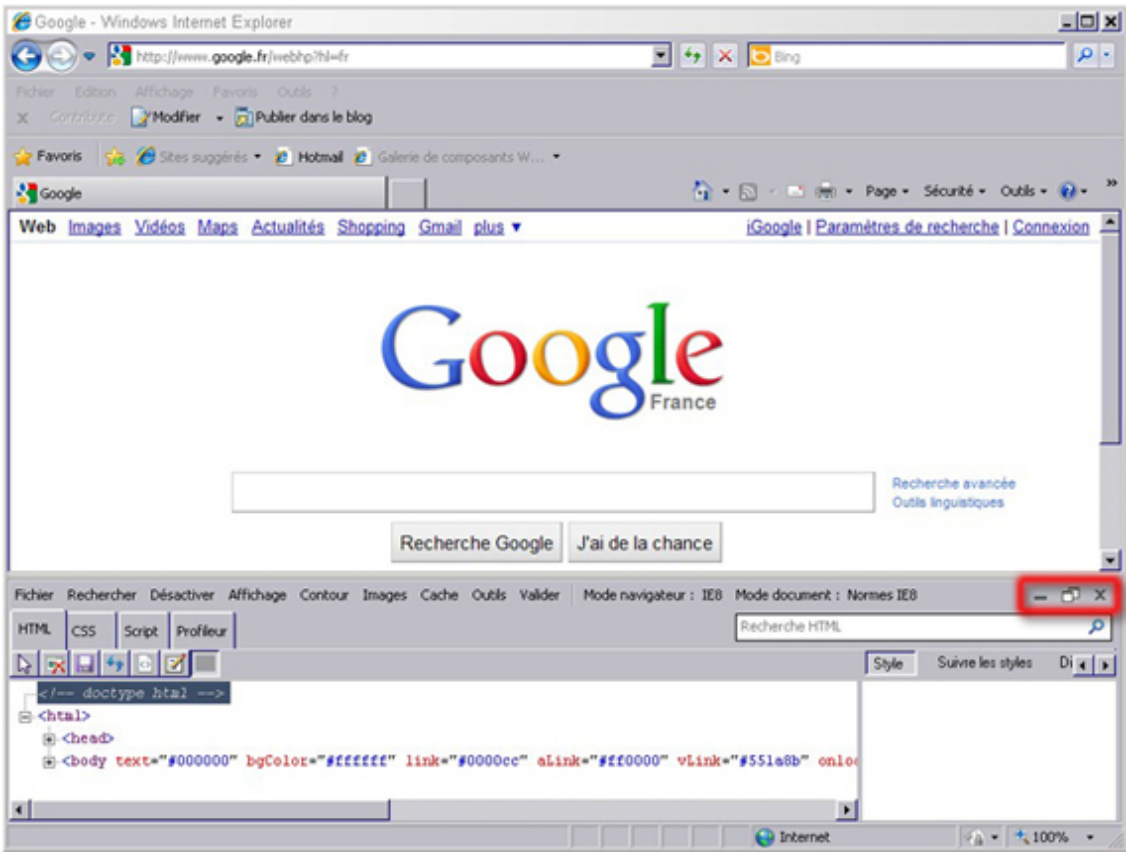
À ma connaissance, tous les explorateurs internet permettent de visualiser le code source d'une page Web, si nous prenons IE, une fois la page chargée, aller dans le menu *Affichage -> Source*.



Tel quel, le code n'est pas très lisible, toujours à partir d'IE, il existe une alternative, ce sont les outils de développement, ils sont accessibles via le menu IE *Outils -> Outils de développement* (raccourci clavier F12), vous avez ainsi accès à un code organisé et hiérarchisé.



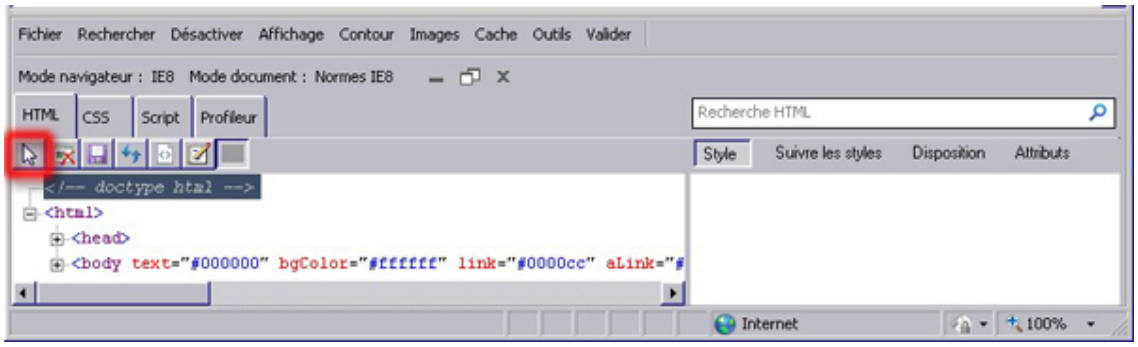
Afin de rendre son utilisation plus conviviale, il vous est possible de l'intégrer à la fenêtre de votre navigateur via le bouton **Attacher** (raccourci clavier Ctrl +P).



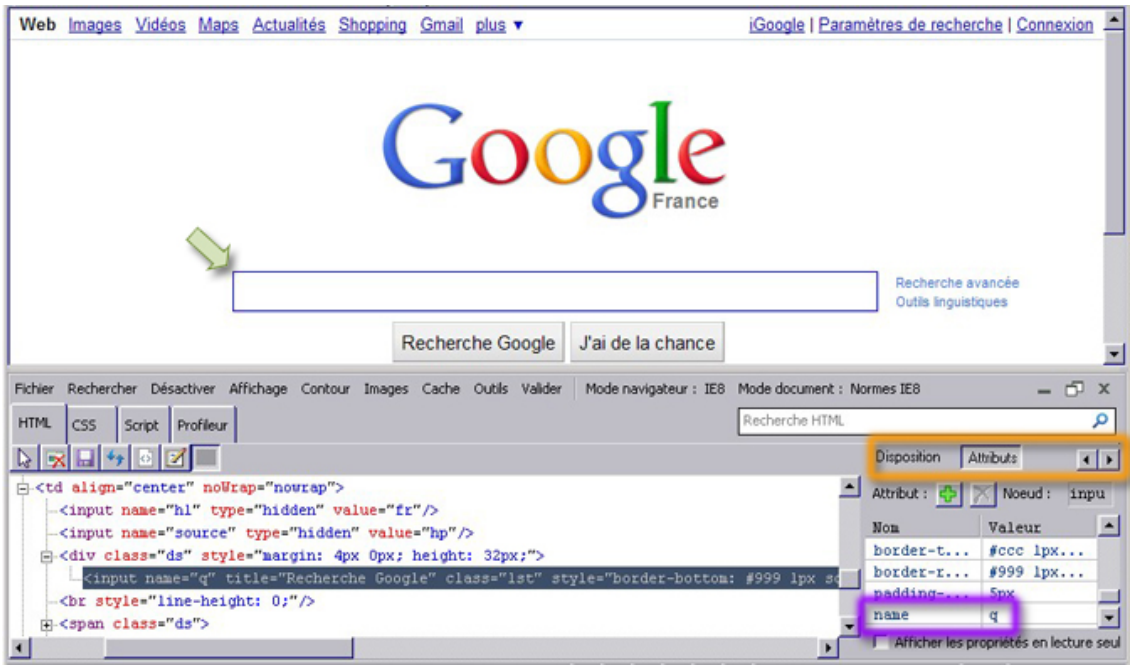
Vous pourrez à tout moment réduire, détacher ou fermer la fenêtre outil via **les boutons classiques**.

Dans la suite du document, je ne présenterai que les éléments dont nous aurons besoin dans le cadre des différents exemples. Vous trouverez plus amples informations sur l'utilisation de *Outils de développement* **dans la base de connaissances Microsoft**.

III-B-2 - Cibler un élément



Bien me direz-vous, mais comment trouver dans tout ce code l'élément qui m'intéresse sur ma page. Pour ce faire nous allons utiliser le **bouton flèche**.



Une fois activé, déplacez la souris sur la page Web, un cadre bleu devrait apparaître ciblant la zone que vous pointez avec votre souris. Une fois l'élément tant convoité **encadré de bleu**, un clic sélectionne la ligne de code correspondant à l'élément, ici la zone de texte. De la même manière, si vous cliquez sur une ligne de code dans la partie basse, la zone correspondante est sélectionnée en bleu sur la page Web.

Au niveau de l'outil de développement, sur la partie de droite, **plusieurs boutons** permettent d'afficher des caractéristiques des éléments, pour la suite de ce document, nous utiliserons par défaut l'affichage des Attributs. Dans cette capture d'écran, nous avons donc le code correspondant à la zone de texte qui est sélectionnée. Dans la partie *Attributs* à droite, on peut voir les différentes propriétés qui caractérisent l'élément, on s'intéressera ici à sa propriété **name**, qui nous apprend que cette zone de texte a pour nom « q », nous verrons par la suite comment utiliser cette information.

Pour ceux qui comme moi, utilisent Google Chrome, il est possible de faire des choses similaires avec la fonction Inspecter l'objet, disponible en faisant un clic droit sur un des éléments de la page, ou raccourci clavier F12.

III-C - Correspondance avec l'arborescence VBA

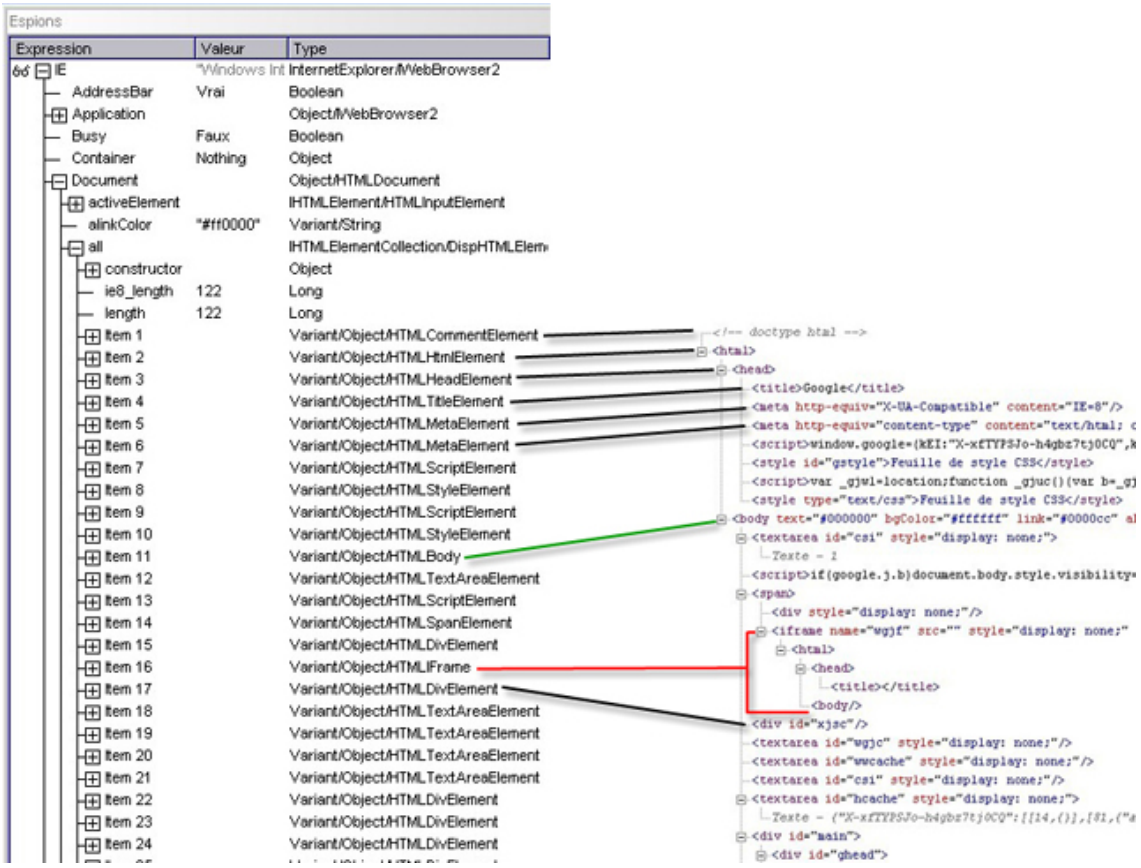
Le code Source de la page est constitué de balises, il serait donc intéressant, de retrouver nos différents éléments représentés par ces balises, dans l'objet ***IE*** que nous avons déclaré plus haut. Pour ce faire, nous allons placer un point d'arrêt, sur une ligne de code et créer un espion VBA sur l'objet ***IE***.

Les espions étant, à tort, peu utilisés par les débutants, vous trouverez en annexe, un petit mode d'emploi de ceux-ci ainsi qu'une information sur les points d'arrêt.

III-C-1 - Comparaison de contenu

Nous allons déployer l'objet ***IE***, les différents membres liés à l'objet ***IE*** apparaissent, intéressons-nous plus particulièrement à ***Document***, que nous allons également déployer, puis l'objet ***all*** de type ***DispHTMLElementCollection***.

Nous allons maintenant comparer le contenu de notre page à celui contenu dans l'objet ***all***.



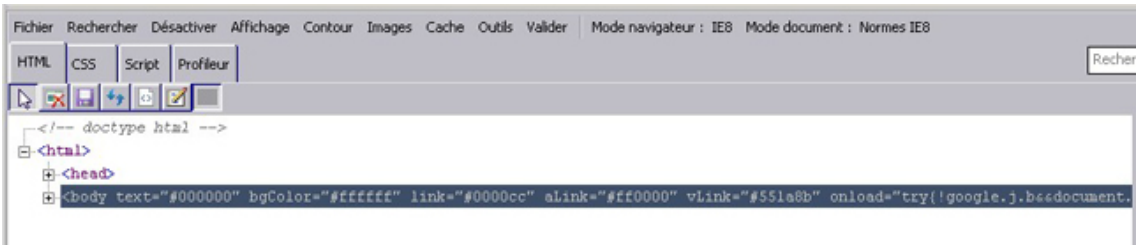
Je ne réalise pas toutes les correspondances, vous l'aurez compris, on retrouve chacun des éléments de notre code, dans l'arborescence de l'objet *all*, dans l'ordre où ceux-ci apparaissent dans le code.

On notera toutefois une exception pour l'élément **<iFrame**, dont la correspondance est en rouge. Le contenu de celui-ci, à savoir **<html>**, **<head>**, **<title>** et leurs balises de clôture respectives, n'apparaît pas dans l'objet ***all***. Le composant iFrame est utilisé pour afficher dans le corps d'une page une autre page Web indépendante. Cette page indépendante a beau être affichée, son contenu n'est pas accessible dans l'arborescence VBA, il faudra charger cette page dans une fenêtre de navigation pour interagir avec elle.

À noter également l'élément **<body**
, dont la correspondance est repérée en vert, et que je vais détailler par la suite.

III-C-2 - La partie Body

Nous allons remonter un peu dans notre code source, en réduisant un peu son arborescence et nous allons sélectionner, à l'aide du sélecteur d'élément, la ligne de code contenant **<body**



Lorsque la sélection est faite, vous remarquerez que la totalité de la page est sélectionnée, ce qui soit dit en passant, est conforme à la lecture du tutoriel sur les pages Html fourni en début de tutoriel et qui, page 8, nous apprend que la balise body encadre le code représentant le contenu de la page.

Regardons une nouvelle fois notre objet ***IE.Document***, on retrouve dans sa structure un objet ***body*** et si l'on compare de nouveau notre code source **<body>** par rapport au contenu de l'objet ***body.all***, on retrouve bien tous nos éléments. L'objet ***body*** sera donc intéressant lors de la recherche d'informations dans la partie objet sous VBA, afin de limiter la zone de recherche.

