

WINTASK

**Développer des scripts
d'automatisation Web fiables
et efficaces**

Version 3.8



AUTOMATE POUR WINDOWS XP, 2003, 2008, Vista et Windows 7

Publié par : TaskWare
18, allée des Lilas
92150 SURESNES France

© Copyright 1997-2011 TaskWare Octobre 2011

WinTask™ est une marque déposée de TaskWare.

TABLE DES MATIERES

Introduction	5
WinTask, pour quoi faire ?	5
Les modules de WinTask	5
TASKEDIT	5
TASKCOMP	5
TASKEEXEC	6
ESPION	6
TASKWIZ	6
EDITEUR DE BOÎTES DE DIALOGUE	6
Mode Enregistrement	7
Exercice 1	7
Mythe des outils de Capture/Rejoue	8
Un Script simple d'automatisation	8
Editeur WinTask	11
Exercice 2	11
Arrêter l'exécution d'un script	12
Synchronisation	13
Synchronisation quand les pages du site ont le même titre	13
Exercice 3	13
Synchronisation manuelle	14
Exercice 4 : Synchronisation sur texte OCR	16
Le langage WinTask	23
Instructions du langage	23
Variables	25
Variables directives	25
Entiers	25
Chaînes de caractères	25
Nombres réels	26
Tableaux	26
Opérateurs	26
Instructions de gestion des pages	29
#IgnoreErrors	29
#ActionTimeout	29
#UsePageExact	30
Itération	32
Itération	32
Exercice 5 : Itération cliquant la ième page	32
Instructions de gestion des fichiers	34
Exist()	34
Kill()	34
Read() et Write()	35
ReadExcel et WriteExcel	35
Instructions de capture de données Web	38
Exercice 6 : Capture de données Web	38
Saisie automatique dans un formulaire Web	46
Exercice 7 : Saisie de constantes dans un formulaire	46

Exercice 8 : Saisie de données issues d'Excel dans un formulaire	47
Autres instructions pour les formulaires	49
Fonctions et Sous-Programmes (Sub)	52
Sub...EndSub	52
Function...EndFunction.....	52
Script complet de capture sur un site "réel"	54
Exercice 9 : Script complet de capture	54
Débogage.....	69
Erreurs de compilation.....	69
Erreurs d'exécution	69
Trace en utilisant MsgBox ou MsgBoxFrame	69
Exercice 10 : Tracer l'exécution via MsgBox ou MsgBoxFrame	69
Fichier Rapport	70
Exercice 11 : Tracer l'exécution via un fichier Rapport (fichier log)	70
Exécution en mode Débogage	71
Délai d'attente dépassé lors du chargement d'une page	78
ClickHTMLElement ne génère pas d'erreur mais ne clique pas	78
Navigate, une solution de contournement	79
Saisie incorrecte dans un champ	79
Erreur élément HTML non trouvé.....	80
La capture d'une Table ne renvoie pas les données attendues.....	81
Exercice 12 : Modification du mot-clé CONTENT pour une capture correcte	81
Option de Menu de Internet Explorer non sélectionnée	89
Option de Menu d'une page Web non sélectionnée	89
La sélection d'une option d'un menu contextuel ne se rejoue pas	90
Enregistrement Bas Niveau quand rien d'autre ne marche.....	90
Le rejoue est lent.....	92
L'Assistant de Capture ne permet pas de capturer les données désirées	92
Conclusion	93
APPENDICE A	95
Barre d'outils de l'Editeur WinTask.....	95
APPENDICE B	97
Barre d'outils WinTask flottante.....	97
APPENDICE C	99
Solutions des exercices	99
Exercice 3, Scriptweb03.src	99
Exercice 5, Scriptweb05a.src	101
Exercice 5, Scriptweb05b.src	102
Exercice 6, Scriptweb06.src	103
Exercice 7, Scriptweb07.src	104
Exercice 8, Scriptweb08a.src	105
Exercice 9, Scriptweb09.src	107
Exercice 10, Scriptweb10.src	109
Exercice 11, Scriptweb11a.src.....	110
Exercice 12, Scriptweb18.src	112
GLOSSAIRE	113
Index	115

Introduction

Ce manuel a été écrit pour vous aider dans le développement de vos scripts d'automatisation de sites Web. Nous avons cherché à vous faire part de notre expérience issue de l'écriture de nombreux scripts et des questions les plus usuelles transmises à notre support technique. Le manuel couvre l'interface WinTask et les instructions du langage d'automatisation.

Ce manuel inclut de nombreux exercices pour illustrer le propos et montrer comment éviter les erreurs les plus courantes. Les scripts d'automatisation utilisent des pages Web simples accessibles par tous sur notre site www.wintask.fr ou www.wintask.fr/demos. Les techniques présentées s'appliquent ensuite sans difficultés aux sites Web réels que vous désirez automatiser. Ce manuel ne traite pas de l'automatisation d'applications Windows, sujet traité dans un autre manuel disponible également sur www.wintask.fr/manuels-wintask.php.

WinTask, pour quoi faire ?

WinTask est une solution complète d'automatisation pour sites Web et applications Windows. Le logiciel peut automatiser n'importe quel site Web et capturer des données sur des pages Web. Avec le moteur OCR inclus dans WinTask, il est même possible de capturer du texte affiché à l'intérieur d'une image. WinTask est plus qu'un simple enregistreur de macros. Le script d'automatisation peut être enrichi grâce aux 300 instructions du langage. WinTask inclut un Planificateur de tâches (non disponible sous Vista, Windows 7, 2008 Server) permettant d'ouvrir un bureau Windows automatiquement, d'exécuter le script et de refermer le bureau une fois la tâche terminée. Sous Vista, Windows 7 ou Windows 2008 Server, vous pouvez utiliser le Planificateur inclus dans ces versions de Windows pour lancer des scripts WinTask.

Les modules de WinTask

WinTask inclut plusieurs modules. Chacun est discuté plus en détails dans la suite et les exercices montrent comment les utiliser.

TASKEDIT

TaskEdit est l'environnement de développement des scripts WinTask. Grâce à cet Editeur, l'utilisateur peut modifier les instructions générées automatiquement par le mode Enregistrement et enrichir le script à l'aide des nombreuses instructions du langage. L'Editeur comporte plusieurs fenêtres, la syntaxe des mots-clés est en couleurs et l'aide sur chaque instruction est disponible en double-cliquant sur son nom. Les fichiers Script écrits dans cet environnement de développement sont ensuite compilés puis exécutés afin de rejouer les actions décrites dans le script. Les fichiers Script de WinTask sont au format ASCII (ou Unicode) et portent l'extension *.SRC*.

TASKCOMP

Le module Compilateur de WinTask est un *.EXE* qui, via une ligne de commande, compile un script WinTask en un fichier exécutable par le module d'exécution de WinTask. Le Compilateur peut également être lancé à partir de l'Editeur. Le Compilateur vérifie la syntaxe des instructions listées dans le fichier *.SRC* et génère un fichier d'extension *.LST* si des erreurs de syntaxe sont trouvées. Si aucune erreur de

compilation n'est trouvée, le Compilateur génère un fichier prêt à être exécuté, fichier binaire d'extension *.ROB*. Un fichier *.LST* est un fichier ASCII contenant les erreurs et les avertissements renvoyés par le Compilateur.

TASKEEXEC

C'est le module d'exécution. Il peut être appelé via une ligne de commande ou directement via l'Editeur. Il exécute ligne par ligne les instructions rencontrées dans le fichier d'extension *.ROB*, résultat de la compilation du script source.

ESPION

Le module Espion permet de connaître la structure interne des différents objets affichés sur le bureau Windows, en particulier les objets HTML présents dans une page Web. L'objet HTML sur lequel doit s'effectuer l'action est spécifié par son descripteur HTML, identifiant unique de l'objet pour la page considérée. Ce descripteur HTML est utilisé par de nombreuses instructions Web du langage WinTask.

TASKWIZ

Ce module regroupe les outils de synchronisation et de capture de données. Dans l'automatisation de sites Web, l'assistant de capture de données est celui le plus utilisé. La plupart du temps, les outils de synchronisation ne sont pas nécessaires pour attendre que la nouvelle page Web soit complètement chargée avant d'effectuer l'action suivante. Ces assistants de synchronisation sont décrits en détail dans le chapitre **Synchronisation**.

EDITEUR DE BOÎTES DE DIALOGUE


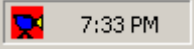


Dans l'Editeur WinTask, un éditeur de boîtes de dialogue est fourni : le script d'automatisation peut ainsi faire afficher de telles boîtes lors de l'exécution d'un script. Les informations saisies dans la boîte sont alors transmises au script en cours d'exécution. La création et l'appel de boîtes de dialogue sont expliqués au chapitre **Boîtes de dialogue**.