Encore une fois, je vous incite à utiliser l'Explorateur d'objet VBA (touche F2), celui-ci vous permet de découvrir les éléments contenus dans les objets ***IE***, regroupés dans la bibliothèque ***SHDocVw*** et ses sous-objets, tels que ***document***, ***body***, ***all***, ***links***, etc., qui sont, quant à eux, regroupés dans la bibliothèque MSHTML.

Vous pourrez également trouver la liste des méthodes et propriétés sur le site **Msdn**.

Puisqu'on parle de ça, regardons un peu plus en détail les méthodes suivantes de l'objet ***document***, ***all***, ***anchors***, ***body.all***, ***body.children***, ***forms***, ***images***, ***links*** et ***scripts***. Je vois déjà vos yeux pétillants de malice ! Ça ne vous a pas échappé hein, tous ont en effet un point commun, leur type, à savoir ***DispHtmlElementCollection***, gardez-le dans un coin de votre tête, ça nous servira plus tard. Il y a d'autres collections de ce type, mais je ne les utiliserai pas dans ce tutoriel.

IV - Interaction simple avec les éléments de la page

Maintenant que nous savons retrouver un élément de notre page, il va falloir jouer avec, afin d'en faire ce que l'on en veut. Nous allons donc automatiser une recherche sur le site de google.fr. Mais avant de pouvoir manipuler un élément dans la page, il faut être certain que celle-ci est bien chargée.

IV-A - Attente du chargement de la page

Lors du chargement d'une page internet, IE renvoie l'état de chargement de cette page. Cette information est contenue dans la propriété ***ReadyState***. Cette propriété apparaît dans chaque élément de la page, nous nous intéresserons pour ce tutoriel uniquement à celui contenu dans l'objet *IE*, qui détermine l'état de chargement de la page complète. Sachez qu'il existe une autre façon de faire, en passant par l'utilisation des événements déclenchés par IE, cette méthode est basée sur une déclaration de type

```
Dim Withevents IE as InternetExplorer
```

*Pour ceux qui souhaiteraient étudier cette approche, vous trouverez plus de renseignements dans le tutoriel d'**Arkham46**, plus précisément dans la section traitant des **événements liés au contrôle InternetExplorer**.*

Il existe cinq états pour cette propriété.

- READYSTATE_UNINITIALIZED = 0

À cet état, IE est lancé, mais aucune page n'a été demandée. Vous aurez ce retour avec ce type de code

VBA
```
Sub PremierIE()
'Déclaration des variables
Dim IE As New InternetExplorer

   'Affichage de la fenêtre IE
   IE.Visible = True
```

- READYSTATE_LOADING = 1

Ici, la page est en cours de chargement, IE est dans cet état suite à ce type de code

VBA
```
    'Chargement d'une page Web Google
   IE.Navigate "www.google.fr"
```

- READYSTATE_LOADED = 2

Je n'ai jamais vu pointer cet état, en interprétant son nom, j'en déduis qu'il est émis à la fin du chargement de la page, mais il doit être couvert par un des états suivants (je pense que c'est par le dernier des cinq).

- READYSTATE_INTERACTIVE = 3

Ici la page est partiellement chargée, elle ne nous apparaît pas forcément visuellement, il reste encore des éléments graphiques en cours de téléchargement. Par contre l'interaction avec les différents éléments fonctionne et nous pourrons commencer à renseigner des champs, exécuter des liens. Inutile de perdre du temps à attendre d'avoir une page visuellement aboutie donc. Toutefois, pour plus de sûreté, j'attendrais tout de même systématiquement l'état 4.

- READYSTATE_COMPLETE = 4

Et pour finir, la page est entièrement organisée, structurellement et graphiquement. Je pense que l'état LOADED est recouvert par celui-ci.

Afin de vous familiariser avec ces différents états, vous trouverez, dans le Source **fichier Excel source** fourni à la fin du document, un UserForm qui permet de surveiller l'état de votre page internet, vous pourrez ainsi suivre dans quel ordre apparaissent les états. Je vous conseille de vous balader sur différents sites, celui de Deezer.fr donne des résultats intéressants.

Vous y trouverez également une version plus aboutie de feuille d'attente. Pour « standardiser » le tutoriel, je n'utiliserai pas cette UserForm d'attente au profit du code passe-partout suivant :

VBA
```
Sub WaitIE(IE As InternetExplorer)
   'On boucle tant que la page n'est pas totalement chargée
   Do Until IE.ReadyState = READYSTATE_COMPLETE
      DoEvents
   Loop
End Sub
```

Ici je choisis d'attendre le chargement total de la page. On utilisera cette procédure de la manière suivante :

VBA
```
    ' [...] Initialisation des variables

   'Chargement d'une page Web Google
   IE.Navigate "www.google.fr"

   'Affichage de la fenêtre IE
   IE.Visible = True

   'On attend le chargement complet de la page
   WaitIE IE

   '[...] Suite du code
```

*J'en profite pour faire une petite remarque concernant la visibilité d'Internet Explorer. À la fin du code, je rends visible l'instance d'IE, si vous choisissez de ne pas le faire, veillez bien à clôturer cette instance dans votre code. Par exemple en appelant la méthode **Quit** de l'objet **IE**.*

VBA
```
    'Fermer l'instance d'IE
   IE.Quit
```

Pensez également que si lors d'un débogage, vous arrêtez l'exécution du code sans que cette fenêtre ne soit ou fermée ou affichée, il vous faudra passer par le gestionnaire de tâches Windows (Ctrl+Alt+Supp) pour fermer IE.

Dans tous les cas, un redémarrage de votre poste fermera toutes les sessions ouvertes, ce qui bien sûr, ne doit pas vous empêcher de gérer comme il se doit dans votre code, les instances d'IE.

IV-B - Les fonctions de recherche

Nous avons vu que l'arborescence de ***IE.document.all*** est assez volumineuse, il serait donc fastidieux, d'avoir à rechercher systématiquement notre élément de page, dans cette arborescence. L'objet ***document*** met donc à notre disposition trois fonctions de recherche. Mise à part la dernière, que je détaillerai dans un exemple un peu plus loin dans ce document, je n'utilise peu, voire pas du tout, ces fonctions mais toujours dans une optique de vous offrir le plus de cordes à votre arc, je les cite ici.

IV-B-1 - GetElementById

Je survole rapidement cette fonction, nous aurons l'occasion d'y revenir, sachez juste que certains éléments peuvent être affublés d'un **id**, voici un exemple, toujours sur le site Google :

Un petit tour sur le site **Msdn**, nous informe que ***GetElementById*** nous renvoie le **premier** élément ayant l'**id** demandé, le **premier**... donc l'**id** n'est pas forcement unique sur une page, faites attention.

Au cours de la réalisation de ce document un autre élément me pousse à faire remarquer l'inutilité de cette fonction de recherche, en effet nous verrons plus en avant, que le résultat d'une recherche faite avec cette fonction, afin de mettre la main sur un élément ayant un id particulier, peut, sous certaines conditions, nous retourner un objet ayant une id vide. Alors méfiance.

IV-B-2 - GetElementsByName

Nous avions découvert que notre zone de texte se nomme « q », cette fonction va nous retourner le ou les élément(s) de la page ayant pour attribut **name** « q » ainsi que tous les éléments ayant pour attribut **id** « q ».

Plusieurs éléments peuvent donc répondre au même **name** / **id**, la fonction va donc nous retourner un ensemble d'éléments correspondant au critère, d'où le « s » à ***GetElementsByName***.

Prêtons attention au type retourné par cette fonction, ***IHtmlElementCollection*** et faisons de même pour la méthode ***All*** de l'objet ***document***, on voit que cette dernière nous retourne le même type et renverra, de plus, la même chose que ***GetElementsByName***, voilà pourquoi je n'utiliserai pas cette fonction dans ce tutoriel.

IV-B-3 - GetElementsByTagName

Le Tag dans VBA c'est le nom de la balise utilisée dans le code Html, **div**, **a**, **body**, **td**, **input**, etc. Avec sa fonction ***GetElementsByTagName***, vous noterez le « s » à « Elements », ***document*** nous offre la possibilité de lister tous les

éléments appartenant à un même type de balise. Et cerise sur le gâteau, cette fonction nous renvoie une structure ***DispHtmlElementCollection***. Nous pourrons donc utiliser l'objet qu'elle nous retourne pour faire une recherche **name**/**id** à l'intérieur de la liste d'éléments qu'il contient.

Comme précisé au début, un exemple sera détaillé plus loin, je ne m'y attarde donc pas plus ici.

*Lorsque vous réalisez une recherche, privilégiez le plus possible les recherches via l'**id** lorsque celui-ci est disponible. Votre code sera plus facilement adaptable, en cas de basculement du site Web siège de la manipulation, à la norme w3c. Celle-ci préfère l'utilisation de l'**id** vous risquez fort d'avoir à retravailler votre code lorsque vous aurez basé vos recherches sur les **names**.*

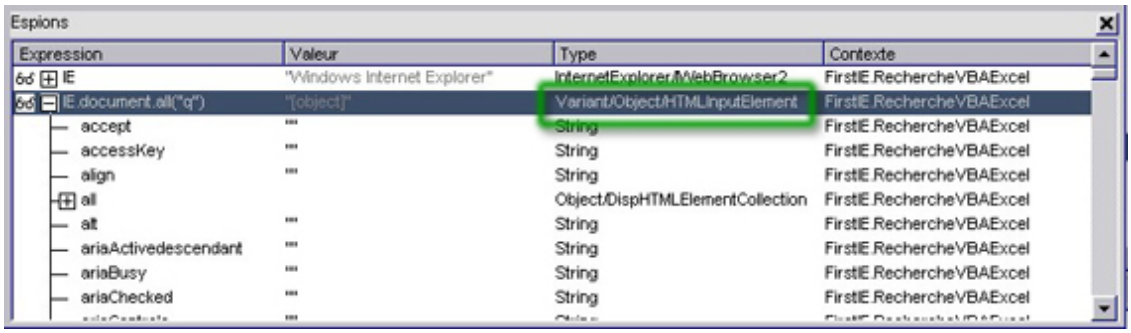
IV-C - Inscrire du texte dans une zone de texte

Maintenant que notre page est chargée et que nous avons nos outils de recherche, nous allons taper notre sujet sur Google, qui sera pour la démonstration « VBA Excel ».

Sur ce type de page, avec les éléments nommés « bien comme il faut », nous utiliserons les objets de type ***DispHtmlElementCollection***, je vous avais dit de le garder dans un coin de votre tête, tels que ***all***, ***body.all...*** et de lui passer en paramètre le nom de l'élément recherché (ou son id), ici j'ai pris soin de vérifier que l'élément « q » est unique et qu'aucun autre n'utilise « q » comme **name** ou comme **id**, nous aurons donc :

VBA
```
Sub RechercheVBAExcel()
'Déclaration des variables
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim InputGoogleZoneTexte As HTMLInputElement
Dim InputGoogleBouton As HTMLInputElement

   'Chargement d'une page Web Google
   IE.Navigate "www.google.fr"

   'Affichage de la fenêtre IE
   IE.Visible = True

   'On attend le chargement complet de la page
   WaitIE IE

   'On pointe le membre Document
   Set IEDoc = IE.document

   'On pointe notre Zone de texte
   Set InputGoogleZoneTexte = IEDoc.all("q")

   'On définit le texte que l'on souhaite placer à l'intérieur
   InputGoogleZoneTexte.Value = "VBA Excel"

   'On attend la fin de la recherche
   WaitIE IE

   'On libère les variables
   Set IE = Nothing
   Set IEDoc = Nothing
End Sub
```

Toujours afin d'avoir le maximum d'aide à la saisie du code, il est préférable de déclarer les éléments avec les bons types. Pour celles et ceux qui se demanderaient encore d'où je sors le type ***HTMLInputElement***, je leur rappelle l'utilisation de l'explorateur d'objets de VBA ainsi que des espions, qui nous permettent de déterminer un type avec précision.



On sait ainsi que notre élément peut être stocké dans un Variant, dans une variable ***Object*** et que son type natif est ***HTMLInputElement***. Nous remarquons au passage, je vous laisse le soin d'explorer les autres éléments, que le bouton sera lui aussi de ce type-là.

Bien, si tout s'est bien passé, notre texte est en place, il ne reste plus qu'à lancer la recherche !

IV-D - Agir sur un bouton

Rien de bien compliqué, le code parle de lui-même

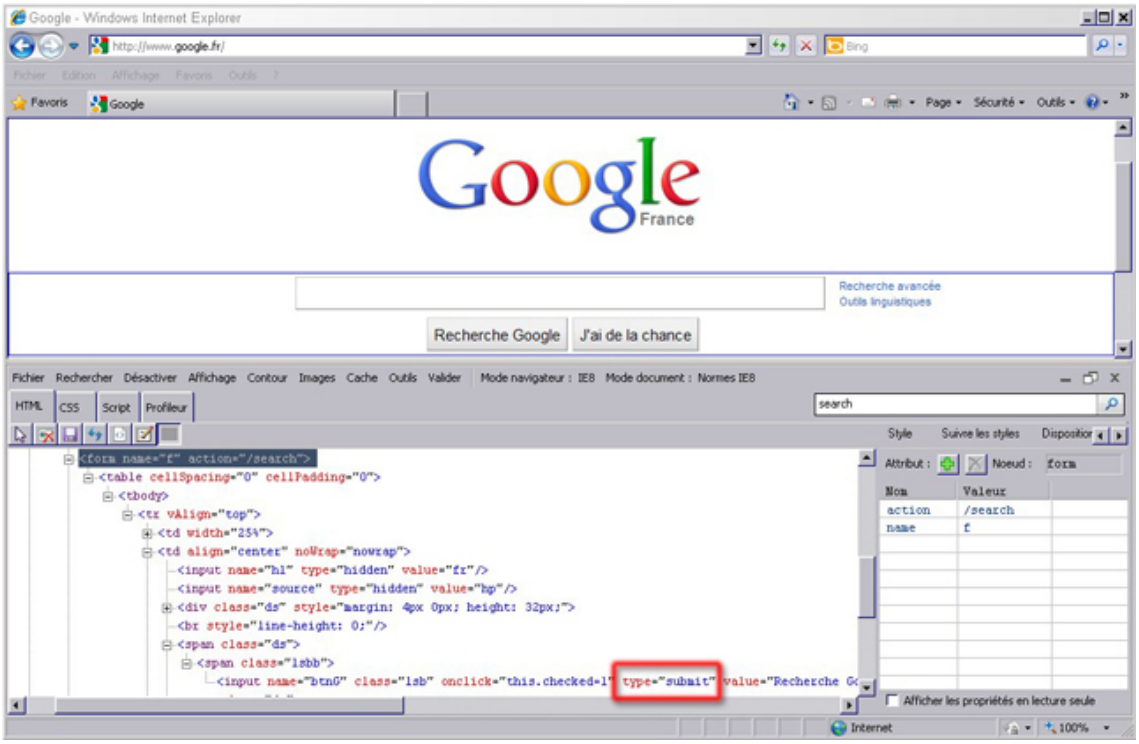
VBA
```
'On pointe notre Zone de texte
Set InputGoogleZoneTexte = IEDoc.all("q")

'On définit le texte que l'on souhaite placer à l'intérieur
InputGoogleZoneTexte.Value = "VBA Excel"

'On pointe notre bouton
Set InputGoogleBouton = IEDoc.all("btnG")

'On simule un clic
InputGoogleBouton.Click

'On attend la fin de la recherche
WaitIE IE

'[...] Suite du code
```

Une fois la recherche lancée par le ***Click***, on prendra soin d'attendre la fin de la recherche, via ***WaitIE***. Ici cela peut paraître inutile étant donné la rapidité de recherche, mais prenez l'habitude de toujours mettre une boucle d'attente après une action entraînant un chargement, afin d'être toujours sûr que les éléments sur lesquels vous travaillez sont bien disponibles.