Mode Enregistrement

Pour automatiser rapidement des actions dans un site Web, le mode Enregistrement de WinTask permet d'enregistrer ses saisies clavier et ses clics sur des liens dans un script. Le mode Enregistrement est accessible depuis l'Editeur. Une fois le script généré, il est enregistré et son exécution peut être lancée.

Exercice 1

Cet exercice montre comment enregistrer les actions utilisateur sur des pages Web. Les pages Web à automatiser sont sous www.wintask.fr/demos.

1. Démarrez l'Editeur WinTask en cliquant sur le bouton **Démarrer** de Windows, puis en sélectionnant **WinTask** et dans le groupe WinTask en cliquant **WinTask**. Si l'Assistant de création de script s'affiche, cliquez sur le bouton **Annuler**.
2. La fenêtre de l'Editeur s'affiche, cliquez sur l'icône **Enreg**  de la barre d'outils.
3. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** ou **Mozilla Firefox** suivant le navigateur que vous préférez utiliser, et cliquez sur le bouton **OK**.
4. La boîte de dialogue **Démarrez Internet Explorer** ou **Démarrer Mozilla Firefox** s'affiche. Dans le champ **Adresse Web**, tapez **www.wintask.fr/demos** et cliquez sur le bouton **OK**.
5. La page de titre **Pages Démonstration WinTask** s'ouvre, la fenêtre de l'Editeur est minimisée et la **barre d'outils WinTask** s'affiche. Une icône clignotante  est également affichée dans la zone de notification de la barre d'état de Windows pour rappeler que le mode Enregistrement est actif. Cliquez sur le lien **Page 2**.
6. La page de titre **Page 2** s'affiche. Dans le champ **Votre Société**, tapez **Test WinTask** puis cliquez sur le lien **ici** de la phrase **et cliquez ici** pour revenir à la page d'accueil.
7. La page de titre **Pages Démonstration WinTask** s'affiche à nouveau, fermez la fenêtre Internet Explorer ou Firefox en cliquant sur l'icône de fermeture  en haut à droite de la fenêtre.
8. Arrêtez le **Mode Enregistrement** en cliquant sur l'icône  première icône de la barre d'outils flottante de WinTask.

9. La fenêtre de l'Editeur revient au premier plan et le script généré par le mode Enregistrement est affiché.



10. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script. Une boîte de dialogue vous demande de donner un nom à ce script, donnez le nom **scriptweb01** (l'extension .SRC est ajoutée automatiquement). Cliquez sur le bouton **Enregistrer** après avoir donné un nom. Vous voyez alors toutes vos actions se rejouer.
11. Quittez WinTask en sélectionnant dans la fenêtre de l'Editeur le menu **Fichier/Quitter**.

Mythe des outils de Capture/Rejoue

A première vue, le mode Enregistrement apparaît suffisant pour automatiser des tâches dans des pages Web. Mais le rejoue ne se fait pas aussi facilement dès lors que l'environnement Windows varie.

Sans des ajustements manuels, un script généré automatiquement par le mode Enregistrement ne se rejouera correctement que si l'environnement est exactement identique à celui lors de l'enregistrement des actions. En particulier, si le script démarre le navigateur, n'oubliez pas de refermer ce navigateur ouvert avant de rejouer le script.

Un Script simple d'automatisation

Dans ce paragraphe, nous allons examiner la signification des différentes lignes générées dans l'exercice 1, le script *scriptweb01.src*. Les lignes affichées dans le fichier script sont en italiques et sont suivies d'une explication.

L'utilisateur peut visualiser le fichier script dans la fenêtre de l'Editeur WinTask. En mettant le curseur texte sous le nom d'une instruction et en appuyant sur la touche F1, l'aide pour cette instruction s'affiche.

StartBrowser("IE", "www.wintask.fr/demos", 3)

StartBrowser démarre Internet Explorer et charge l'adresse Web spécifiée en deuxième paramètre. Le premier paramètre "IE" indique que le navigateur à utiliser est Internet Explorer. Le troisième paramètre 3 ouvre le navigateur dans une fenêtre maximisée.

OU

StartBrowser("FF", "www.wintask.fr/demos", 3)

StartBrowser démarre Mozilla Firefox et charge l'adresse Web spécifiée en deuxième paramètre. Le premier paramètre "FF" indique que le navigateur à utiliser est Mozilla Firefox. Le troisième paramètre 3 ouvre le navigateur dans une fenêtre maximisée.

UsePage("Pages Démonstration WinTask")

UsePage spécifie la page Web cible vers laquelle les actions suivantes dans le script sont envoyées. Cette instruction attend que la page Web soit complètement chargée avant de continuer. Le délai d'attente par défaut est de 30 secondes. Si au bout de 30 secondes, la page Web spécifiée n'a pas fini de se charger, une erreur d'exécution WinTask est affichée. Cette ligne attend donc que la page de titre "Pages Démonstration WinTask" soit chargée.

ClickHTMLElement("A[INNERTEXT= 'Page 2']")

ClickHTMLElement clique sur le lien spécifié dans le paramètre de l'instruction. L'objet HTML sur lequel cliquer est défini par son descripteur HTML. Le mode Enregistrement génère automatiquement ce descripteur qui utilise le plus souvent le texte du lien sur lequel cliquer. Le descripteur HTML est une chaîne de caractères (du texte) spécifiant de manière unique sur la page courante, un objet HTML affiché sur cette page. Utilisez toujours le mode Enregistrement pour générer les descripteurs HTML.

UsePage("Page 2")

Cette instruction *UsePage* indique que les prochaines actions doivent maintenant être envoyées à la page de titre "Page 2".

WriteHTML("INPUT TEXT[NAME= 'societe']", "Test WinTask")

WriteHTML écrit dans un champ de formulaire Web. Le nom du champ dans lequel écrire est spécifié comme premier paramètre de l'instruction, c'est le descripteur HTML pour ce champ. Le deuxième paramètre indique la chaîne de caractères (le texte) à saisir dans ce champ.

ClickHTMLElement("A[INNERTEXT= 'ici']")

ClickHTMLElement clique sur le lien spécifié dans le paramètre de l'instruction. L'objet HTML sur lequel cliquer est défini par son descripteur HTML. Le mode Enregistrement génère automatiquement ce descripteur qui utilise le plus souvent le texte du lien sur lequel cliquer, dans cette instruction, c'est le texte **ici**.

CloseWindow("IEXPLORE.EXE|IEFrame|Pages Démonstration WinTask - Windows Internet Explorer", 1)

CloseWindow ferme la fenêtre Internet Explorer spécifiée. A nouveau, le mode Enregistrement génère le nom de fenêtre, paramètre pour cette instruction *CloseWindow*. Le nom d'une fenêtre peut être trouvé également en utilisant l'outil Espion. Cette ligne ferme donc la fenêtre Internet Explorer qui affichait la page Pages Démonstration WinTask. Sous Internet Explorer 6, la fin du nom de la fenêtre est *Microsoft Internet Explorer* (au lieu de *Windows Internet Explorer* si vous utilisez Internet Explorer 7 ou 8 ou 9).

OU

CloseWindow("FIREFOX.EXE/MozillaWindowClass/Pages Démonstration WinTask - Mozilla Firefox", 1)

CloseWindow ferme la fenêtre Mozilla Firefox spécifiée.

Editeur WinTask

L'Editeur de WinTask fournit un environnement de développement complet pour faciliter la création et la mise au point des scripts d'automatisation. A partir de l'Editeur, l'utilisateur peut enregistrer des actions, modifier les lignes générées, compiler le code et exécuter le script. Une fenêtre de résultats donne les messages de compilation, une autre fenêtre liste les instructions disponibles dans le langage. Une aide contextuelle sur chaque instruction est accessible en appuyant sur la touche F1 quand le curseur est sous l'instruction désirée, ou en double cliquant sur le nom de l'instruction dans la fenêtre du langage affichée à droite (l'appui sur la touche F4 fait apparaître cette fenêtre listant toutes les instructions si elle n'est pas affichée). Si plusieurs scripts sont ouverts simultanément, un clic sur l'onglet du script désiré met en premier plan celui-ci.

Exercice 2

Cet exercice illustre les possibilités de l'environnement de développement, l'Editeur WinTask.

1. Démarrez l'Editeur WinTask. Si l'Assistant de création de script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Si WinTask ouvre le script réalisé avant (**scriptweb01.src**), passez à l'étape 2, sinon cliquez sur l'icône **Ouvrir** de la barre d'outils et ouvrez le script **scriptweb01.src**.
2. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **scriptweb02.src**. Cette étape permet de conserver le script **scriptweb01.src** tel qu'il était à l'exercice 1.
3. Dans la fenêtre de l'Editeur, mettez le curseur texte à la fin de la ligne contenant l'instruction *WriteHTML* puis appuyez sur la touche **Entrée**.
4. Dans la ligne blanche ainsi générée, écrivez : **MsgBox(Bonjour Testeur")** et appuyez sur la touche **Entrée** (tapez le texte exactement comme indiqué).
5. Mettez le curseur texte en fin de dernière ligne du script et appuyez sur la touche **Entrée**. Dans cette ligne blanche, écrivez **MsgBox("Testeur : fin du script")**.
6. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script.
7. Le Compilateur WinTask détecte des erreurs et écrit les messages d'erreur dans la fenêtre Résultats de l'Editeur. Double-cliquez sur le premier message d'erreur dans cette fenêtre Résultats, le curseur vient se positionner sur la ligne qui a provoqué l'erreur.

8. Mettez en commentaire cette ligne qui a provoqué l'erreur en insérant au tout début de la ligne le caractère ' (simple quote). Notez que le texte de cette ligne passe en vert pour indiquer que c'est un commentaire. Vous remarquez également que le nom des instructions est en bleu.
9. Mettez maintenant le curseur texte au début de la première ligne du script. Sélectionnez le menu **Edition/Remplacer**. Utilisez la boîte de dialogue **Remplacer** pour remplacer le mot **Testeur** par le mot **Développeur** dans tout le script.
10. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script. A la fin de l'exécution, la boîte de dialogue **Développeur : fin du script** s'affiche. Cliquez sur le bouton **OK**.
11. Revenez à la fenêtre de l'Editeur WinTask et sur la ligne mise en commentaire, supprimez le caractère ' de début et mettez le curseur texte sous le mot **MsgBox**, puis appuyez sur la touche **F1**.
12. L'aide pour l'instruction MsgBox s'affiche, corrigez la syntaxe de l'instruction pour qu'il n'y ait plus d'erreur de compilation. Notez le changement de couleur du texte passé en paramètre à l'instruction MsgBox une fois la syntaxe correcte. Fermez la fenêtre de l'aide.
13. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script. Cliquez sur le bouton **OK** lorsque la première boîte de dialogue s'affiche, cliquez à nouveau sur **OK** quand **Développeur : fin du script** s'affiche.

Arrêter l'exécution d'un script

Pour arrêter un script en cours d'exécution, il faut appuyer sur les touches **Ctrl+Shift+Pause**, le script s'arrête une fois la ligne en cours d'exécution terminée.

Synchronisation

Lors de l'exécution d'un script d'automatisation de pages Web, WinTask doit attendre que la page soit complètement chargée avant d'exécuter l'action suivante. Dans les exercices précédents, le script devait attendre que la page de titre Page 2 soit complètement chargée avant de saisir du texte dans le champ Saisissez votre nom. Les techniques de synchronisation sont au coeur d'un rejeu fiable d'un script d'automatisation.

WinTask effectue cette synchronisation via ses propres instructions qui attendent que l'objet HTML spécifié soit affiché et prêt à recevoir une action avant de poursuivre. Par exemple l'instruction UsePage attend que la page dont le titre est spécifié soit bien complètement chargée avant de continuer. Et une erreur d'exécution est renvoyée si la page spécifiée n'a pas fini de se charger au bout d'un certain laps de temps (ce timeout est à 30 secondes par défaut). Les instructions ClickHTMLElement, WriteHTML, ... attendent également que l'objet HTML spécifié soit prêt avant d'effectuer l'action (clic sur le lien, saisie dans le champ,...).

Il existe un certain nombre de cas où la synchronisation automatique via un UsePage n'est pas suffisante. Sur les exemples simples de pages Web sous www.wintask.fr/demos, quelques cas simples sont traités dans ce chapitre.





Synchronisation quand les pages du site ont le même titre

Exercice 3

Cet exercice illustre comment effectuer une synchronisation quand les pages Web à automatiser ont le même titre. La solution de l'exercice est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de création de script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Si WinTask ouvre un script réalisé avant (par exemple **scriptweb02.src**), fermez-le en sélectionnant le menu **Fichier/Fermer** puis sélectionnez le menu **Fichier/Nouveau** ou cliquez sur l'icône **Nouveau** de la barre d'outils.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement.
3. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** ou **Mozilla Firefox** suivant le navigateur que vous préférez utiliser, et cliquez sur le bouton **OK**.
4. La boîte de dialogue **Démarez Internet Explorer** ou **Démarrer Mozilla Firefox** s'affiche. Dans le champ **Adresse Web**, tapez

www.wintask.fr/demos/pageidentique1.htm et cliquez sur le bouton **OK**.

5. La page de titre **Page Titre identique** s'ouvre, la fenêtre de l'Editeur est minimisée et la **barre d'outils WinTask** s'affiche. Cliquez sur le mot **ici** dans la phrase Cliquez ici....
6. La page de titre **Page Titre identique** s'ouvre, elle porte le même titre que la précédente.
7. Cliquez sur le lien **page précédente** affiché dans la phrase Revenez à la page précédente. La page précédente de même titre s'affiche. Saisissez votre nom dans le champ Votre Nom et cliquez dans le champ Votre Société sans rien saisir.
8. Fermez la fenêtre du navigateur and cliquant sur l'icône de fermeture en haut à droite de la fenêtre .
9. Arrêtez le **Mode Enregistrement** en cliquant sur l'icône  première icône de la barre d'outils flottante de WinTask.
10. La fenêtre de l'Editeur revient au premier plan et le script généré par le mode Enregistrement est affiché.
11. Cliquez sur l'icône **Exéc**  de la barre d'outils pour lancer l'exécution du script. Une boîte de dialogue vous demande de donner un nom à ce script. donnez le nom **scriptweb03** et cliquez sur le bouton **Enregistrer**. Les résultats de la **Compilation** sont affichées dans la fenêtre Résultats, fenêtre du bas de l'Editeur, puis l'exécution du script compilé démarre.
12. Le script tel quel a peu de chances de se synchroniser correctement car les instructions UsePage ne sont pas créées par le mode Enregistrement si les pages ont le même titre.
13. Ajoutez manuellement les instructions UsePage aux bons endroits pour garantir une synchronisation correcte.
14. Cliquez sur l'icône **Exéc**  de la barre d'outils pour lancer l'exécution du script et constater désormais le rejoue correct.

Synchronisation manuelle

WinTask inclut des méthodes de synchronisation manuelle, synchronisations pouvant être ajoutées pendant le mode Enregistrement si une instruction UsePage ou les instructions ClickHTML, WriteHTML, ... ne suffisent pas. Un assistant est disponible pour chaque méthode de synchronisation. Si cet assistant est appelé de l'Editeur, le code généré est inséré là où le curseur texte se trouve. Si cet assistant est appelé via la barre d'outils flottante en mode Enregistrement, le code est inséré au bon endroit de la séquence des actions enregistrées.

L'aide WinTask fournit des indications supplémentaires sur chaque paramètre disponible dans ces méthodes de synchronisation. Consultez l'appendice A pour la signification des différents icônes de synchronisation dans la barre d'outils de l'Editeur. Consultez l'appendice B pour les différents icônes de synchronisation dans la barre d'outils flottante WinTask en mode Enregistrement.

- **Synchronisation sur Texte**

Cette synchronisation attend qu'un certain texte apparaisse quelque part dans une fenêtre Windows. Une telle synchronisation doit être utilisée par exemple dans un logiciel d'émulation i-series (émulation 5250) pour attendre qu'un écran du système hôte a bien fini de se charger avant d'effectuer des saisies dans le nouvel écran. Si le texte n'utilise pas une police standard Windows, le texte peut ne pas être reconnu par WinTask. Si c'est le cas, il faut utiliser la Synchronisation sur Texte OCR. L'assistant de synchronisation texte est appelé par le menu **Insérer/Synchronisation/Sur Texte**.

- **Synchronisation sur Texte OCR**

Cette synchronisation attend qu'un certain texte affiché dans un graphique apparaisse quelque part dans une fenêtre. La reconnaissance de texte OCR n'est pas immédiate, le temps nécessaire à la reconnaissance OCR varie de quelques dixièmes de secondes à 3 secondes suivant la quantité de texte à reconnaître. Donc cette synchronisation ne doit être utilisée que si ni un UsePage, un ClickHTML, un WriteHTML, ... ne conviennent pour assurer la synchronisation. L'assistant de synchronisation texte OCR est appelé par le menu **Insérer/Synchronisation/Sur Texte OCR**.