IV-E - Petite variante

« Lorsque j'utilise le site de Google, moi, je ne clique jamais sur le bouton, je tape sur ma touche Entrée et hop ça lance la recherche » :

C'est vrai ça, pourquoi ne ferait-on pas la même chose, dans l'outil de développement d'IE, remontons un peu pour voir si un élément n'engloberait pas notre zone de texte et notre bouton.



On trouve effectivement un élément **<form**, on remarque également, que notre bouton est défini comme étant de type **submit**, nous allons donc utiliser ce référencement du bouton et utiliser la méthode ***submit*** de l'objet ***form***. Interrogation surprise : de quel type est le membre ***forms*** de l'objet VBA ***document*** ?

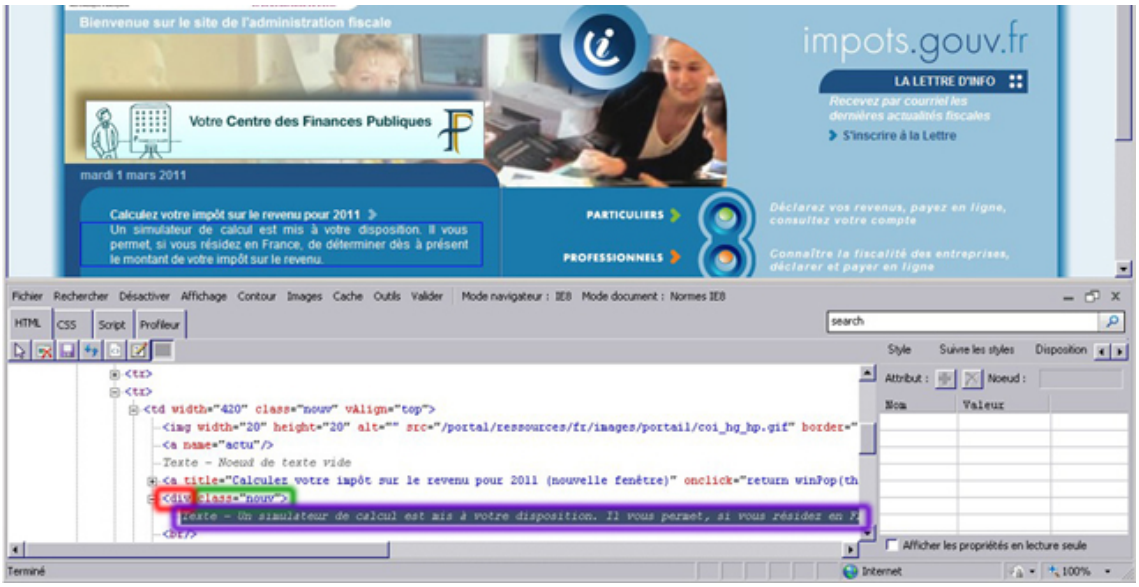
```vba
Dim FormGoogleCherche As HTMLFormElement
'On pointe notre Zone de texte
Set InputGoogleZoneTexte = IEDoc.all("q")

'On définit le texte que l'on souhaite placer à l'intérieur
InputGoogleZoneTexte.Value = "VBA Excel"

'On pointe la Form qui contient Zone de Texte + Bouton (entre autres)
Set FormGoogleCherche = IEDoc.forms("f")

'On exécute l'action Submit de la Form
FormGoogleCherche.submit

'On attend la fin de la recherche
WaitIE IE
```

DispHtmlElementCollection = 10/10, en effet ***forms*** fait partie de la liste et on peut donc l'utiliser pour faire notre recherche.

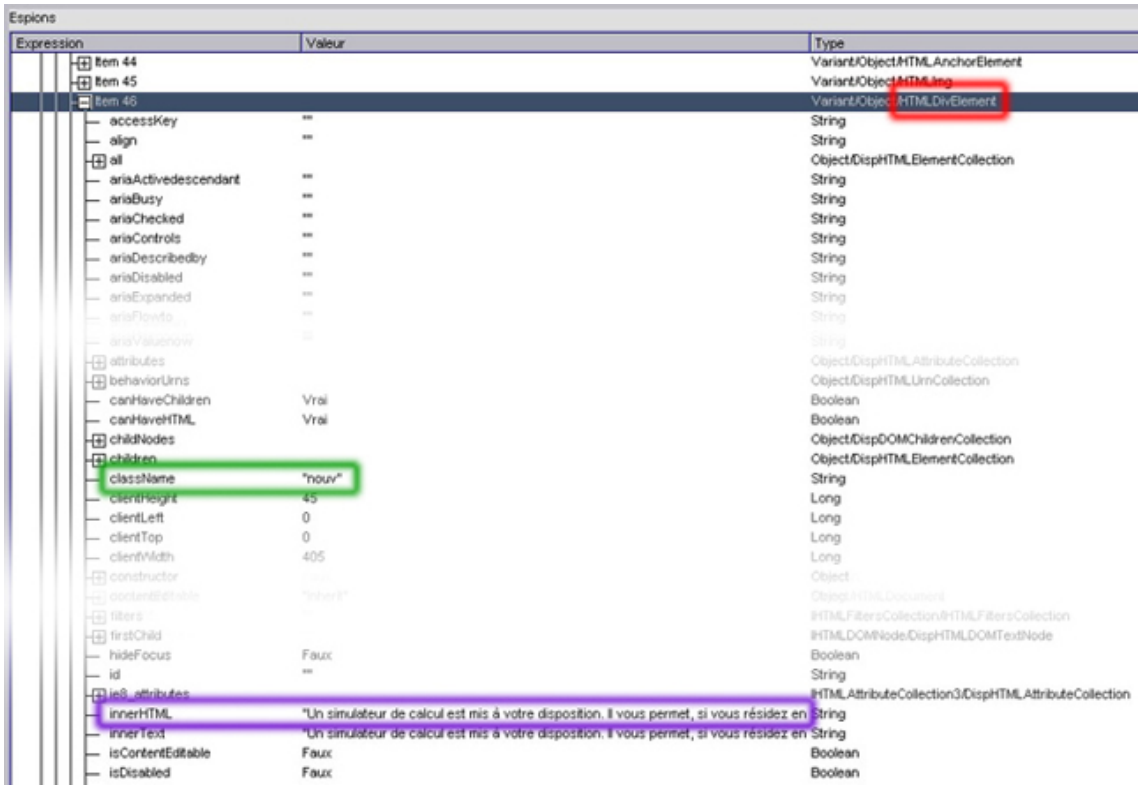
*Pour les utilisateurs de Chrome, les informations fournies sur l'élément **<form** sont plus précises et le lien avec la méthode **submit** y est clairement défini.*

IV-F - Récupérer du texte sur la page

Ici, nous allons faire un tour du côté de nos impôts, avec le site **http://impots.gouv.fr/**. Je souhaiterais copier un texte figurant sur la page



Impossible d'utiliser les mêmes méthodes vues précédemment, en effet le texte en question est dans un élément **<div** qui n'a pas de nom. Nous cherchons donc un élément de type ***HTMLDivElement***, ce qui va faciliter notre recherche dans l'arborescence de l'objet ***IE.document.body***. Après quelques recherches, on trouve le membre ***HTMLDivElement***, correspondant à l'élément **<div** convoité, au niveau de l'item n° 46.



On remarque au passage les différentes concordances permettant d'identifier l'élément, **ClassName « nouv »**, **InnerText** qui contient le texte en question et le type de l'élément **HTMLDivElement**.

Nous allons faire une utilisation que je définirais de plus « basique », en se basant sur l'arborescence de la page dans VBA. Je vous fournis le code qui ne contient pas de difficulté particulière, juste trois petites particularités ! Ne soyez pas surpris lors de l'exécution du code, la page prend un peu plus de temps pour charger que n'en prend la page de Google.

VBA
```
Sub RecupTextImpots()
'Récupère un texte brut contenu dans une page Web

'Déclaration des variables
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim HtmlElementStandard As HTMLGenericElement
Dim LeTexteExtrait As String

   'Chargement d'une page Web
   IE.Navigate "http://www.impots.gouv.fr/"

   'Affichage de la fenêtre IE
   IE.Visible = True

   'On attend le chargement complet de la page
   WaitIE IE

   'On pointe le document
   Set IEDoc = IE.document

   'Allons chercher ce texte dans notre Item46
   Set HtmlElementStandard = IEDoc.body.all(45)

   'On le place dans notre variable prévue à cet effet
   LeTexteExtrait = HtmlElementStandard.innerText

   'On affiche le texte
   MsgBox LeTexteExtrait, Title:="Le texte extrait de la page"

   'On libère les variables
   Set IE = Nothing
   Set IEDoc = Nothing
End Sub
```

Allons-y pour les particularités...

- « Pff, il s'est trompé il a mis *Item46* et dans le code *all(45)*. »

En fait dans l'arborescence affichée par l'espion VBA, les ***Items*** sont numérotés en partant de 1(de 1 à n éléments) et par contre lors de l'appel via ***all***, on commence à zéro (de 0 à n-1 éléments). Donc l'Item1 correspond bien à ***all(0)***. Autre détail important à connaître, lorsqu'un espion vous montre les items contenus dans une collection, il ne vous montre qu'un maximum de 256 éléments, les autres éléments sont présents dans la collection, mais ne seront pas affichés.

- « Bon OK, mais c'est quoi cette variable ***HTMLGenericElement***, pourquoi ne pas avoir mis le type réel ? »

Juste pour vous présenter un élément polyvalent, de cette manière s'il vous prenait l'envie de lister tous les éléments dans une boucle ***For Each Next***, ben vous savez comment déclarer votre variable ;).

Un petit inconvénient, il manque certains membres tels que *Value* dans la description de ***HTMLGenericElement***, ils sont implémentés et vous pouvez donc les utiliser sans problème, mais ne soyez pas surpris de ne pas les voir apparaître, ni lors de l'autocomplétion du code, ni dans l'inspecteur d'objets si cher à mon cœur (Excel 2007 - MsHtml.Tlb version 8.0.6001.18702).

Un autre avantage est d'éviter de multiplier abusivement le nombre de variables déclarées en pouvant utiliser une même variable pour pointer différents éléments de la page au fur et à mesure de l'avancement du code.

- « Ouais, admettons, mais justement, c'est quand même super lourd de déclarer une variable juste pour un élément et tu le fais souvent ! »

Cette fois, c'est vrai, vous avez raison, je le fais souvent, étant donné que vous avez tapé le code et que vous n'avez pas juste fait un copier/coller, vous avez dû remarquer que l'autocomplétion, si pratique, ne s'affiche plus après avoir tapé « ***document***.». Ceci est dû à la déclaration de ***document*** dans la Classe ***InternetExplorer*** qui est défini comme ***Object*** et non pas comme ***HTMLDocument***. Donc afin de remettre les choses en ordre, voilà pourquoi je place cet objet de type natif ***HTMLDocument*** dans une variable qui lui correspond. Ainsi on peut profiter de l'autocomplétion si pratique !

IV-G - Agir sur un lien HyperText

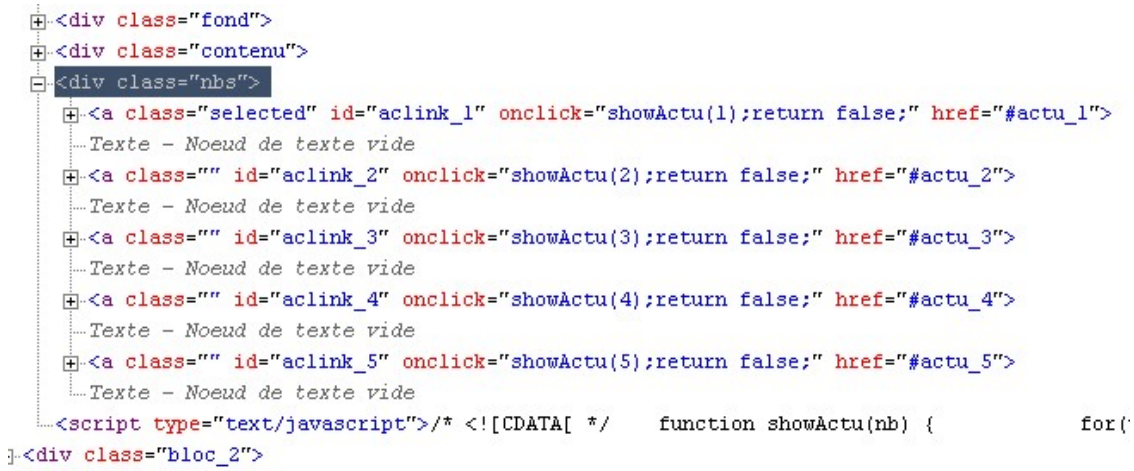
Une page internet contient souvent pléthore de liens internet, il serait donc intéressant de les retrouver dans notre objet ***IE*** et de naviguer grâce à eux. Nous allons utiliser le site **www.science.gouv.fr** afin de nous exercer. On peut voir sur le site des petites news, elles changent régulièrement et sont au nombre de cinq. On y accède via cinq petits boutons, qui sont en fait des liens, pour s'en convaincre, il suffit de survoler ceux-ci avec le pointeur de la souris. On voit alors apparaître dans la « Status bar » (ou Barre d'état), en bas de notre explorateur, les liens qui les composent (ex : http://www.science.gouv.fr/#actu_2)

On va profiter de ce cas pour utiliser l'**id**, car une nouvelle fois, cet élément n'a pas de propriété **Name** renseignée. Mais avant cela une petite mise en garde, je vous avais fait remarquer que les **id** n'étaient pas forcement uniques sur une page, mais ça n'est pas tout ! Parfois les **id** des éléments de la page sont générés automatiquement, à chaque chargement de la page, les rendant uniques mais variables. Si cela devait être le cas, il est évident que l'utilisation de l'**id** comme identifiant serait caduque.

Un appui sur F5 (rafraîchir) nous permet de conclure qu'ici l'**id** est fixe.

Alors pour pimenter un peu tout ça, que diriez-vous d'une petite boucle ***For Next***, vu que nous avons cinq liens représentant chacun une nouvelle, on va les faire défiler.

Étudions un peu cela



On apprend ici que nos éléments sont contenus dans des balises **<anchor**, donc ***HtmlAnchorElement*** dans VBA, je vous laisse vérifier avec l'espion, les **id** de nos liens commencent tous par « aclink_ » suivi du numéro d'ordre de la nouvelle.

VBA
```
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Sub LienHyperTextScienceActu()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim Actualite As HTMLAnchorElement
'Dim Actualite As HTMLLinkElement
Dim yNumActu As Byte


   'Ouvre la page Web + Affiche IE + Attente
   IE.Navigate "http://www.science.gouv.fr/"
   IE.Visible = True
   WaitIE IE

   'On pointe le document
   Set IEDoc = IE.document

   'On boucle de 2 à 5
   For yNumActu = 2 To 5
      'On pointe notre lien
      Set Actualite = IEDoc.anchors("aclink_" & yNumActu)
     'Set Actualite = IEDoc.links("aclink_" & yNumActu)
      'On exécute le lien
      Actualite.Click
      'On laisse 2 secondes au lecteur
      Sleep 2000
   Next

   Set IE = Nothing
   Set IEDoc = Nothing
End Sub
```

Nous avons ici un lien placé dans une balise **<anchors**, il sera donc, entre autres, référencé dans les ***links*** et dans les ***anchors***, nous aurions donc pu utiliser indifféremment l'une ou l'autre de ces collections qui, je ne vous le rappelle pas, font partie de la fameuse liste (pas cailloux, choux, genoux. hein), celle des ***DispHtmlElementCollection*** !