- **Synchronisation sur Image**

Cette synchronisation attend qu'une image (un bitmap) soit affichée quelque part dans une fenêtre. Cette méthode de synchronisation est par exemple utilisée sur des sites Flash où il n'est pas possible d'utiliser la synchronisation sur texte OCR (par exemple quand les icônes du site ne contiennent pas de texte). La synchronisation sur Image n'est à utiliser que si les synchronisations précédentes sont inopérantes, car le bitmap généré dépend de la résolution de l'écran qui peut varier d'un PC à un autre. L'assistant de synchronisation sur Image est appelé par le menu **Insérer/Synchronisation/Sur Image**.

- **Synchronisation sur Fenêtre**

Cette synchronisation attend qu'une fenêtre devienne active ou qu'une fenêtre disparaisse. Cette synchronisation est utilisée par exemple pour attendre qu'un fichier téléchargé d'un site Web ait fini de se télécharger. La synchronisation sur fenêtre permet d'attendre que la fenêtre de téléchargement ait disparu avant de continuer. L'assistant de synchronisation sur Fenêtre est appelé par le menu **Insérer/Synchronisation/Sur Fenêtre**.

- **Synchronisation sur Durée**

C'est la méthode de synchronisation la plus simple, le script attend un intervalle de temps fixe avant de continuer. Cette durée peut s'exprimer en ticks (environ 1/100 sec), en secondes, minutes, heures, jours. Cette méthode de synchronisation est par exemple utilisée sur les formulaires Ajax où le contenu d'un champ dépend de l'action sur le champ précédent et il faut un peu de temps

pour que le champ se remplisse des données correctes. L'assistant de synchronisation sur Durée est appelé par le menu **Insérer/Synchronisation/Sur Durée**.

- **Synchronisation sur Date/Heure**


Cette synchronisation attend qu'il soit la date ou l'heure spécifiée. Cette méthode de synchronisation n'est plus guère utilisée, il est préférable d'utiliser le Planificateur WinTask pour lancer un script par exemple tous les jours à la même heure. L'assistant de synchronisation sur Date/Heure est appelé par le menu **Insérer/Synchronisation/Sur Date/Heure**.

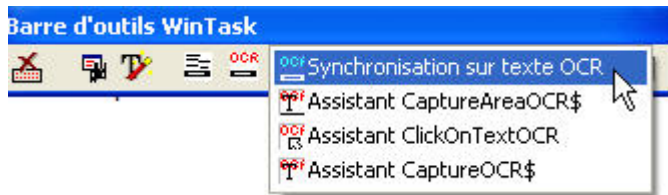
- **Attente action clavier, action menu, action souris**

Ces attentes ne sont pas utilisées dans des scripts d'automatisation Web.

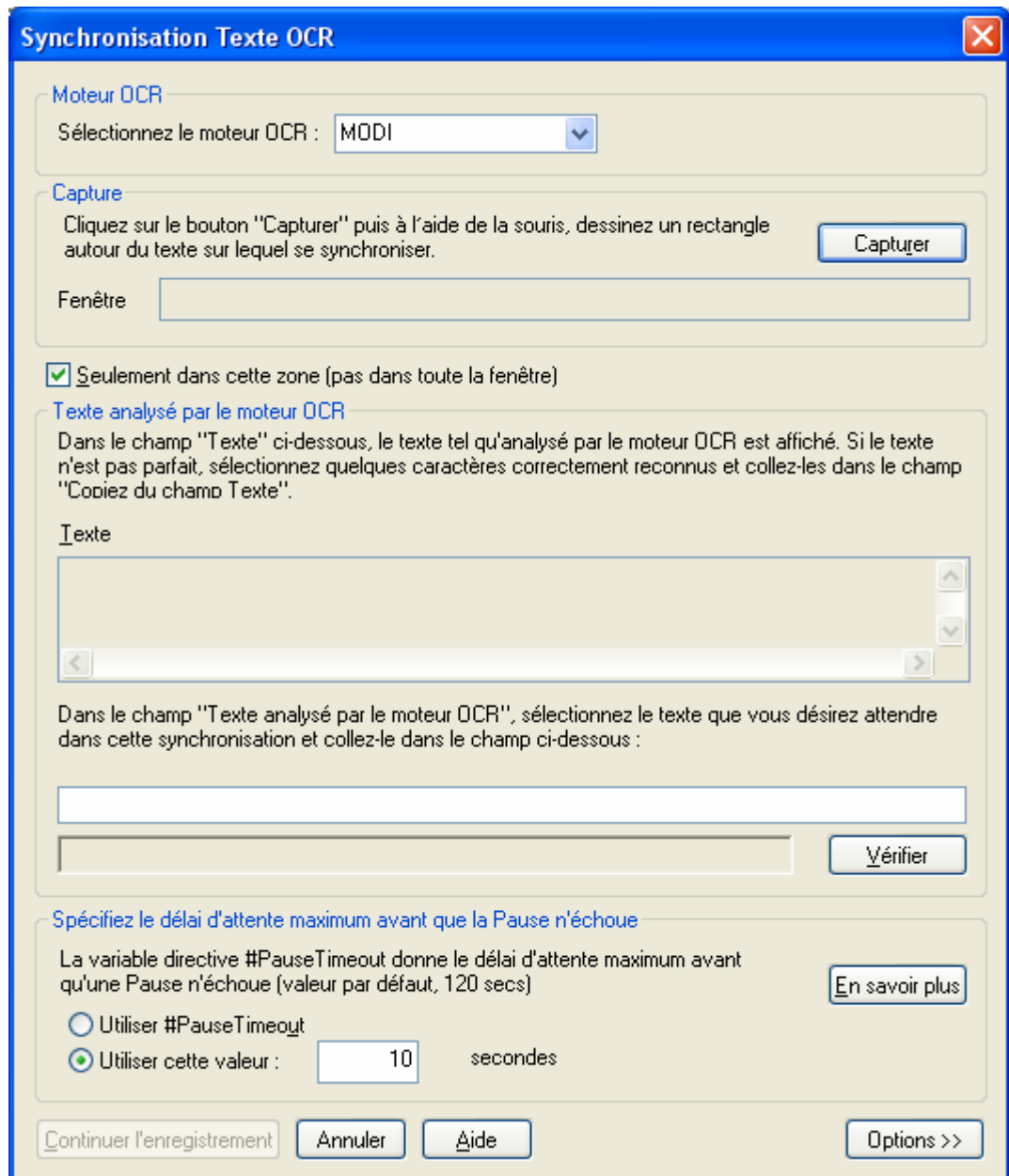
Exercice 4 : Synchronisation sur texte OCR

Cet exercice utilise la synchronisation sur texte OCR pour attendre qu'une page ait fini de se charger. La solution de l'exercice est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de création de script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Si WinTask ouvre un script réalisé avant (par exemple **scriptweb03.src**), fermez-le en sélectionnant le menu **Fichier/Fermer** puis sélectionnez le menu **Fichier/Nouveau** ou cliquez sur l'icône **Nouveau** de la barre d'outils.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement.
3. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** ou **Mozilla Firefox** suivant le navigateur que vous désirez utiliser et cliquez sur le bouton **OK**.
4. La boîte de dialogue **Démarrer Internet Explorer** ou **Démarrer Mozilla Firefox** s'affiche. Dans le champ **Adresse Web**, tapez **www.wintask.fr/demos/pageidentique1.htm** et cliquez sur le bouton **OK**.
5. La page de titre **Page Titre identique** s'ouvre, la fenêtre de l'Editeur est minimisée et la **barre d'outils WinTask** s'affiche. Cliquez sur le mot **ici** dans la phrase Cliquez ici...
6. La page de titre **Page Titre identique** s'ouvre, elle porte le même titre que la précédente et il faut être sûr que le texte de la page a bien fini de s'afficher avant de capturer une information sur cette page.
7. Appelez l'Assistant de synchronisation sur Texte OCR en cliquant dans la barre d'outils WinTask flottante sur  , puis sur Synchronisation sur texte OCR



Le mode Enregistrement est momentanément arrêté.



8. Le champ **Sélectionnez le moteur OCR** permet de choisir le moteur OCR WinTask ou celui fourni avec Office 2003 ou Office 2007, moteur appelé MODI (Microsoft Office Document Imaging). Ce dernier est plus puissant mais il faut une de ces versions d'Office.
9. Cliquez sur le bouton **Capturer** de l'écran Synchronisation Texte OCR. La boîte de dialogue se met en réduction et le curseur souris devient une croix.

10. Entourez à l'aide de ce curseur le texte **chargement de cette nouvelle page** situé dans le premier paragraphe de la page. Le texte capturé est maintenant dans le champ **Texte analysé par le moteur OCR** de la boîte de dialogue Synchronisation Texte OCR

Synchronisation Texte OCR

Moteur OCR
Sélectionnez le moteur OCR : WinTask

Capture
Cliquez sur le bouton "Capturer" puis à l'aide de la souris, dessinez un rectangle autour du texte sur lequel se synchroniser. **Capturer**
Fenêtre "IEXPLORE.EXE|Internet Explorer_Server|Page Titre identique - Windows Internet Explore"

Seulement dans cette zone (pas dans toute la fenêtre)

Texte analysé par le moteur OCR
Dans le champ "Texte" ci-dessous, le texte tel qu'analysé par le moteur OCR est affiché. Si le texte n'est pas parfait, sélectionnez quelques caractères correctement reconnus et collez-les dans le champ "Copiez du champ Texte".