Je profite de cette notion de regroupement de balises pour vous présenter un exemple avec la fonction ***getElementsByTagName*** de l'objet ***document***, dans le paragraphe suivant.

IV-H - Collectionner les balises

Dans l'exemple précédent, nous utilisons le membre ***anchors***, afin de limiter la zone de recherche, mettons en place une petite comparaison.

Pour accéder aux objets à comparer, voici le code utilisé, qui retourne simplement une collection contenant l'ensemble de balises **<anchor** de la page.

VBA
```
Sub ScienceTags()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTagCol As IHTMLElementCollection
Dim Generic As HTMLGenericElement

   'Ouvre la page Web
   IE.Navigate "http://www.science.gouv.fr/"
   IE.Visible = True
   WaitIE IE
   Set IEDoc = IE.document

   'On liste les éléments de type anchor
   Set htmlTagCol = IEDoc.getElementsByTagName("a")

   Set IE = Nothing
   Set IEDoc = Nothing
End Sub
```

*Attention à la déclaration de **htmlTagCol**, l'objet est de type **IHtmlElementCollection**.*

Plaçons des espions sur ***IEDoc.anchors*** et ***htmlTagCol*** :

Espions

Expression	Valeur	Type
IEDoc.anchors	"[object]"	Object
constructor		Object
ie8_length	8	Long
length	8	Long
Item 1		Variant/Object/HTMLAnchorElement
Item 2		Variant/Object/HTMLAnchorElement
Item 3		Variant/Object/HTMLAnchorElement
Item 4		Variant/Object/HTMLAnchorElement
Item 5		Variant/Object/HTMLAnchorElement
Item 6		Variant/Object/HTMLAnchorElement
Item 7		Variant/Object/HTMLAnchorElement
Item 8		Variant/Object/HTMLAnchorElement
htmlTagCol	"[object]"	IHTMLElementCollection
constructor		Object
ie8_length	85	Long
length	85	Long
Item 1		Variant/Object/HTMLAnchorElement
Item 2		Variant/Object/HTMLAnchorElement
[...]		Variant/Object/HTMLAnchorElement
Item 85		Variant/Object/HTMLAnchorElement

Surpris ? Si pour vous ça n'est pas le cas, moi le résultat m'a étonné, je m'attendais en effet à retrouver deux listes identiques et m'apprêtais donc à vous dire que cette fonction n'était utile, que pour lister des balises autres que celles déjà regroupées dans les collections de ***document*** (***anchors***, ***links***...). Mais que nenni ! La réponse se trouve sur le site **Msdn** qui nous apprend que la collection ***document.anchors*** ne regroupe que les balises de type **<anchors** ayant un ***name*** et/ou un ***id*** de renseignés.

Une petite réflexion personnelle nous amène à une explication, ***anchors*** est de type ***DispHtmlElementCollection*** et on sait que ce type d'objet, nous permet de pointer un ou plusieurs membres qu'il contient, en lui passant en paramètre un **name** ou un **id**, il semble donc logique que celui-ci ne s'encombre pas à lister des balises qui n'ont pas ces attributs renseignés.

À propos du lien **Msdn** fourni au-dessus, attention, toutes les collections listées ne sont pas accessibles... fiez-vous à votre Inspecteur d'objets VBA... même si certains éléments n'y sont pas décrits non plus...

Ainsi si l'on souhaite lister les cellules des tables contenues sur notre page (balises **<td>**), on procédera de la sorte :

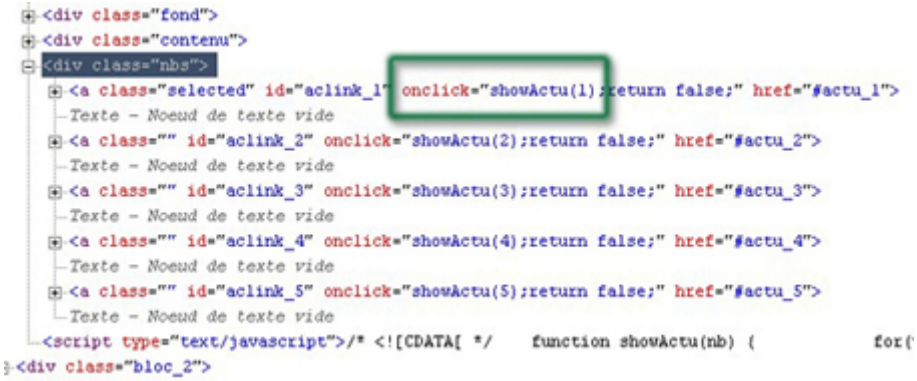
VBA
```
Sub ListerBaliseTd()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTagCol As IHTMLElementCollection
Dim Generic As HTMLGenericElement


    'Ouvre la page Web
    IE.Navigate "http://www.developpez.net/forums/"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On liste les éléments de type td
    Set htmlTagCol = IEDoc.getElementsByTagName("td")

    'On recherche la cellule ayant pour id "f26"
    Set Generic = htmlTagCol.Item("f26")

    Set IE = Nothing
    Set IEDoc = Nothing
End Sub
```

Nous voici donc sur le site **www.developpez.net**, dans l'objet ***Generic***, nous obtenons bien la cellule ayant pour **id** « f26 », j'ai pris celle-ci au hasard dans la source. Bien sûr ici, il n'y aurait aucun intérêt de passer par cette étape intermédiaire, un appel direct dans ***body.all*** nous aurait fourni le même objet.

IV-I - Utilisation des fonctions disponibles sur la page

Reprenons l'exemple précédent l'aparté sur getElementByTagName.

J'en vois déjà qui ont vu un autre détail sur cette capture d'écran :



En effet, vous avez bien vu, les cinq « boutons » Actu utilisent une fonction incluse dans le code source de la page, **showActu**, avec comme paramètre le numéro de l'actualité à afficher.

Pour les curieux qui souhaitent jeter un œil sur cette fonction, il y a l'onglet *Script* dans l'outil de développement d'Internet Explorer et voilà ce qu'une petite recherche sur **showActu** nous dévoile.

Donc plutôt que de rechercher notre « Bouton-Lien » et de cliquer dessus, nous allons nous contenter d'exécuter le même code que lui, ce qui donne :

VBA
```
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub ScriptScienceActu()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim yNumActu As Byte

   'Ouvre la page Web + Affiche IE + Attente
   IE.Navigate "http://www.science.gouv.fr/"
   IE.Visible = True
   WaitIE IE

   'On pointe le document
   Set IEDoc = IE.document

   'On boucle de 2 à 5
   For yNumActu = 2 To 5
      'On exécute le script showActu 'Attention à la Casse!!!! showActu
      IEDoc.parentWindow.execScript "showActu(" & yNumActu & ")", "JavaScript"
      'On laisse 2 secondes au lecteur
      Sleep 2000
   Next

   Set IE = Nothing
   Set IEDoc = Nothing
End Sub
```

Comme précisé dans le code, attention à la casse lors de l'inscription du nom du Script.
On perd vite une demi-heure à comprendre pourquoi ça ne fonctionne pas.

IV-J - Télécharger un fichier

Ici nous allons nous intéresser à la récupération d'une image contenue dans une page Web, il faudra pour cela utiliser la fonction suivante :

VBA
```
Sub SaveHtmlFile(aUrl As String, aDestination As String)
'Pris sur le forum de la msdn (avec quelques menues modifs)
'http://social.msdn.microsoft.com/Forums/en/isvvba/thread/bd0ee306-7bb5-4ce4-8341-edd9475f84ad
Dim WinHttpReq As Object, oStream As Object
Dim TheURL As String

   On Error Resume Next 'On ne gère pas les erreurs

   Set WinHttpReq = CreateObject("Microsoft.XMLHTTP")
   WinHttpReq.Open "GET", aUrl, False
   WinHttpReq.send

   TheURL = WinHttpReq.responseBody

   If WinHttpReq.Status = 200 Then
      Set oStream = CreateObject("ADODB.Stream")
      oStream.Open
      oStream.Type = 1
      oStream.Write WinHttpReq.responseBody
      oStream.SaveToFile aDestination
      oStream.Close
   End If
End Sub
```

Il permet de placer notre image dans un flux et d'enregistrer cette image dans un répertoire. Attention, si le fichier existe déjà, une erreur (non gérée et ignorée grâce à l'utilisation de *On error resume next*) est retournée. Une vérification de l'existence du fichier peut donc être implémentée au besoin.

Voilà l'image qu'on se propose de rapatrier et son code associé


```
▼ <body>
  ▶ <ul id="prelude">…</ul>
  ▼ <div id="container">
    ▼ <p id="img_home">
        <img src="img/img_home.jpg" alt>
      </p>
```

On l'utilisera de cette manière.

VBA
```
Sub ImageSite()
```

VBA
```
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim ImgElem As htmlImg
Const CheminRep As String = "c:\Logo\"

    'Ouvre la page Web
    IE.Navigate "http://www.cadastre.gouv.fr/scpc/accueil.do"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On recherche l'élément contenant le logo
    Set ImgElem = IEDoc.all("img_home").all(0)

    'On crée le répertoire où sera placée l'image
    'Si celui-ci existe déjà on ne gère pas l'erreur retournée et on passe à la suite
    On Error Resume Next
    MkDir CheminRep
    On Error GoTo 0 'On réactive la gestion d'erreur

    'On passe le chemin de l'image et le répertoire où elle sera copiée
    SaveHtmlFile ImgElem.href, CheminRep & ImgElem.nameProp

End Sub
```

*Vous trouverez d'autres méthodes de téléchargement dans le tutoriel d'**Arkham46** traitant de **VBA et du développement Web***

IV-K - Liste déroulante

Pour faciliter ou limiter les choix sur un formulaire, on utilise souvent des listes déroulantes (ComboBox), nous allons, dans un premier temps, voir comment sélectionner une des entrées contenues dans une liste déroulante. Puis acquérir le contenu d'une liste déroulante pour le stocker dans un tableau.

Un petit tour sur le site du cadastre va nous permettre de mettre en pratique.

IV-K-1 - Sélectionner une entrée


```
▼<select name="indiceRepetition">
    <option value>--</option>
    <option value="B">bis</option>
    <option value="T">ter</option>
    <option value="Q">quater</option>
  </select>
```

L'indice de répétition « ter » doit être sélectionné dans cette **liste déroulante**, observons le code qui lui est associé. La structure de la liste elle-même est réalisée par une balise **<select**, son contenu est renseigné à l'aide de balises **<option**, chaque entrée possible a sa balise correspondante ayant une **value** unique.
Je vous propose deux utilisations possibles, soit via la valeur unique, soit via l'index correspondant à la position de l'entrée dans la liste.

VBA
```
Sub ListeDeroulante()
'Selectionner une valeur dans une liste déroulante
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTagCol As IHTMLElementCollection
Dim htmlSelectElem As HTMLSelectElement
```

VBA
```
    'Ouvre la page Web
    IE.Navigate "http://www.cadastre.gouv.fr/scpc/accueil.do"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On va sur l'objet qui contient la liste des indices
    Set htmlSelectElem = IEDoc.all("indiceRepetition")

    'On sélectionne l'indice "ter" via sa valeur unique
    htmlSelectElem.Value = "T"

'## Autre solution ##
    'On sélectionne l'indice "ter" via l'index
    htmlSelectElem.selectedIndex = 2

'[...]
```

Voilà une bonne chose de faite, mais il serait intéressant de pouvoir utiliser le contenu de ces listes, dans un tableau Excel ou dans une liste déroulante de notre cru.

IV-K-2 - Récupérer les valeurs contenues dans la liste

Nous utiliserons la même liste déroulante que précédemment. Certains se disent certainement que la liste des départements serait plus intéressante ! Ce qui est mon avis également. Mais celle-ci cache un petit piège, qui sera traité quelques paragraphes plus bas et vous pourrez alors, comme exercice, reproduire ce qui va suivre pour fixer les connaissances acquises !

Nous n'irons pas jusqu'à la création d'une zone de liste modifiable dans le cadre de ce document, mais vous pourrez bien sûr utiliser les données que nous allons rapatrier dans un tableau, pour en alimenter une.

VBA
```
Sub ListeDeroulanteRecup()
'Selectionner une valeur dans une liste déroulante
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTagCol As IHTMLElementCollection
Dim htmlSelectElem As HTMLSelectElement
Dim NbrEntree As Integer
Dim TableauValeur()
Dim TheEntree As Integer

    'Ouvre la page Web
    IE.Navigate "http://www.cadastre.gouv.fr/scpc/accueil.do"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On va sur l'objet qui contient la liste des indices
    Set htmlSelectElem = IEDoc.all("indiceRepetition")

    'On détermine le nombre d'entrées contenues dans la liste
    'Ce nombre correspond au nombre de balises <option
    Set htmlTagCol = htmlSelectElem.getElementsByTagName("option")
    NbrEntree = htmlTagCol.Length

    'On redimensionne le tableau qui va contenir les valeurs
    ReDim TableauValeur(NbrEntree)


    'On boucle sur chaque entrée
    For TheEntree = 0 To NbrEntree - 1
       TableauValeur(TheEntree) = htmlTagCol(TheEntree).innerText
    Next
```

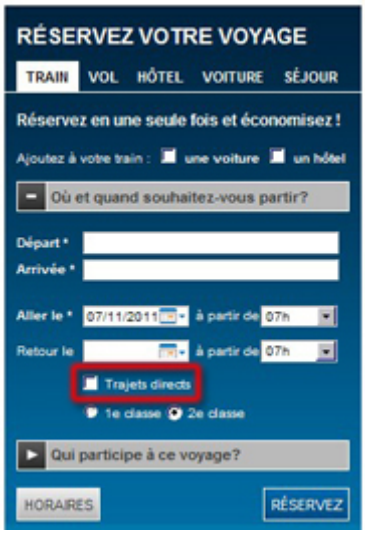
```vba
    'On place ces valeurs dans une Feuille Excel
    ThisWorkbook.Sheets("Valeur_Liste").[A1].Resize(NbrEntree).Value =
 WorksheetFunction.Transpose(TableauValeur)

    Set IE = Nothing
    Set IEDoc = Nothing
```

IV-L - Radio bouton et cases à cocher

Pour traiter ce sujet, nous allons voyager un peu. La SNCF va donc, pour un temps, nous accueillir sur son site, **http://www.voyages-sncf.com/**.

Les cases à cocher et les boutons radio ont des structures quasi similaires au niveau du code source de la page. Le code VBA utilisé pour actionner ces composants sera donc le même, je vous propose tout de même deux méthodes différentes, visant à choisir un trajet direct et une place en première classe.

IV-L-1 - Un trajet direct




```
<div class="travel-detail">
<p id="directTravelBox">
    <input name="direct_travel_check" id="direct_travel_check" type="checkbox" jQuery1320700146906="192" value="on"/>
    Texte - Noeud de texte vide
    <label for="direct_travel_check">
    Texte - Noeud de texte vide
<fieldset id="classTravel">
```

La case à cocher est constituée d'une balise **<Input** pour la partie saisie et d'une balise **<label**, qui lui est attachée pour la partie affichage du texte.

Nous allons agir sur l'élément ***Input*** afin de cocher la case avec le code suivant :

```vba
Sub CaseACocher()
'Agir sur un radio bouton
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim CaseDirect As HTMLInputElement
Dim Generic As HTMLGenericElement
Dim iCollection As IHTMLElementCollection

   'Ouvre la page Web
```

VBA
```
    IE.Navigate "http://www.voyages-sncf.com/"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On pointe la case à cocher
    Set CaseDirect = IEDoc.all("direct_travel_check")

    'On simule un clic sur la case à cocher
    CaseDirect.Click

End Sub
```

IV-L-2 - En première classe


```
<fieldset id="classTravel">
  <p>
    <input name="classe" id="classe1" type="radio" jQuery1320700146906="193" value="1"/>
    Texte - Noeud de texte vide
    <label for="classe1">
    Texte - Noeud de texte vide
  <p>
    <input name="classe" id="classe2" type="radio" CHECKED="checked" jQuery1320700146906="194" value="2"/>
    Texte - Noeud de texte vide
    <label for="classe2">
    Texte - Noeud de texte vide
```

Comme dit en amont, le code précédent fonctionnerait parfaitement dans ce cas de figure, mais il existe une autre possibilité que de cliquer sur l'élément, il est possible de modifier les attributs d'un élément de la page, ici, nous nous intéresserons à l'attribut ***checked***.

VBA
```
Sub BoutonRadio()
'Agir sur un radio bouton
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim BRFirstClass As HTMLInputElement

    'Ouvre la page Web
    IE.Navigate "http://www.voyages-sncf.com/"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document
```

VBA
```
    'On pointe le bouton radio première classe
    Set BRFirstClass = IEDoc.all("classe1")

    'On modifie son attribut checked
    BRFirstClass.setAttribute "checked", True

    '[...] Suite du code
```

Attention tout de même lors de l'utilisation de cette méthode, si à la suite du code nous rajoutons le code suivant :

VBA
```
 '[...]
    'On modifie de nouveau son attribut checked
    BRFirstClass.setAttribute "checked", False
```

On se retrouve alors avec les deux boutons radio en position non cochés, ce qui risque fort de poser quelques problèmes dans la suite de notre inscription.

Néanmoins, la méthode ***setAttribute*** permet de modifier la majorité des membres d'un élément, visibles en utilisant une fois de plus l'explorateur d'objets.

V - Cas concrets

V-A - Avec homonymie

Depuis le début, je vous dis de faire attention, lors de la recherche d'un élément de page par son **name** ou son **id**, à savoir que ceux-ci ne sont pas forcément uniques.

Il faut bien garder une chose en tête, il n'y a pas de code miracle, j'y travaille, mais ce n'est pas encore ça ;). Il faudra toujours prendre le temps de bien explorer votre page pour pouvoir faire un code efficace. On va voir comment travailler avec une liste d'objets, mais bien évidemment, on saura à l'avance lequel de ces objets va nous être utile.

Nous allons utiliser pour illustrer cela le site **http://www.science.gouv.fr/**, oui en effet je l'aime bien.

V-A-1 - Zone de texte

Si l'on regarde le code source de la page, on voit que la zone de texte a un **name** de renseigné, nous allons donc l'utiliser, comme précédemment.

VBA

```
Sub ScienceHomonymieAvErreur()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim Generic As HTMLGenericElement

    'Ouvre la page Web
    IE.Navigate "http://www.science.gouv.fr/"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On pointe l'élément "exa_q"
    Set Generic = IEDoc.all("exa_q")'##Erreur##

    Set IE = Nothing
    Set IEDoc = Nothing
End Sub
```

Échec : erreur d'exécution '438': Propriété ou méthode non gérée par cet objet.

Explication : on fait un tour dans l'arborescence de l'objet

IEDoc.all("exa_q") en plaçant un espion. ***All***, nous retourne deux items, nous n'avons pas pris le temps de bien étudier notre page Web avant de commencer notre code, revenons au code source et faisons une recherche plus poussée.

« exa_q » apparaît deux fois dans le code source de la page, voyons ce qu'il en est sur notre page

Nous avons en fait, deux fois la fonction de recherche sur la page, quelle importance me direz-vous, on s'en fiche, c'est la même boite de recherche, l'une ou l'autre fera l'affaire. À ceci, je répondrai oui, sauf que rappelez-vous le message d'erreur qui lui ne vous laisse pas autant de latitude.

La solution est simple et vous l'avez sûrement déjà trouvée, c'est de sélectionner un des deux items. Parce que le luxe, c'est le choix, je vous propose deux écritures différentes.

- Soit en utilisant un objet qui accepte les collections, puis en sélectionnant le premier item.

VBA
```
Sub ScienceHomonymieSoluce1()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim Generic As HTMLGenericElement
Dim iCollection As IHTMLElementCollection

    'Ouvre la page Web
    IE.Navigate "http://www.science.gouv.fr/"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On place les éléments "exa_q" dans la variable
    Set iCollection = IEDoc.all("exa_q")

    'On pointe arbitrairement le premier élément
    Set Generic = iCollection(0)

    'On place le texte de notre recherche
    Generic.Value = "Excel VBA"

    Set IE = Nothing
    Set IEDoc = Nothing
End Sub
```

- Soit en sélectionnant directement le premier Item de la collection sur la même ligne, on pourra alors supprimer la déclaration de la variable ***iCollection*** devenue inutile.

```VBA
'On pointe le premier élément de la collection
Set Generic = IEDoc.all("exa_q")(0)
```

Ou encore, pour détailler l'utilisation de la collection

```VBA
'On pointe le premier élément de la collection
Set Generic = IEDoc.all("exa_q").Item(0)
```

V-A-2 - Le bouton OK (des éléments qui font des petits)

Si l'on regarde avec l'outil de développement, on détecte le bouton sur la ligne :

```
<input class="submitimage" type="image" name="" id="exa_submit650" value="exa_submit" src="/img/exa/
ok.gif" />
```

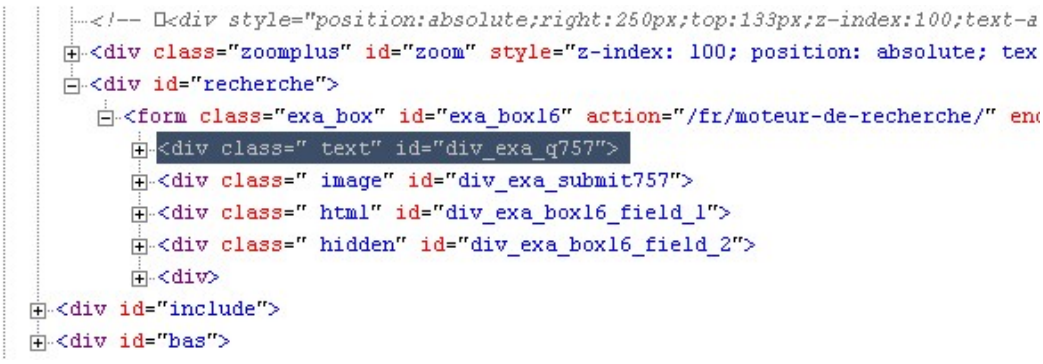
Or nous n'avons, ni **name**, ni **id** que nous puissions utiliser, **l'id** étant ici dynamique. Pire, un bouton doit exécuter une action, et ici l'image n'a pas d'action qui lui est assignée, elle renvoie juste un paramètre **value**. Par contre on peut voir le type **submitimage**. Tiens... **submit**. Regardons le code du container qui regroupe la zone de texte et le bouton.

```
<div id="recherche">
<form  id="exa_box85"  class="exa_box"  enctype="multipart/form-data" method="get" action="/fr/moteur-
de-recherche/">
```

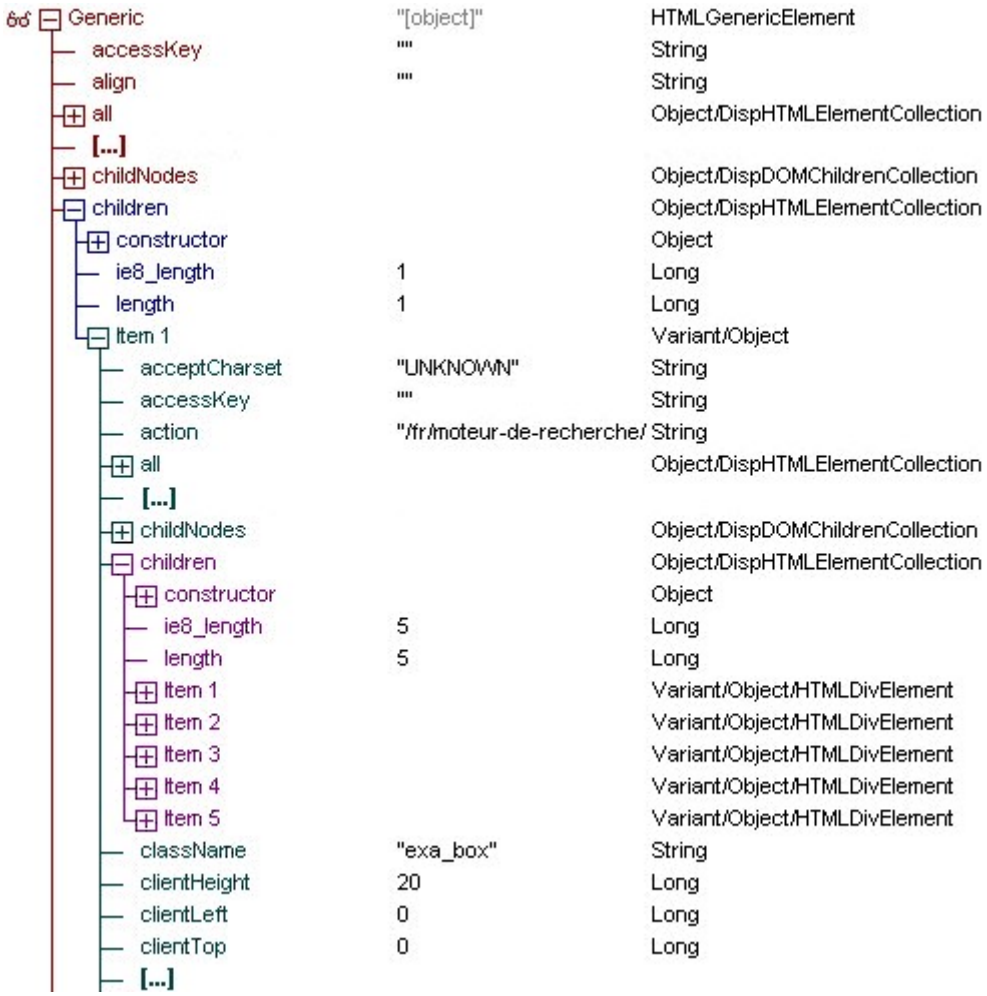
Ici, on trouve une balise **<form** et il apparaît bien une action qui charge une page Web nommée « /fr/moteur-de-recherche », ça parait bon signe. Et pour ce qui est d'identifier l'élément, nous allons remonter d'un cran et utiliser **id** « recherche » via l'utilisation de ***all***. Bien sûr, nous prenons soin de vérifier l'absence de doublon « id="recherche**"** » ainsi que « Name="recherche" », sur l'ensemble de la page.

Bien, il nous reste maintenant à pointer soit notre élément **<input**, pour agir directement sur lui, car via l'objet VBA nous aurons la méthode ***Click***, soit l'élément **form** pour utiliser sa méthode ***submit*** dans VBA.

Regardons l'organisation du code.



Notre élément ** ** a un lien hiérarchique avec l'élément **<form**, qui lui-même, a sous sa coupe cinq sous-éléments **<div**. Nous voyons apparaître ici la notion de père et fils, on dira donc que l'élément **<div** a un fils et que son fils (l'élément **form**) a quant à lui cinq fils. Voyons comment cela peut nous être utile dans notre objet VBA.



Nous retrouvons bien, au travers du membre ***children***, cette notion de filiation. On repère également la propriété action du premier Item1, qui correspond bien à ce que nous avions dans le code source de la page « /fr/moteur-de-recherche », ce qui nous conforte dans l'idée d'avoir affaire au bon élément. Nous allons pour l'exemple aller chercher notre bouton

Celui-ci est l'arrière-petit-fils de notre élément **<div** , ce qui donnerait

VBA
```
Sub ScienceHomonymieRechercheClickImage()
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim Generic As HTMLGenericElement

    'Ouvre la page Web
    IE.Navigate "http://www.science.gouv.fr/"
    IE.Visible = True
```

VBA
```
    WaitIE IE
    Set IEDoc = IE.document

    'On pointe le premier élément de la collection
    Set Generic = IEDoc.all("exa_q")(0)

    'On place le texte de notre recherche
    Generic.Value = "Excel VBA"

    'On recherche l'élément DIV contenant le bouton
    Set Generic = IEDoc.all("recherche")

    'On exécute l'action de la form
    Generic.Children(0).Children(1).Children(0).Click

    Set IE = Nothing
    Set IEDoc = Nothing
End Sub
```

En regardant le code, on s'aperçoit vite que l'utilisation de **submit** sera plus digeste, pensez toujours à la reprise de votre code, soit par vous-même plus tard, soit par un de vos collègues.

VBA
```
'On exécute l'action de la form
Generic.Children(0).submit
```

V-B - Méli-mélo de Name et d'Id

Reprendre le cas du cadastre avec la liste déroulante des départements, nous l'avions volontairement laissée de côté. En effet celle-ci a une particularité similaire au cas précédent, son **id** , « codeDepartement », est utilisé comme **name** pour un autre élément de la page.

```
▼<form name="critereRechercheForm" method="post" action="/scpc/aff
  ▼<div>
     <input type="hidden" name="codeDepartement" value>
```

Ici « codeDepartement » est utilisé comme nom pour un élément invisible, contenu dans la page.

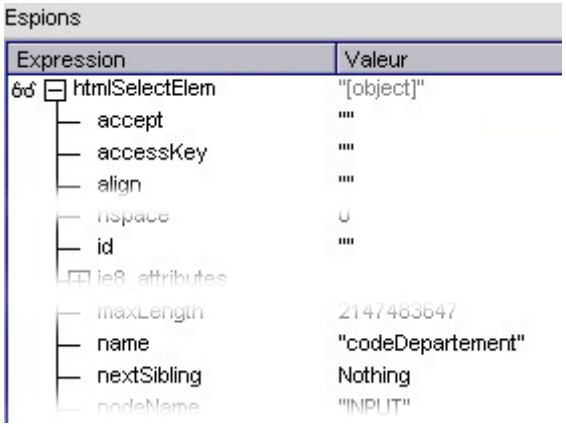
```
▼<p>
   <label for="codeDepartement" class="float">Département</label>
  ▼<select name="codeDepartement" class="long " id="codeDepartement">
     <option value>Choisir</option>
     <option value="001">01 - AIN</option>
     <option value="002">02 - AISNE</option>
     <option value="003">03 - ALLIER</option>
```

Ici « codeDepartement » est également utilisé comme nom pour notre Liste déroulante mais aussi, ce qui va être intéressant, comme Id. Vous allez me dire, pour ceux qui suivent, utilisons getElementById, ce à quoi je répondrai, oui, en effet cela semble parfaitement adapté à nos besoins. Mettons donc en place un petit exemple.

VBA
```
Sub ListeDeroulanteDepartement()
'Sélectionner une valeur dans une liste déroulante
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlSelectElem As HTMLSelectElement

    'Ouvre la page Web
    IE.Navigate "http://www.cadastre.gouv.fr/scpc/accueil.do"
    IE.Visible = True
    WaitIE IE
```

```VBA
    Set IEDoc = IE.document

    'On va sur l'objet qui contient la liste des départements
    'Un autre élément ayant "codeDepartement" pour nom existe
    'On va forcer IE à trouver l'élément ayant comme id "codeDepartement"
    Set htmlSelectElem = IEDoc.getElementById("codeDepartement")
```

Observons grâce à un espion le contenu de notre pêche



Pas vraiment le poisson attendu, comme je l'avais fait remarquer à la présentation de getElementById, il faut faire très attention lors de son utilisation, son nom étant assez trompeur au vu du résultat.