Texte
ch argemen t de cette n ouve e page

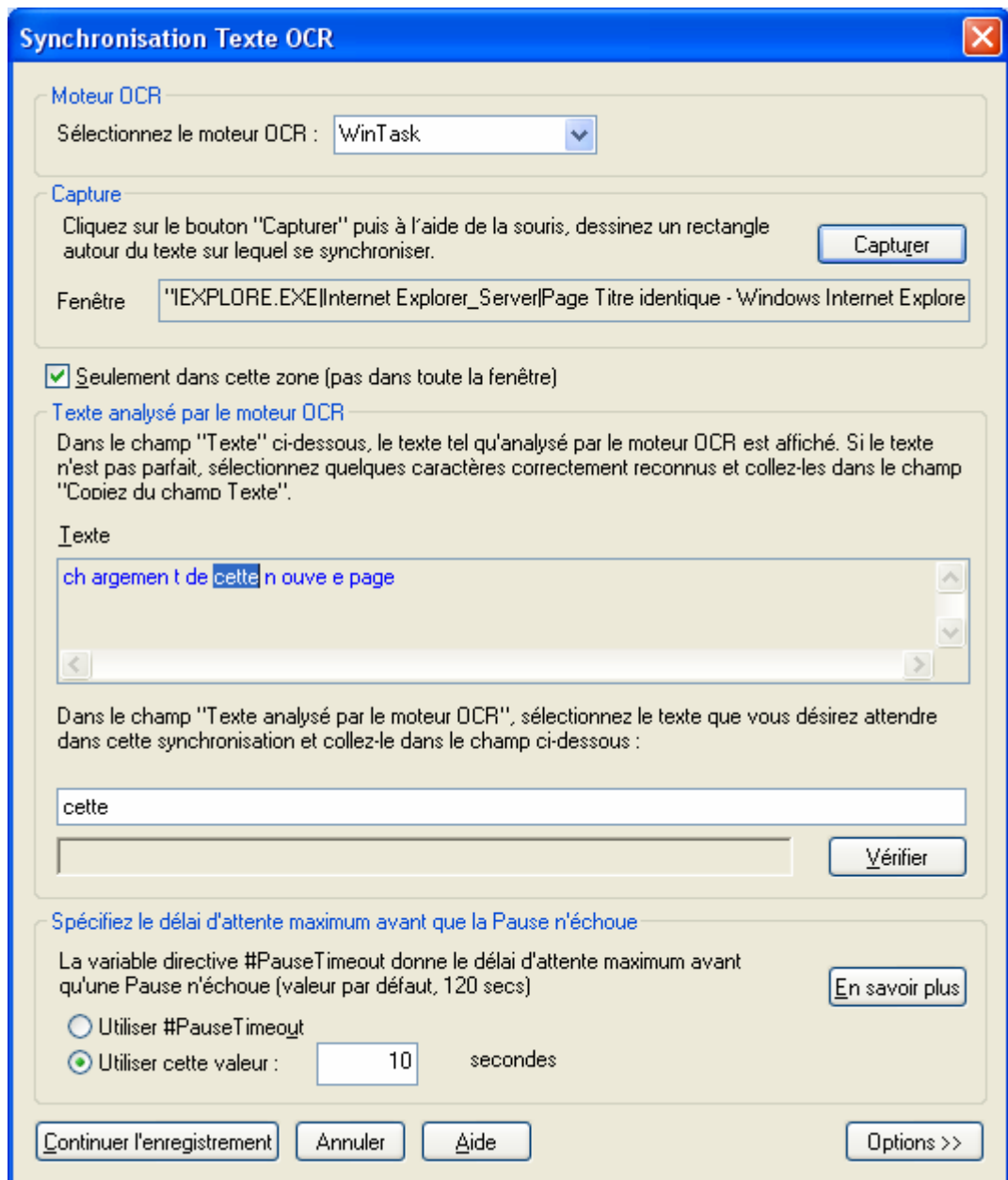
Dans le champ "Texte analysé par le moteur OCR", sélectionnez le texte que vous désirez attendre dans cette synchronisation et collez-le dans le champ ci-dessous :


 Vérifier

Spécifiez le délai d'attente maximum avant que la Pause n'échoue
La variable directive #PauseTimeout donne le délai d'attente maximum avant qu'une Pause n'échoue (valeur par défaut, 120 secs) **En savoir plus**

Utiliser #PauseTimeout
 Utiliser cette valeur : 10 secondes

11. Vous voyez que le moteur OCR ne reconnaît pas tous les mots correctement, vous allez donc prendre pour la synchronisation du texte reconnu correctement, par exemple le texte **cette**. Si vous avez sélectionné le moteur MODI, le texte est reconnu parfaitement mais ne prenez quand même en compte que le mot **cette** pour une reconnaissance OCR plus rapide.
12. Sélectionnez et copiez ce texte, puis collez-le dans le champ en dessous. Vous devez avoir une boîte de dialogue Synchronisation Texte OCR ainsi :



13. Cliquez sur le bouton **Vérifier** pour vérifier qu'au rejeu le texte est reconnu correctement par le moteur OCR. Un message s'affiche indiquant que le texte est reconnu correctement.
14. Cliquez sur le bouton **Continuer l'enregistrement**. Le mode Enregistrement redevient actif.
15. Cliquez sur le lien **page précédente** dans la phrase Revenez à la page précédente.
16. Fermez la fenêtre du navigateur en cliquant sur l'icône de fermeture en haut à droite de la fenêtre .
17. Arrêtez le mode Enregistrement en cliquant sur la première icône de la barre d'outils WinTask flottante.

L'Assistant de synchronisation sur Texte OCR a inséré ces lignes dans le code généré par le mode Enregistrement :

Avec Internet Explorer :

```
ret = UseOCREngine(2)
Pause Until
  TextOCR("cette")
  InWindow("IEXPLORE.EXE|Internet Explorer_Server| Page Titre identique -
    Windows Internet Explorer|1",2)
  InArea(387,181,23,229)
PauseFalse
  MsgBox("L'attente à la ligne " + #ErrorLine$ + " a échoué !")
End
EndPause
```

Avec Firefox :

```
ret = UseOCREngine(2)
Pause Until
  TextOCR("cette")
  InWindow("FIREFOX.EXE|MozillaWindowClass|Page Titre identique - Mozilla
    Firefox",1))
  InArea(387,181,23,229)
PauseFalse
  MsgBox("L'attente à la ligne " + #ErrorLine$ + " a échoué !")
End
EndPause
```

La ligne Pause Until attend la valeur par défaut de 120 secondes avant de renvoyer une erreur si le texte n'est pas trouvé. Modifiez cette ligne pour autoriser une attente maximum de 20 secondes. Modifiez également le message d'erreur spécifié dans la ligne MsgBox pour faire afficher le message **La page ne s'est pas chargée en moins de 20 secondes !**

Si vous utilisez Internet Explorer 9, remplacez le chiffre 2 par le chiffre 1 dans la ligne InWindow (le chiffre juste avant la parenthèse fermante). Le bloc de lignes devient donc :

```
ret = UseOCREngine(2)
Pause 20 secs Until
  TextOCR("cette")
  InWindow("IEXPLORE.EXE|Internet Explorer_Server| Page Titre identique -
    Windows Internet Explorer|1",1)
  InArea(387,181,23,229)
PauseFalse
  MsgBox("La page ne s'est pas chargée en moins de 20 secondes !")
End
EndPause
```

18. Pour bien constater la synchronisation texte lors du rejoue, ajoutez la ligne *msgbox("le texte de la page est maintenant affiché")* juste après la ligne *EndPause*.



19. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script. Une boîte de dialogue vous demande de donner un nom à ce script. donnez le nom **scriptweb04** et cliquez sur le bouton **Enregistrer**. Les résultats de la **Compilation** sont affichées dans la fenêtre Résultats, fenêtre du bas de l'Editeur, puis l'exécution du script compilé démarre. Le texte attendu dans la page s'affiche, la boîte de dialogue avec le message s'affiche, cliquez sur le bouton **OK**.

Les autres assistants de synchronisation sont similaires et leur appel se fait également en cliquant sur leur icône dans la barre d'outils WinTask en mode Enregistrement.