Il est donc préférable ici, de traiter cette particularité comme le cas d'homonymie précédent, le code corrigé pour répondre à nos attentes pourra ressembler à ceci par exemple :

```VBA
Sub ListeDeroulanteDepartement()
'Sélectionner une valeur dans une liste déroulante
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTagCol As IHTMLElementCollection
Dim htmlSelectElem As HTMLSelectElement
Dim NbrEntree As Integer
Dim TableauValeur()
Dim TheEntree As Integer

    'Ouvre la page Web
    IE.Navigate "http://www.cadastre.gouv.fr/scpc/accueil.do"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

    'On va sur l'objet qui contient la liste des départements
    'Un autre élément ayant "codeDepartement", comme nom existe
    'Il faut donc aller chercher celui dont nous avons besoin
    Set htmlSelectElem = IEDoc.all("codeDepartement")(1)

    'On détermine le nombre d'entrées contenues dans la liste
    'Ce nombre correspond au nombre de balises <option
    Set htmlTagCol = htmlSelectElem.getElementsByTagName("option")
    NbrEntree = htmlTagCol.Length

    'On redimensionne le tableau qui va contenir les valeurs
    ReDim TableauValeur(NbrEntree)

    'On boucle sur chaque entrée
    For TheEntree = 0 To NbrEntree - 1
       TableauValeur(TheEntree) = htmlTagCol(TheEntree).innerText
    Next
```

VBA
```
    'On place ces valeurs dans une feuille Excel
    ThisWorkbook.Sheets("Valeur_Liste").[C1].Resize(NbrEntree).Value =
  WorksheetFunction.Transpose(TableauValeur)

    Set IE = Nothing
    Set IEDoc = Nothing
End Sub
```

On remarque au passage que la structure du code pour la partie récupération des valeurs est le même que pour notre liste d'indices. Il serait donc peut-être intéressant, si vous avez souvent de tels éléments à traiter, de faire une fonction personnelle standard, qui vous permettra de récupérer le contenu d'une liste déroulante, en lui passant en paramètre l'élément contenant la liste, pour avoir en retour un tableau de valeurs par exemple.

V-C - Droit au but, un cas plus complexe

V-C-1 - Présentation du match

Celui-ci va se dérouler sur le site de la Française des jeux **Parions Sport**.

Pour gagner le match, plusieurs étapes devront être franchies, alors c'est parti.

À partir du tableau LotoFoot, voilà le tableau récapitulatif que l'on se propose d'obtenir à la fin de la partie.

	A	B	C	D
1	Loto Foot 7 n° 24			
2	n° Match	Equipe n°1	Equipe n°2	Résulat
3	1	Marseille	Lille	2
4	2	Montpellier	Rennes	2
5	3	Lyon	Arles Avignon	1
6	4	Brest	Bordeaux	2
7	5	Valenciennes	Monaco	N
8	6	Toulouse	Sochaux	2
9	7	Auxerre	Paris SG	1
10				

V-C-2 - Déroulement du match

V-C-2-a - Atteindre le stade

Arrivé sur le site **Parions Sport**, un joli carrousel nous accueille, le but ici est de nous rendre sur les résultats du Loto Foot.

Pour ne pas surcharger le tutoriel, je place ici le début de la procédure VBA, et je rajouterai la suite au fur et à mesure. Une fois n'étant pas coutume, pour une meilleure compréhension, je déclarerai les variables au fur et à mesure de l'avancement du code, je ne saurais que vous conseiller, une fois le code regroupé, de rassembler les déclarations en tête de code.

VBA
```vba
Sub fdjFoot()
'Lancer la recherche par le bouton
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTagCol As IHTMLElementCollection
Dim htmlInput As HTMLInputElement
Dim htmlGeneric As HTMLGenericElement

    'On ouvre la page Web
    IE.Navigate "https://www.fdj.fr/jouer/parions-sport"
    IE.Visible = True
    WaitIE IE
    Set IEDoc = IE.document

'#### Le Carrousel ####
    'On clique sur l'image Foot résultat
    Set htmlGeneric = IEDoc.images("img1")
    htmlGeneric.Click

    'On profite de l'animation ;)
    Sleep 1000

    'On clique sur le "bouton" valider
    Set htmlGeneric = IEDoc.body.all("CFBtConsulter")

    'Ici il faut aller chercher le anchor fils
    'de l'élément div pointé par htmlGeneric
    htmlGeneric.Children(0).Click

    'On patiente
    WaitIE IE

    '[...]
```

Rien de particulier, on utilise ***IEDoc.images*** pour limiter la zone de recherche.

V-C-2-b - Fortes intempéries : Match annulé

Parce que mère Nature, le foot, elle s'en fiche, des fois, le jour du match, c'est le déluge et vu qu'ils n'aiment pas bien mouiller leur maillot tout neuf, les joueurs rentrent à la maison.

Dans ce cas-là, un élément supplémentaire apparaît sur la page pour nous prévenir

Il nous suffira donc avant de passer à la suite de vérifier que cet élément est absent. « Il nous suffira », sauf qu'ici, notre élément, il n'a ni **Name**, ni **Id**.
Je vais donc profiter de ce petit contretemps, pour vous donner une fonction supplémentaire. Il n'existe pas, dans les différents outils mis à notre disposition, de fonction ***GetElementsBy*** permettant de rechercher des éléments en fonction de leur **Class**. Qu'à cela ne tienne ! En voici une de mon cru, que vous trouverez en annexe.

VBA
```
'[...]

'#### Match annulé ? ####

Dim htmlTabElement() As IHTMLElement

   'On vérifie l'absence du ClassName d'élément lotofoot_indispo
   htmlTabElement = getElementsByClassName(IEdoc.body, "lotofoot_indispo", False)

   'On regarde si notre variable contient des données
   On Error Resume Next
   TabVide = UBound(htmlTabElement)
   On Error GoTo 0

   If Not TabVide = -1 Then
      MsgBox "Le LotoFoot est annulé"
      Exit Sub
   End If

   '[...]
```

Petite explication sur ce code :
On sait que UBound peut potentiellement générer une erreur si **htmlTabElement** n'est pas un tableau.
Pour éviter qu'un message d'erreur fasse son apparition, on décide de bloquer la gestion des erreurs et pour cela on place **On Error Resume Next**. Si le code qui suit provoque une erreur, celle-ci n'est pas gérée (pas de message d'erreur) et l'exécution du code continue comme si de rien n'était.
Il faut bien sûr faire très attention lors de son utilisation, car vous ne saurez pas si votre code contient des erreurs puisqu'aucun message ne sera affiché.
Un fois la ligne passée, on réactive la gestion des erreurs avec **On Error GoTo 0**, dans la suite du code on se charge bien sûr de contrôler le contenu de **TabVide** pour savoir si tout s'est bien passé et connaître le nombre d'éléments contenus dans la variable **htmlTabElement** le cas échéant.

V-C-2-c - Mettre une tactique en place

En guise de terrain de jeu, nous aurons le tableau où se trouvent les résultats des matchs. Je vais juste afficher la partie qui nous intéresse, je vous laisse la rechercher dans l'ensemble du code.

```
<div class="lotofoot_resultats" id="filtres">
  <h3 class="tetiereblue tetiereTexte">
    <span class="bhgblue">
      Texte - Noeud de texte vide
    <span class="bhdblue">
      Texte -  Loto Foot 7
    <span class="sp">
      Texte - Noeud de texte vide
    <span class="sp">
  <div class="sportpdv_bg_res">
    <div class="sport_subtitle">
      <div class="intro"/>
    <div class="event_date_num">
      <div id="event_num">
        Texte - Evénement n° 24
      <div id="dates_valid">
      <div class="sp">
    <div id="lotofoot_pager">
    <table id="events">
      <tbody>
        <tr>
          <td class="index">
            Texte - 1
          <td class="teaml">
            Texte - Marseille
          <td class="outcomes">
            <img class="lf_1" alt="1" src=
            <img class="lf_N" alt="N" src=
            <img class="lf_C" alt="2" src=
          <td class="team2">
            Texte - Lille
        <tr>
```

Loto Foot 7

Derniers résultats Loto Foot 7

Evénement n° 24

1	Marseille	1 N ✓	Lille
2	Montpellier	1 N ✓	Rennes
3	Lyon	✓ N 2	Arles Avignon
4	Brest	1 N ✓	Bordeaux
5	Valenciennes	1 ✓ 2	Monaco
6	Toulouse	1 N ✓	Sochaux
7	Auxerre	✓ N 2	Paris SG

	A	B	C	D
1	Loto Foot 7 n° 24			
2	n° Match	Equipe n°1	Equipe n°2	Résulat
3	1	Marseille	Lille	2
4	2	Montpellier	Rennes	2
5	3	Lyon	Arles Avignon	1
6	4	Brest	Bordeaux	2
7	5	Valenciennes	Monaco	N
8	6	Toulouse	Sochaux	2
9	7	Auxerre	Paris SG	1
10				

Si on regarde notre tableau Excel de début de match, nous souhaitons avoir sur la première ligne, le numéro de l'événement, ici le n° 24 ainsi que le type de jeu, en l'occurrence Loto Foot 7.

Puis il nous faudra prendre les données de chaque ligne, afin de les placer dans l'ordre souhaité dans notre feuille Excel. **La partie centrale** étant constituée d'images, il faudra trouver le moyen de contourner cela, afin de placer un texte dans le tableau Excel correspondant aux informations données par les images.

V-C-2-d - Rentrer sur le terrain

Le nom du jeu est contenu dans une balise **<h3**, représentant un titre hiérarchique de rang 3. On pourra obtenir le nom du jeu directement dans sa propriété **InnerText** (se repère simplement avec un espion). On pourra accéder à cet élément, en ciblant dans un premier temps, sa balise parent dont l'**id** est « filtres ».

Le numéro d'événement quant à lui, est accessible directement dans la balise **<div** dont l'**id** est « event_num ».

VBA

```
'#### Le cartouche ####

Dim Ws_Scores As Worksheet
Dim sCartouche As String
```

VBA
```
Dim strTmp As String

    'On pointe la feuille Scores, si besoin & _
    Il faudra bien sûr ajouter une feuille nommée Scores
    Set Ws_Scores = ThisWorkbook.Sheets("Scores")

    'On récupère le nom du jeu
    Set htmlGeneric = IEDoc.body.all("filtres")
    sCartouche = Trim(htmlGeneric.all(0).innerText)

    'On récupère le n° de l''évènements
    Set htmlGeneric = IEDoc.body.all("event_num")
    strTmp = htmlGeneric.innerText
    strTmp = Trim(Left(strTmp, Length(strTmp) - InStrRev(strTmp, "")))
    sCartouche = sCartouche & " " & strTmp

    'On place l'information dans le tableau
    Ws_Scores.Range("A1") = sCartouche

    '[...]
```

Les éléments sont facilement accessibles, j'utilise quelques fonctions de base, dont vous trouverez facilement la description dans l'aide si besoin, qui me servent à extraire des parties d'une chaîne de caractères.

V-C-2-e - En route vers le but

Nous voici face à la défense, devant nous un tableau, avec un nombre de lignes indéfinies. On pourrait bien sûr se dire qu'il y en a sept et puis c'est tout. Sauf que si vous regardez le début du code, on a fait en sorte de toujours extraire les informations. On aurait déjà pu mettre Loto Foot 7 en cartouche, sans aller le chercher dans les éléments. Mais l'avantage d'aller chercher l'information, c'est que si on regarde les autres LotoFoot, le 15 par exemple, on voit bien que la structure est la même et qu'on pourra donc utiliser le même code. C'est quand même plus intéressant, que d'avoir plusieurs codes, qui au final, font la même chose... et ça ne coûte pas beaucoup d'effort.

Prenons le temps d'étudier l'adversaire.

La structure du tableau est simple, **<table** et **<tbody** représentent le tableau dans son ensemble. Chaque **<tr** génère une ligne et chaque **<td** tronçonne cette ligne en plusieurs cellules. Ici donc, sept lignes, chacune découpée en quatre cellules contenant les informations que nous devons extraire.

Première phase, on se rapproche des défenseurs, par chance on connaît bien un des joueurs dont l'**Id** est « events ». Sachez profiter des avantages du terrain, on va faire une petite passe à ForNext qui va se charger de dribbler de ligne en ligne et de tirer les informations nécessaires.

VBA
```
'[...]
'#### Les Données ####
Dim htmlTabResultat As HTMLGenericElement
Dim htmlLigneResultat As HTMLGenericElement
Dim NumLigne As Byte
Dim NumImg As Byte

   'On pointe sur la table contenant les données
   'Et on se décale sur le premier élément qu'il contient (le tbody)
   Set htmlTabResultat = IEdoc.body.all("events").Children(0)

   'On crée la ligne d'entête du tableau Excel
   'n° Match    Equipe n°1  Equipe n°2  Résulat
   Ws_Scores.Range("A2") = "n° Match"
   Ws_Scores.Range("B2") = "Equipe n°1"
   Ws_Scores.Range("C2") = "Equipe n°2"
   Ws_Scores.Range("D2") = "Résultat"

   'On commencera à renseigner le tableau à la ligne 3
   NumLigne = 3

   'On parcourt chaque ligne de la table Html
   For Each htmlLigneResultat In htmlTabResultat.Children


       '[...]
```

Encore une fois, ***Children*** nous sauve la mise, en nous permettant de connaître le nombre de lignes à traiter. Il serait donc possible d'utiliser ***htmlTabResultat.Children.Length*** si l'on souhaite connaître d'avance le nombre d'itérations qui seront à effectuer.

Nous voilà à deux pas du but, il ne nous reste plus qu'à relever le résultat du match, étudions un peu le code source une nouvelle fois

```
<div id="lotofoot_pager">
<table id="events">
  <tbody>
    <tr>
      <td class="index">
        Texte - 1
      <td class="team1">
        Texte - Marseille
      <td class="outcomes">
        <img class="lf_1" alt="1" src="//media.fdj.fr/generated/s/img/dm_1245846372_dot.gif"
        <img class="lf_N" alt="N" src="//media.fdj.fr/generated/s/img/dm_1245846372_dot.gif"
        <img class="lf_C" alt="2" src="//media.fdj.fr/generated/s/img/dm_1245846372_dot.gif"
      <td class="team2">
        Texte - Lille
    <tr>
```

Le **<td** dont la classe est « **outcom** », contient trois sous-éléments, constitués de deux informations différentes, du texte (membre **Alt**) et une image (membre **src**). C'est la partie image qui est affichée sur la page du site. Le résultat du match est traduit sur la page par une coche rouge.

Au niveau du code source, on peut différencier quatre classes différentes (seulement trois sur le screenshot), à savoir **If_1**, **If_2** et **If_N**en fonction de sa position 1, N ou 2 et If_C qui représente la valeur cochée (2 sur notre screenshot). On renseignera la colonne résultat de notre tableau Score, avec la valeur qui est cochée, identifiée par le nom de classe If_C.