Le langage WinTask

Le langage de programmation de WinTask est voisin de Visual Basic™ de Microsoft. Les scripts d'automatisation doivent respecter une certaine structure et la syntaxe des instructions pour éviter des erreurs de compilation. Les scripts sont composés de trois sections distinctes. La première comporte la déclaration des tableaux de données utilisés dans le script, la deuxième liste les fonctions et sous-routines définies par l'utilisateur et la troisième est le programme principal appelant les tableaux et les fonctions déclarées dans les sections précédentes. Le langage WinTask peut manipuler des entiers, des chaînes de caractères, des tableaux d'entiers ou de chaînes à une seule dimension, et des entiers de type Unsigned (pour des appels à des fonctions internes de Windows).

Les scripts d'automatisation créés par le mode Enregistrement génèrent la partie programme principal. Ensuite, comme dans tout langage de programmation, des fonctions ou des sous-routines peuvent être écrites pour traiter les mêmes actions par simple appel d'une fonction, si ces actions à effectuer reviennent plusieurs fois dans le programme principal.

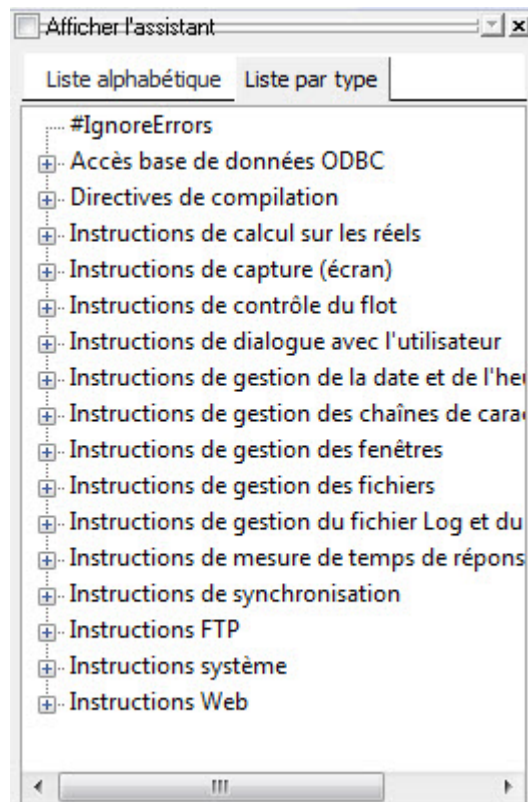
Instructions du langage

Les instructions du langage peuvent avoir des paramètres et peuvent retourner un code retour ou une variable résultat. Si l'instruction a des paramètres, la liste de ceux-ci est mise entre parenthèses juste après le nom de l'instruction et les différents paramètres sont séparés par des virgules. Si une constante de type chaîne de caractères est passée en paramètre à une instruction, la chaîne doit être mise entre double quotes (par exemple "Bonjour"). Les instructions renvoyant une variable résultat de type chaîne de caractères ont leur nom se terminant par le caractère \$. Les instructions renvoyant un entier NE SE TERMINE PAS par le caractère \$. Voici un exemple de syntaxe :

```
StartBrowser("IE", "www.wintask.fr",3)
```

La fenêtre listant toutes les instructions du langage est dans la partie droite de

l'Editeur. Si elle n'est pas affichée, cliquez sur l'icône **Langage** de la barre d'outils de l'Editeur ou appuyez sur la touche **F4**. Les instructions sont classées par type en arborescence classique. En double cliquant une instruction, l'aide pour cette instruction s'affiche ou un assistant est proposé. Un clic droit sur une instruction affiche également un menu contextuel. Si vous cochez Afficher l'assistant, un double clic sur une instruction fait afficher une boîte de dialogue où les paramètres pour cette instruction sont à saisir. Le chapitre Instructions Web liste en particulier toutes les instructions utiles pour l'automatisation de sites Web.



N'hésitez pas à utiliser cette fenêtre Langage pour accéder aux différentes instructions de WinTask.

Variables

L'utilisation de variables dans un script WinTask permet de stocker en mémoire une donnée pour ensuite la réutiliser plus loin. Une variable n'a pas besoin d'être déclarée en début de script, sauf si c'est une variable de type tableau. Un nom de variable ne peut pas comporter de caractères accentués. Dans cette ligne de script :

```
Read("monfichier.txt", ligne$, crlf)
```

L'instruction *Read* ouvre le fichier *monfichier.txt*, lit la première ligne du fichier et stocke le contenu de cette ligne dans la variable de type chaîne de caractères *ligne\$*. Le script peut ensuite extraire l'information désirée du contenu de la variable *ligne\$*. Comme pour les instructions retournant une chaîne de caractères, les variables de type chaîne de caractères doivent avoir un nom se terminant par le caractère **\$**.

Voici un autre exemple d'utilisation de variables :

```
url$ = "http://www.wintask.fr"  
StartBrowser("IE",url$)
```

```
StartBrowser("IE","http://www.wintask.fr")
```

Les deux premières lignes sont équivalentes à la troisième. Si le script doit souvent faire appel à l'adresse Web *http://www.wintask.fr*, il est préférable d'utiliser une variable et ensuite d'y faire appel comme dans : *StartBrowser("IE",url\$)*.

Variables directives

WinTask dispose d'un ensemble de variables directives afin de paramétrer la manière dont s'exécute un script. Par exemple, si un site Web met beaucoup de temps à se charger, le timeout par défaut peut être modifié par une variable directive afin que la première page du site ait le temps de se charger sans que WinTask n'indique une erreur.

Toutes les variables directives commencent par le caractère **#** . Dans la fenêtre Langage de l'Editeur, double-cliquez sur la tête de chapitre Toutes les instructions, vous voyez en début de liste toutes les variables directives disponibles. Voici un exemple :

```
#IgnoreErrors = 1
```

Entiers

Dans WinTask, les entiers peuvent prendre des valeurs entières de -2 147 483 648 à +2 147 483 647. Une variable de type entier NE DOIT PAS se terminer par le caractère **\$**.

Chaînes de caractères

Une variable de type chaîne de caractères doit se terminer par le caractère **\$**. Une chaîne vide est `""`, une chaîne contenant le caractère Espace est `" "`. Voici un exemple

d'assignation d'une chaîne de caractères à une variable de type chaîne :

```
Formulepolitesses$ = "Nous vous prions d'agr er, Madame, Monsieur, l'expression  
de nos sentiments distingu s"
```

Nombres r els

Ce type n'est pas support  par WinTask. Les nombres r els sont stock s sous la forme d'une cha ne de caract res et les instructions **Add\$**, **Divide\$**, **Multiply\$** et **Subtract\$** permettent d'effectuer des op rations sur ces cha nes repr esentant des nombres r els. Voici un exemple :

```
Reel1$ = "135,46"  
Reel2$ = "-10,46"  
Resultat$=Add$(Reel1$,Reel2$)  
Msgbox(Resultat$)
```

Tableaux

Seuls les tableaux   une dimension sont accept s dans la syntaxe WinTask. Ils doivent  tre d clar s en tout d but de script en utilisant l'instruction Dim. La taille maximum d'un tableau est de 65535  l ments. Le premier  l ment d'un tableau est   l'index 0. Voici un exemple de d claration de tableau de type entier ayant 1000  l ments :

```
Dim Data(1000)
```

Voici un exemple de d claration de tableau de type cha ne de caract res :

```
Dim Data$(1000)
```

Op rateurs

Les op rateurs support s dans WinTask sont ceux classiques de tout langage, assignation, op rateurs arithm tiques, op rateurs logiques et op rateur de concat nation de cha nes de caract res. Les op rateurs logiques sont utilis s dans les instructions conditionnelles telles **If**, **While** et **Repeat**. Voici la liste avec un exemple pour chacun :

= Assignation : Assigne une valeur   une variable

```
compteur = 38  
nom$ = "Olivier DUPONT"  
ville$(compteur)="Cherbourg"
```

+ Addition sur des entiers

```
colonne = ligne + 7
```

- Soustraction sur des entiers

```
Profit = Revenus - Depenses
```

* Multiplication

```
Doigts = Mains * 5
```

/ Division

Heures = Jours / 24

= Egalité logique

If (employees = 15) Then ... Endif

<> Non égal

If (employees <> 15) Then ... Endif

< Inférieur à

While (employees < 15) ... Wend

<= Inférieur ou égal

If (employees <= 15) Then ... Endif

> Plus grand que

Repeat ... Until (employees > 15)

>= Plus grand ou égal

If (employees >= 15) Then ... Endif

AND ET logique

If (employees = 15) AND (bureaux < 10) Then ... Endif

OR OU logique

If (employees <> bureaux) OR (employees <> ordinateurs) Then ... Endif

+ Concaténation de chaînes

Nom\$ = Prenom\$ + " " + Nomfamille\$

Titre\$ = "Le directeur est : " + Nom\$

Instructions de gestion des pages

WinTask inclut différents mécanismes de recherche d'une page lorsqu'à l'exécution l'automate doit trouver la page spécifiée dans le script. Les variables directives et les instructions les plus importantes gérant l'identification des pages sont décrites dans ce paragraphe.