VBA
```
'[...]

      'On récupère le contenu de chaque partie
      'Et on renseigne le tableau Score
      'Numéro de match
      Ws_Scores.Cells(NumLigne, "A") = htmlLigneResultat.Children(0).innerText
      'Equipe n°1
      Ws_Scores.Cells(NumLigne, "B") = htmlLigneResultat.Children(1).innerText
      'Equipe n°2
      Ws_Scores.Cells(NumLigne, "C") = htmlLigneResultat.Children(3).innerText
      'Résultat
      'On passe en revue les images
      For NumImg = 0 To 3
          With htmlLigneResultat.Children(2).all(NumImg)
              'On cherche l'image cochée
              If .ClassName = "lf_C" Then
                  'On note le résultat
                  Ws_Scores.Cells(NumLigne, "D") = .alt
                  'On quitte la boucle
                  Exit For
              End If
          End With
      Next
      'On passe à la ligne suivante du tableau
      NumLigne = NumLigne + 1
   Next

   '[...]
```

Et voilà, BUT !! Vous pouvez sortir les vouvouzellas, il ne reste plus qu'à mettre en forme le tableau, je vous laisse imaginer le code pour faire cela.

V-C-3 - Débriefing

Ce code n'est volontairement pas optimisé, il serait en effet préférable d'utiliser une variable tableau au niveau du code, afin d'augmenter la rapidité de traitement. Mais ceci irait à l'encontre de la facilité de compréhension du code, qui est l'objectif d'un tutoriel. Pensez juste à cela lors de vos réalisations, qui elles, demanderont certainement une rapidité accrue.

Pour vous prouver l'utilité d'avoir un code polyvalent, je vous propose ceci, placez un point d'arrêt sur la ligne suivante et lancez l'exécution du code.

VBA
```
'[...]

'#### Match annulé ? ####

Dim htmlTabElement() As IHTMLElement

   'On vérifie l'absence du ClassName d'élément lotofoot_indispo
   htmlTabElement = getElementsByClassName(IEdoc.body, "lotofoot_indispo", False) '### Point d'arrêt

   'On regarde si notre variable contient des données

   '[...]
```

Une fois le code interrompu, allez sur la page Web et affichez les résultats du LotoFoot 15, retournez ensuite dans votre code et terminez l'exécution du code.

Notre code est donc bien polyvalent !!

VI - Conclusion

Comme vous avez pu le voir, il faut gratter dans le code source afin de trouver ce à quoi vous voulez faire référence dans votre code, il faudra donc vous familiariser un minimum avec le contenu de la page sur laquelle vous allez travailler. Il existe beaucoup d'autres fonctions, l'arborescence des éléments est dense, je n'ai fait ici que vous présenter une partie de ce qu'il est possible de faire. On peut sentir en regardant le contenu des objets IE qu'il est possible de faire bien plus, peut-être l'objet d'un prochain tutoriel ! À vos claviers !

Dû à l'entendue des objets IE, il est important de prêter attention aux types des différents éléments que vous souhaitez manipuler, afin de bénéficier au mieux de l'aide à la saisie et des méthodes qui leur sont liées.

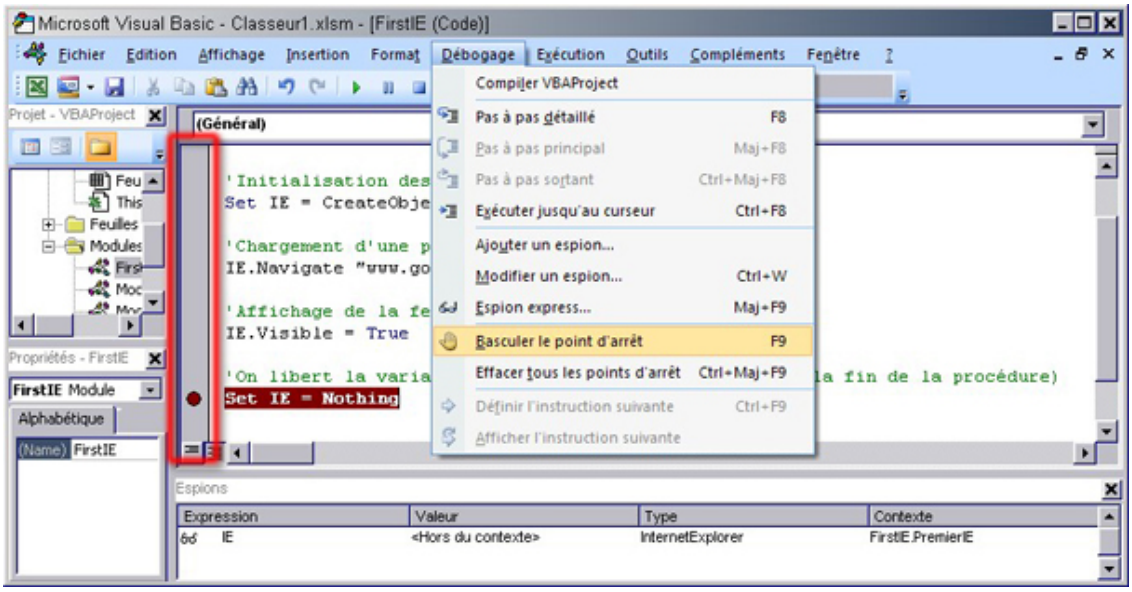
Quand j'étais en BEP, notre professeur d'électrotechnique quand il nous donnait des schémas à réaliser, nous disait toujours, « Messieurs, vous êtes 25, je veux 25 schémas différents ! ». Si je vous parle de ça, c'est qu'en programmation c'est pareil, il y a rarement une seule solution à un problème, soyez imaginatif ! Si une solution ne fonctionne pas, testez-en d'autres !

Gardez également à l'esprit qu'internet évolue, les sites Web aussi, même s'il n'est pas toujours en mouvement et heureusement dans le cadre de l'utilisation que nous venons d'en faire, le code d'une page peut évoluer, il vous faudra donc faire évoluer votre code en conséquence si vous voulez que celui-ci garde toute sa fonctionnalité.

VII - Les Annexes

VII-A - Le Point d'arrêt

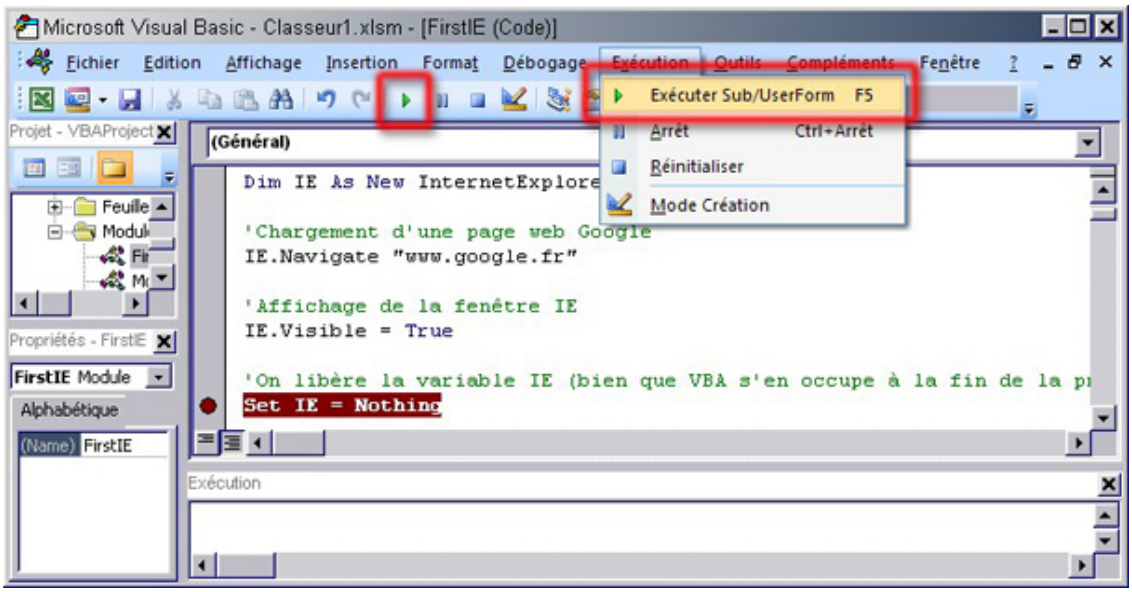
Il est particulièrement utile lorsque l'on souhaite regarder comment notre code modifie les variables. On peut ainsi choisir d'interrompre l'exécution du code à un endroit précis de celui-ci. Pour ce faire, nous allons donc placer un point d'arrêt, celui-ci devra nous permettre de visualiser le contenu de l'objet IE une fois celui-ci initialisé et avant qu'il ne soit libéré, nous allons donc stopper le code sur la ligne de libération elle-même, comme ceci.



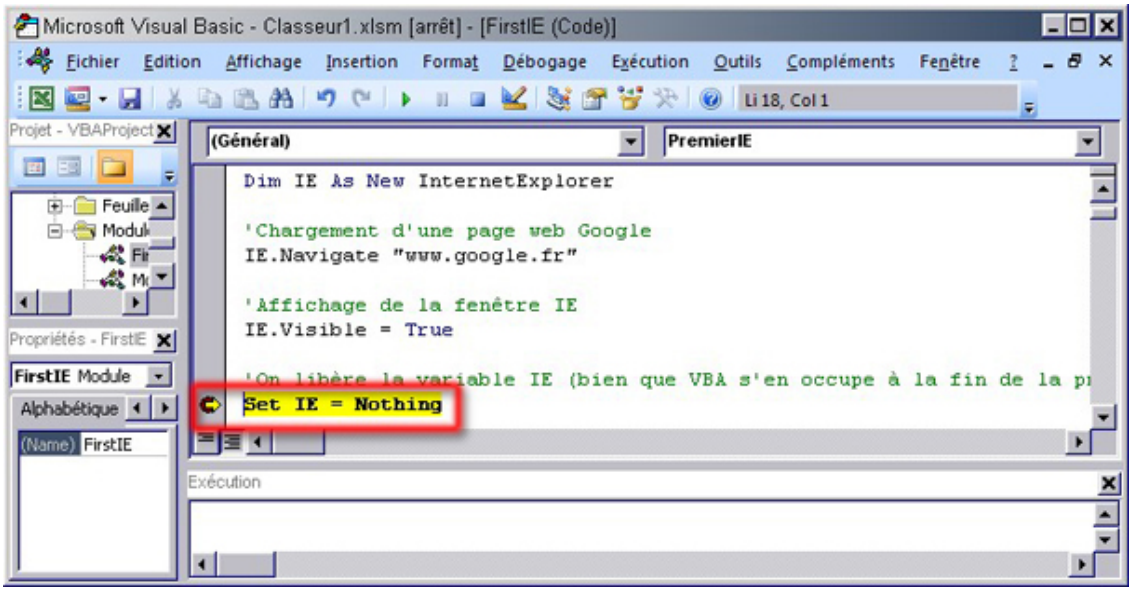
Après avoir cliqué dans la ligne où l'on veut stopper l'exécution du code, il existe trois méthodes pour placer le point d'arrêt.

- Par le menu : utilisez le menu *Débogage -> Basculez le point d'arrêt.*
- En utilisant le raccourci clavier : touche F9 du clavier (vous pouvez voir les raccourcis clavier associés aux actions en regardant dans la partie droite des menus déroulants).
- **La gouttière** : elle se trouve sur la partie gauche de l'éditeur de code, un clic de souris dans celle-ci place ou retire un point d'arrêt, un point rouge apparaît dans la gouttière et la ligne est surlignée en rouge.

Notre point d'arrêt étant en place, lançons l'exécution de notre code, menu Exécution -> Exécuter Sub/UserForm (raccourci F5) ou le bouton lecture sur la barre de menu.



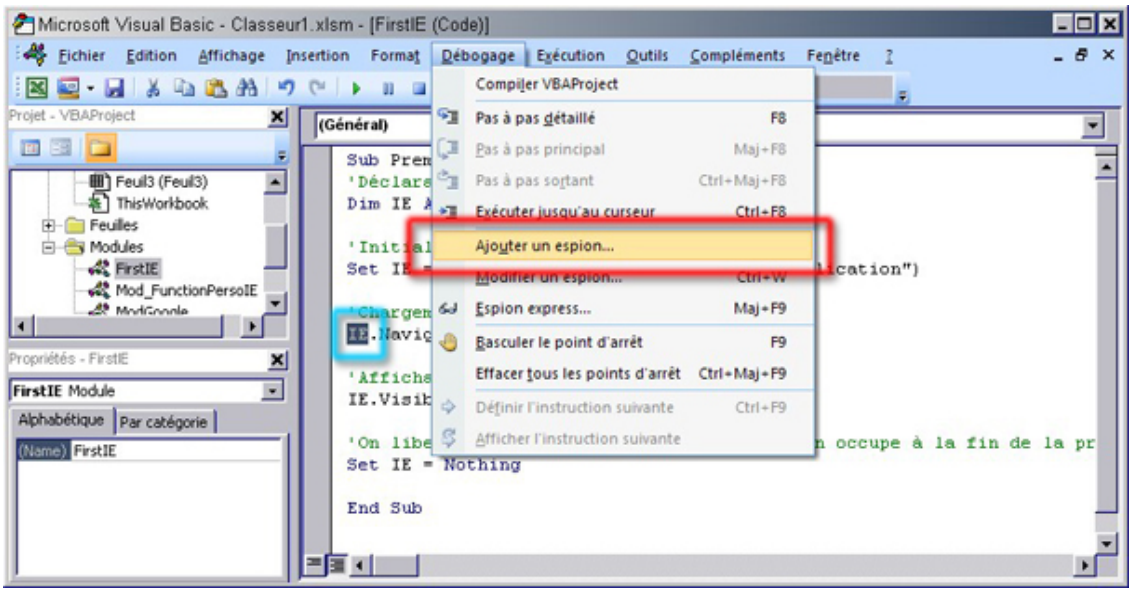
Le code va être exécuté jusqu'à ce que le point d'arrêt soit rencontré, l'exécution du code sera alors mise en attente.

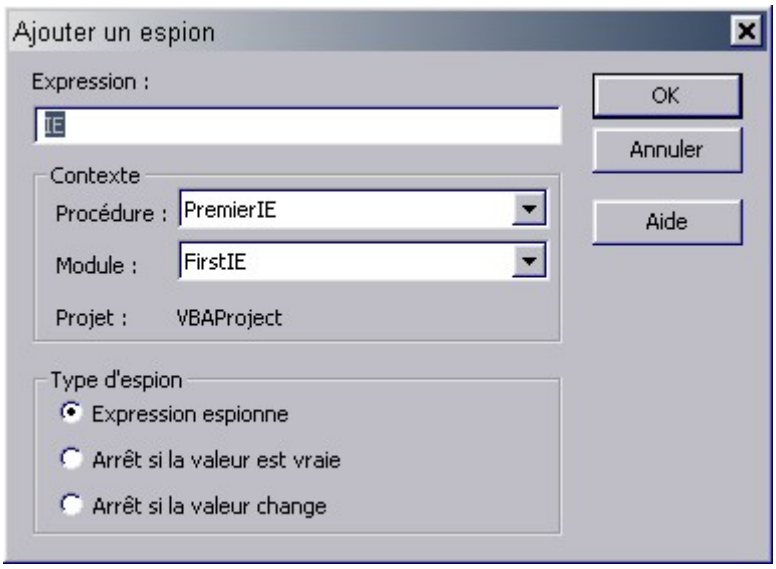


La flèche jaune nous indique où est stoppée l'exécution du code, cette ligne de code n'est pas encore exécutée. Il vous est possible dans cette configuration, en survolant le code avec votre souris, de connaître la valeur contenue dans une variable simple (***String***, ***Integer***, ***Boolean...***) ou de savoir si une variable de type objet est définie ou non. Le contenu apparaît dans une info-bulle, ***Nothing*** apparaît dans le cas d'un objet vide, pour avoir plus de détail sur les objets ou pouvoir suivre l'évolution d'une variable, vous devrez utiliser un espion dont l'utilisation est expliquée dans le chapitre suivant.

VII-B - L'espion

Nous allons commencer par l'espion, **sélectionner le terme** IE dans le code, puis dans le menu *Débogage* -> **Ajouter un espion.**

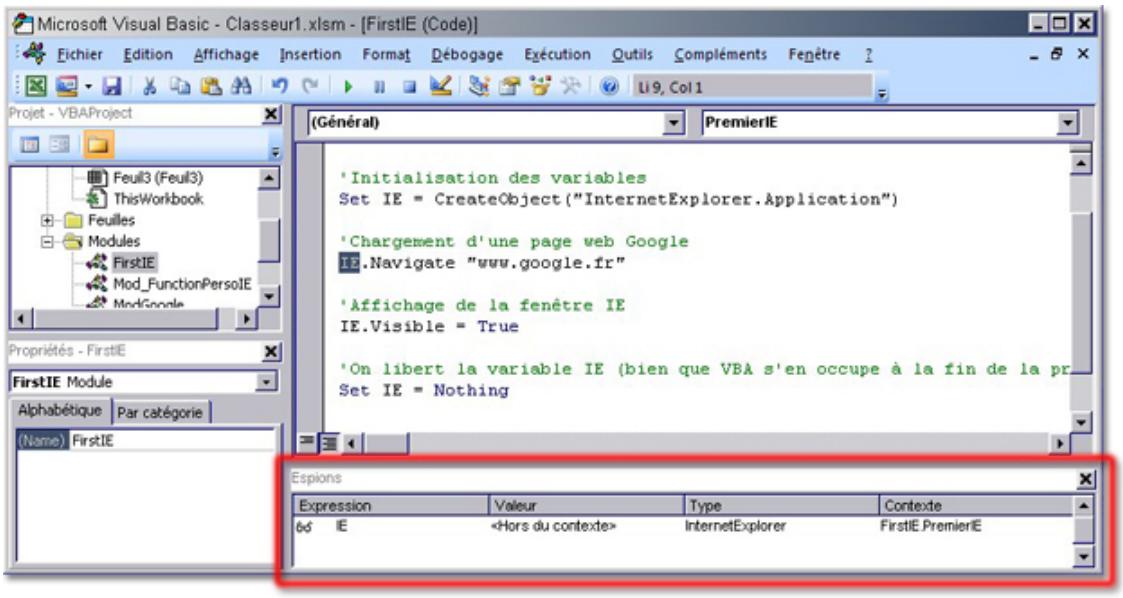




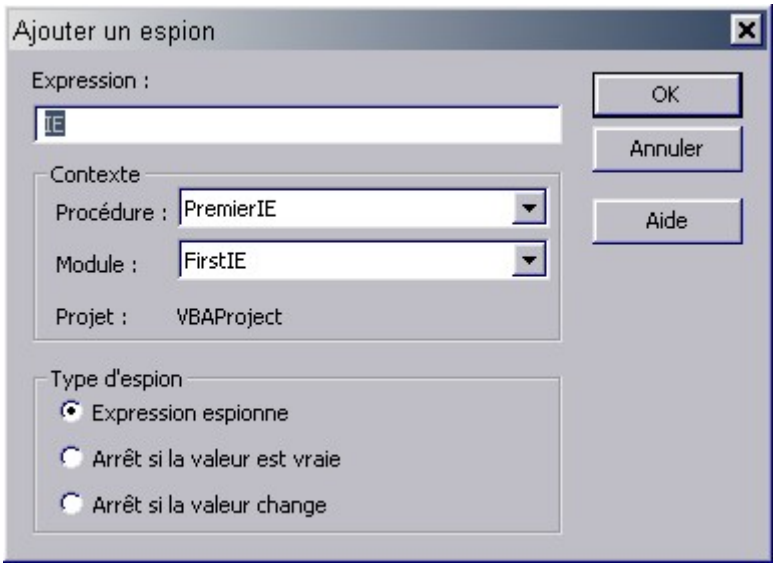
Une nouvelle fenêtre va apparaître.

Son contenu sera peut-être légèrement différent de celui figurant sur ma capture d'écran, laissez les valeurs par défaut présentes dans votre fenêtre et validez par OK.

Vous verrez apparaître ce nouvel espion dans la fenêtre Espions, que vous pourrez intégrer à l'environnement VBA comme ceci



Il vous est possible à tout moment de modifier cet espion, en effectuant un clic droit sur celui-ci et en choisissant « Modifier un espion. ». Vous retrouverez la fenêtre précédente, que je vous présente rapidement.



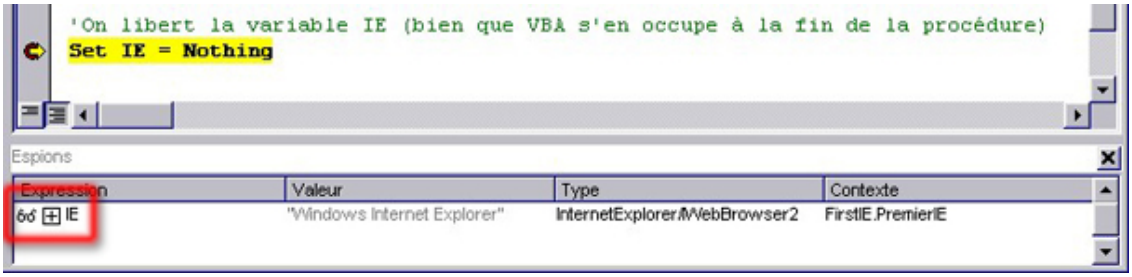
- Dans le champ « Expression », vous pouvez soit mettre un nom de variable, soit mettre une expression retournant une valeur. Par exemple, si vous souhaitez savoir si une variable de type numérique, que nous nommerons ***VarNumeric*** pour l'exemple, est égale à 3, vous placez dans le champ, ***VarNumeric = 3***.
- Dans la partie « Contexte », vous définissez où se trouve la variable à espionner, vous utiliserez peu cette partie, inutile de s'y attarder.
- Dans la section « Type d'espion », il vous est possible de modifier le fonctionnement de l'espion, par défaut « Expression espionne » est validée et permet de suivre en temps réel le contenu d'une variable. « Arrêt si la valeur est vraie » permet, si l'on espionne une expression de type booléen, de mettre en pause l'exécution du code si celle-ci retourne *Vrai*. Ceci est particulièrement intéressant lorsque, par exemple, vous voulez suivre l'exécution du code en mode pas à pas, mais uniquement si une variable donnée à une valeur *Vrai*. « Arrêt si la valeur change », le comportement est similaire au type précédent, hormis le fait que l'expression peut être de n'importe quelle sorte. Pour une expression numérique, le code passera en mode pause si vous augmentez la valeur de la variable par exemple.

Les espions peuvent également vous permettre de regarder la structure d'un objet, ainsi que les valeurs de ses propriétés. Prenons l'exemple d'un objet ***InternetExplorer*** (au hasard), dans le code suivant.

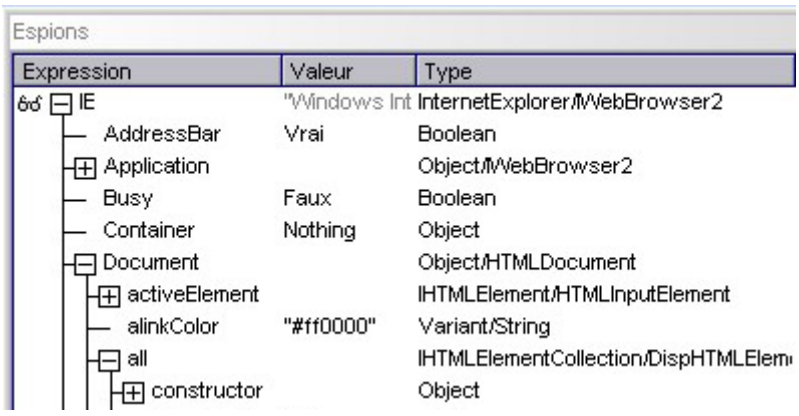
VBA
```
Sub PremierIE()
'Déclaration des variables
Dim IE As InternetExplorer

   'Initialisation des variables
   Set IE = CreateObject("InternetExplorer.Application")

   'Chargement d'une page Web Google
   IE.Navigate "www.google.fr"

   'Affichage de la fenêtre IE
   IE.Visible = True

   'On libère la variable IE
   Set IE = Nothing

End Sub
```

et plaçons un point d'arrêt sur la ligne ***Set IE = Nothing***. Ainsi qu'un espion sur la variable ***IE***.

Lancez l'exécution de notre code VBA, celui-ci va lancer une session d'Internet Explorer et faire une pause sur la ligne contenant le point d'arrêt et attendra une action de votre part pour continuer. Nous allons donc profiter de cette petite pause pour regarder ce que contient notre objet ***IE***.



On remarquera que le contenu de notre espion a changé. Et qu'il est désormais affublé d'un petit ⊞ nous indiquant qu'il contient quelque chose. Un clic sur ce + permet d'accéder au cœur de l'objet et d'y espionner le contenu de ses propriétés.



Il est également possible de modifier le contenu d'une variable ou le contenu d'une propriété d'un objet, directement dans la fenêtre d'espionnage. Pour ce faire, sélectionnez l'espion dont vous souhaitez modifier la valeur (la ligne passe en bleu), puis cliquez une seconde fois, toujours sur le même espion, mais cette fois dans la colonne Valeur, la valeur passe en mode édition. Attention bien sûr à respecter le type de la variable (texte, numérique...). Il est à noter également que les variables en lecture seule ne sont pas modifiables via ce procédé.

Ne vous privez pas de les utiliser, les espions sont très utiles, pour déboguer votre code mais également pour comprendre un code réalisé par quelqu'un d'autre.

VII-C - GetElementsByClassName

Étant donné qu'elle n'existe pas, je vous propose une version de ***getElements*** qui permet de retourner un ensemble d'éléments identifiés par leur ***ClassName***.

VBA - Fonction getElementsByClassName

```
Function getElementsByClassName(IEParentElement As IHTMLElement, aClassName As String, Optional
 JustChildren As Boolean = False) As IHTMLElement()
'Retourne un tableau contenant les éléments de la page ayant pour Class aClasseName
Dim aElement As IHTMLElement
Dim FuncElements() As IHTMLElement
Dim SourceElem As IHTMLElementCollection
Dim iElem As Integer

   'On prend en compte le lieu de recherche
   If JustChildren Then
       'Ici on ne tiendra compte que des enfants directs de IEParentElement
       Set SourceElem = IEParentElement.Children
   Else
       Set SourceElem = IEParentElement.all
   End If

   'On boucle sur tous les éléments contenus dans SourceElem
   For Each aElement In SourceElem
       'On vérifie si l'élément correspond à notre recherche
       If aElement.ClassName = aClassName Then
           'On redimensionne notre tableau
           'Cela semble inutile de regarder si FuncElements est un Array...
           'Mais sans cette ligne FuncElement n'est jamais reconnu comme tel...
           iElem = IIf(IsArray(FuncElements), UBound(FuncElements) + 1, -1)
           ReDim Preserve FuncElements(iElem)
           'Et on place l'élément trouvé à l'intérieur
           Set FuncElements(UBound(FuncElements)) = aElement
       End If
   Next
   'On place le tableau en retour de notre fonction
   getElementsByClassName = FuncElements
   'On libère l'espace mémoire occupé par notre tableau provisoire
   Erase FuncElements
End Function
```

Éléments	Déscription
IEParentElement	Nom de l'élément dans lequel sera recherché les éléments ayant comme classe *aClassName*.
aClassName	Chaîne de caractères représentant le nom de classe qui sera recherché.
JustChildren	Facultatif. Cette valeur de type booléen prend la valeur ***False*** (valeur par défaut) si la recherche doit se faire dans l'intégralité des sous-éléments. Il prend la valeur ***True***, si la recherche doit se faire uniquement dans les éléments fils, de l'élément *IEParentElement*, la recherche s'effectue alors dans les éléments contenus dans le membre ***Children*** de l'objet ***IEParentElement***.

Valeur retournée : Tableau de base 0 et de type IHTMLElement, contenant les éléments correspondant aux critères de recherche.

Voici un exemple simple d'utilisation de cette fonction, qui permet de retrouver un élément de la page grâce à sa classe.

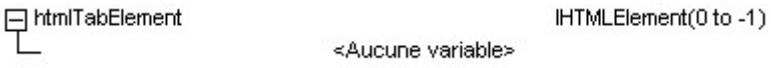
L'exemple suivant joue sur l'utilisation de *JustChildren*. Nous reprendrons pour cela le tableau des scores du LotoFoot et son code associé.

Dans un premier temps, nous allons limiter la recherche aux éléments fils de la balise **<table**, que l'on peut voir sur la deuxième ligne du code ci-dessus.

VBA

```
Sub TestgetElementsByClassName()
'Déclaration des variables
Dim IE As New InternetExplorer
Dim IEDoc As HTMLDocument
Dim htmlTabElement() As IHTMLElement
Dim GenericElem As HTMLGenericElement

   'Chargement d'une page Web Google
   IE.Navigate "https://www.fdj.fr/jeux/lotofoot7et15/resultats"
   IE.Visible = True
   WaitIE IE
   Set IEDoc = IE.document

   'On recherche l'élément le plus proche du tableau
   Set GenericElem = IEDoc.all("events")

   'On recherche les éléments ayant "index" comme classe
   'et qui se trouvent dans les enfants directs de GenericElem
   htmlTabElement = getElementsByClassName(GenericElem, "index", True)

   '[...] Suite du code
```

Un espion nous permet de connaîtreconnaitre le contenu retourné



Conformément au contenu du code, aucun des éléments fils n'ayant une classe correspondante à notre recherche, le tableau retourné est donc vide.
Cette fois incluons l'ensemble des sous-éléments composant le tableau et examinons le contenu du tableau retourné.

VBA

```
'On recherche les éléments ayant "index" comme classe
'et qui se trouvent dans tous les éléments constituant le tableau
htmlTabElement = getElementsByClassName(GenericElem, "index", False)
```

Cette fois nous obtenons sept éléments dans le tableau, ils représentent bien les sept cellules de la première colonne, contenant les index des lignes.
Il est ainsi possible de détecter le nombre de lignes composant le tableau et par exemple de dresser la liste des équipes ou des résultats des matchs.

VIII - Liens utiles

Ce tutoriel vient en complément de celui d'**Arkham46** traitant de **VBA et du développement Web**.
Le tutoriel de **Josselin Willette** traitant du **code HTML**.
Le tutoriel de **Maxence Hubiche** sur **le Late et Early Binding**.
Le site Web de **la base de connaissances Microsoft**.
Toujours la base de connaissances Microsoft, au sujet des **Outils de développement** intégrés à Internet Explorer.
Je vous propose Source **un fichier Excel** contenant l'ensemble des codes source utilisés dans ce document. Il contient également le UserForm "Inspecteur d'état" et une ébauche de UserForm d'attente.

IX - Remerciements

Je tiens tout particulièrement à remercier Pierre Fauconnier, qui m'a contacté pour me proposer l'élaboration de ce document et qui m'a guidé durant la rédaction. J'espère prendre le temps d'en réaliser de nouveaux, la transmission de connaissances étant, à mon avis, une solution intéressante pour faire évoluer l'utilisation de l'informatique (et autre), en la rendant accessible au plus grand nombre.

J'en profite aussi pour remercier les gens qui participent au site DVP, sur les forums où je me suis nourri de matière première en vue de l'élaboration de ce document, ainsi que les correcteurs techniques qui ont pris le temps de lire et commenter mon document. Un grand merci également au correcteur orthographique cette fois, qui vous permet de lire ce document sans vous brûler les yeux. L'orthographe n'étant pas particulièrement mon fort, alors merci pour le temps passé à la relecture. En particulier à Claude Leloup qui a corrigé ce document !