#IgnoreErrors

La variable directive **#IgnoreErrors** contrôle le comportement de l'automate quand une erreur d'exécution est détectée. Par défaut, cette variable directive est à 0 : en cas d'erreur, un message d'erreur est affiché et l'exécution du script est arrêtée. En mettant cette variable directive à 1, le script continue son exécution en cas d'erreur. L'instruction ayant provoqué l'erreur renvoie un code retour qu'il est vivement recommandé de tester afin de gérer le cas d'erreur.

Voici un exemple où sur une portion du code, la valeur par défaut de **#IgnoreErrors** est modifiée :

```
#IgnoreErrors = 1
```

```
ret = StartBrowser("IE", "www.wintask.fr")
```

```
If (ret = 0) Then
```

```
    MsgBox("Le site WinTask a été lancé avec succès")
```

```
Else
```

```
    MsgBox("Impossible d'ouvrir le site WinTask !")
```

```
Endif
```

```
#IgnoreErrors = 0
```

Un message est affiché après avoir essayé de lancer le site Web **www.wintask.fr**. Le message dépend du code retour renvoyé par l'instruction **StartBrowser**.

#ActionTimeout

La variable directive **#ActionTimeout** contrôle le temps d'attente (timeout) maximum avant que WinTask ne retourne une erreur lors de l'exécution d'une ligne dans le script, telle l'identification d'une page ou d'un objet HTML. La valeur par défaut est de 30 secondes.

Voici un exemple où sur une portion du code, la valeur par défaut de **#ActionTimeout** est modifiée :

```
StartBrowser("IE", "www.wintask.fr/demos")
```

```
UsePage("Pages Démonstration WinTask")
```

```
ClickHTMLElement("A[INNERTEXT= 'Page 2']")
```

```
#ActionTimeout = 5
```

```
#IgnoreErrors = 1
```

```
ret=UsePage("Page 2")
```

```
If (ret = 0) Then
```

```
MsgBox("La page 2 s'est chargée en moins de 5 secondes")
```

```
Else
```

```
MsgBox("Achetez un ordinateur plus rapide !")
```

```
Endif
```

```
#ActionTimeout = 30
```

L'instruction UsePage renvoie une erreur si à l'exécution, la page spécifiée n'est pas trouvée au bout de 5 secondes. Pour que l'erreur n'arrête pas l'exécution du script, notez *#IgnoreErrors=1*. Le code retour de UsePage est ensuite testé et le message affiché dépend de la valeur du code retour.

#UsePageExact

La variable directive **#UsePageExact** contrôle l'algorithme d'identification d'une page lors du rejeu du script. Considérez ces deux lignes :

```
UsePage("Pages Démonstration WinTask")
```

```
UsePage("Page 2")
```

Au rejeu, les deux recherchent une page dont le titre commence par **Page**.

Le comportement par défaut de WinTask lors du rejeu est de rechercher d'abord une page dont le titre est exactement celui spécifié dans l'instruction *UsePage*. Si une page de titre exactement celui spécifié n'est pas trouvée, une recherche approchée est lancée : les caractères de droite du titre de la page sont tronqués pour essayer de trouver une page dont le titre a au moins le premier caractère identique à celui spécifié dans le script. Si malgré les caractères tronqués, aucune page de même titre n'est trouvée, alors une erreur d'exécution se produit.

Cette méthode d'identification des pages au rejeu peut être modifiée en mettant la variable directive **#UsePageExact** à la valeur 1 : dans ce cas, seule une page de titre exactement le même que celui spécifié dans le script sera trouvée. La valeur par défaut de **#UsePageExact** est 0 et donc dans ce cas, la recherche approchée se fait.

ASTUCE : La recherche approchée ralentit un peu l'exécution du script. Si le titre d'une page varie d'une exécution à l'autre, alors que les premiers caractères sont toujours les mêmes, vous pouvez tronquer manuellement dans le script le titre de la page. Dans l'exemple avec Page, cette ligne : *UsePage("Page)*, au rejeu, trouvera la page que ce soit la page *Pages Démonstration WinTask* ou la page *Page 2*.

ASTUCE: Si plusieurs pages d'une même site sont ouvertes en même temps et que le titre de chacune d'elles commence par les mêmes caractères, la recherche approchée peut entraîner une erreur d'identification lors du rejeu. Dans ce cas, forcez **#UsePageExact** à 1.

Itération

Le langage WinTask inclut deux instructions pour exécuter répétitivement un bloc de lignes du script. La construction While...Wend exécute les lignes entre ces deux instructions zéro ou n fois tant que la condition du While est vraie. La construction Repeat...Until exécute les lignes entre ces deux instructions une ou n fois tant que la condition du Until est vraie.

Itération

Voici un exemple des deux itérations avec respectivement **While/Wend** et **Repeat/Until**.

```
Debut = 0
Fin = 10
Index = Debut
While (Index < Fin)
    Lignes à exécuter index fois
    Index = Index + 1
Wend
```

```
Debut = 0
Fin = 10
Index = Debut
Repeat
    Lignes à exécuter index fois
    Index = Index + 1
Until (Index = Fin)
```

Exercice 5 : Itération cliquant la ième page

Cet exercice est un exercice de programmation où vous allez mettre en œuvre une itération. La première étape consiste à utiliser le mode Enregistrement pour générer les lignes de code à itérer, puis vous ajouterez des instructions du langage WinTask gérant les itérations. La solution de l'exercice est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de création de script s'affiche, cliquez sur le bouton **Annuler**. Le titre de la fenêtre de l'Editeur doit être **WinTask – [SansNom1]**. Si ce n'est pas le cas, cliquez sur l'icône **Nouveau** dans la barre d'outils.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** ou **Mozilla Firefox** suivant le navigateur que vous préférez utiliser et cliquez sur le bouton **OK**.
3. La boîte de dialogue **Démarez Internet Explorer** ou **Démarez Mozilla Firefox** s'affiche. Dans le champ **Adresse Web**, tapez

www.wintask.fr/demos/page-iteration1.htm et cliquez sur le bouton **OK**.

4. La page de titre **Page Itération 1** s'affiche. Sous la phrase Cliquez sur le numéro de page désiré, vous voyez des liens pour aller à la page 2, 3, 4, jusqu'à 7. Dans cet exercice, vous allez écrire un script qui fait afficher les pages 2 à 7.
5. Cliquez sur le chiffre **2**. La page de titre **Page Itération 2** s'affiche. Cliquez sur le mot *ici* dans la phrase Cliquez ici pour revenir à Page itération 1. La page de titre **Page Itération 1** s'affiche à nouveau.
6. Fermez la fenêtre du navigateur en cliquant sur l'icône de fermeture de la fenêtre.
7. Arrêtez le **Mode Enregistrement** en cliquant sur la première icône de la barre d'outils WinTask flottante.
8. Dans la fenêtre de l'Editeur, modifiez le script pour utiliser la construction *While/Wend* afin de saisir cliquer sur le liens 2 à 7 dans la page de titre WinTask – Page Itération 1. Deux lignes dans le code généré sont à modifier pour tenir compte de l'index :

```
ClickHTMLElement("A[INNERTEXT= '2']")
UsePage("Page Itération 2")
```

ASTUCE: Utilisez l'opérateur de concaténation de chaînes (+) pour concaténer le texte avant le chiffre 2 à l'index 2, 3, ... 7. Comme l'index est un entier, utilisez l'instruction Str\$ pour convertir l'entier index en chaîne.

9. Enregistrez le script sous le nom **scriptweb05a** et cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution. Si des erreurs de compilation sont affichées, corrigez-les et relancez l'exécution. Vous devez voir le script cliquer sur le chiffre 2, aller à la page 2, cliquer sur le lien Page d'accueil, revenir à la page 1, cliquer sur la page 3, etc.....
10. Ajoutez en début de script la ligne *#UsePageExact=1*. Pourquoi cette ligne permet de fiabiliser le rejoue ?
11. Modifiez le script en remplaçant le bloc *Repeat/Until* par un bloc *While/Wend*.
12. Enregistrez le script modifié sous le nom **scriptweb05b** et lancez l'exécution.

Vous savez désormais répéter les mêmes actions dans un script, sans avoir à dupliquer les lignes de code.

Instructions de gestion des fichiers

WinTask inclut des instructions permettant de lire ou d'écrire dans des fichiers texte, des fichiers Excel, des fichiers INI ou des fichiers XML. Ce paragraphe décrit les instructions de gestion des fichiers les plus couramment utilisées dans les scripts d'automatisation de sites Web. Pour une explication exhaustive, allez dans la fenêtre Langage de l'Editeur et double-cliquez sur la tête de chapitre Instructions de gestion des fichiers.

Exist()

L'instruction **Exist** permet de déterminer si un fichier existe ou pas lors du rejoue du script. Le nom de fichier peut être un nom complet incluant le chemin, un nom UNC, ou un nom de fichier à rechercher dans le répertoire courant. L'instruction renvoie un code retour qui peut ensuite être utilisé dans une instruction conditionnelle. Utilisez cette instruction pour éviter des erreurs d'exécution lors d'un appel à un fichier qui n'existe pas.

Voici un exemple :

```
NomFichier$ = "C:\Program Files\WinTask\Help\WinTask.chm"  
ret = Exist(NomFichier$)  
If ret = 1 Then  
    Shell("hh.exe " + Chr$(34) + NomFichier$ + Chr$(34))  
Else  
    MsgBox("Fichier non trouvé : " + NomFichier$)  
Endif
```

ASTUCE : Le nom de fichier passé à l'instruction **Exist** ne comporte pas de *Chr\$(34)* même si le chemin a des blancs, alors que les *CHR\$(34)* sont obligatoires dans l'instruction Shell car le blanc est un séparateur entre le nom de l'exé et le paramètre pour cet exe. *CHR\$(34)* est le symbole ASCII pour double-quote.

Kill()

L'instruction **Kill** supprime le fichier spécifié. Le nom de fichier peut être un nom complet incluant le chemin, un nom UNC, ou un nom de fichier à rechercher dans le répertoire courant. Si le fichier n'existe pas, l'exécution retourne une erreur, voici un exemple pour supprimer un fichier qu'il existe ou pas :

```

NomFichier$ = "C:\Temp\UnFichierInutile.txt "
#IgnoreErrors=1
ret = Kill(NomFichier$)
If ret = 0 Then
    MsgBox("Fichier : " + NomFichier$ + " a été supprimé. ")
Else
    MsgBox("Le fichier n'existe pas ou il est impossible à supprimer : " +
    NomFichier$)
Endif

#IgnoreErrors=0

```

Read() et Write()

L'instruction **Read** permet de lire le contenu d'un fichier de données de type texte. L'instruction **Write** permet d'écrire des données dans un fichier texte. Ces instructions ne sont pas décrites dans ce manuel car dans l'automatisation de sites Web, les instructions ReadExcel et WriteExcel sont les plus utilisées. Vous pouvez vous reporter au livre WinTask traitant de l'automatisation d'applications Windows ou à l'aide en ligne pour ces deux instructions.

ReadExcel et WriteExcel

Le langage WinTask inclut deux instructions pour lire ou écrire dans des fichiers Excel (fichiers d'extension .XLS ou .XLSX pour Excel 2007/2010). Microsoft Excel doit être installé sur le PC.

L'instruction **ReadExcel** permet de lire une rangée de cellules se trouvant dans une même colonne ou même ligne. Le fichier Excel peut rester ouvert quand cette instruction est utilisée.

L'instruction **WriteExcel** permet d'écrire des données dans une rangée de cellules se trouvant dans une même colonne ou une même ligne. Comme WriteExcel met à jour le fichier Excel directement, il ne faut pas que le fichier Excel soit ouvert quand WriteExcel est utilisé. Le fichier Excel doit avoir été créé auparavant (WinTask inclut une instruction de création de fichier Excel, **CreateExcelFile**).

Les deux instructions ont la syntaxe suivante :

```
ret = ReadExcel(FichierExcel$, Rangee_cellules$, Tableau_donnees$())
```

```
ret = WriteExcel(FichierExcel$, Rangee_cellules$, Tableau_donnees$())
```

Le premier paramètre **FichierExcel\$** contient le nom du fichier Excel à lire ou dans lequel écrire. Si le chemin n'est pas spécifié, le fichier Excel est recherché dans le répertoire par défaut des scripts WinTask.

Le deuxième paramètre **Rangee_cellules\$** spécifie dans quelles cellules, les données sont écrites ou lues. La chaîne de caractères est composée de deux parties : le nom de la feuille et la rangée de cellules séparés par un point d'exclamation (par exemple,

"Feuil1!A9:F9"). Si le nom de la feuille est omis, la rangée de cellules est celle de la première feuille. Par exemple "Dépenses!B3:B5" spécifie les cellules B3, B4 et B5 de la feuille Dépenses, "D8:H8" spécifie les cellules D8, E8, F8, G8, H8 de la première feuille.


Le paramètre **Tableau_donnees\$()** spécifie le tableau contenant les données à écrire, ou le tableau qui va recevoir les données lues. La première donnée lue est stockée dans l'élément 0 de ce tableau. Le tableau doit être défini en début de script en utilisant l'instruction *DIM*.

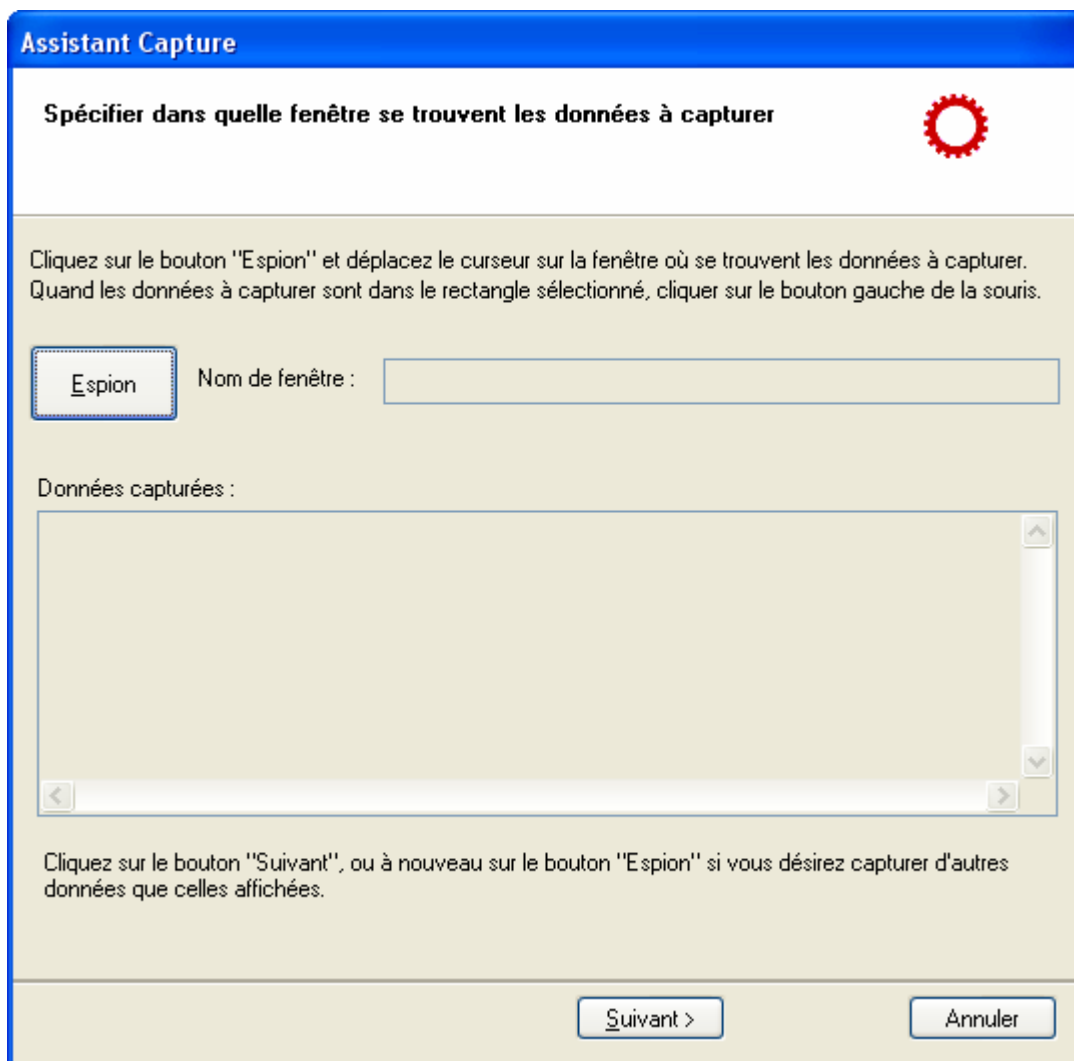
Instructions de capture de données Web

Les instructions **CaptureHTML** et **CaptureTableHTML** permettent de capturer des données sur une page Web. L'Assistant de Capture permet de générer les instructions et l'exercice suivant illustre sa mise en œuvre.

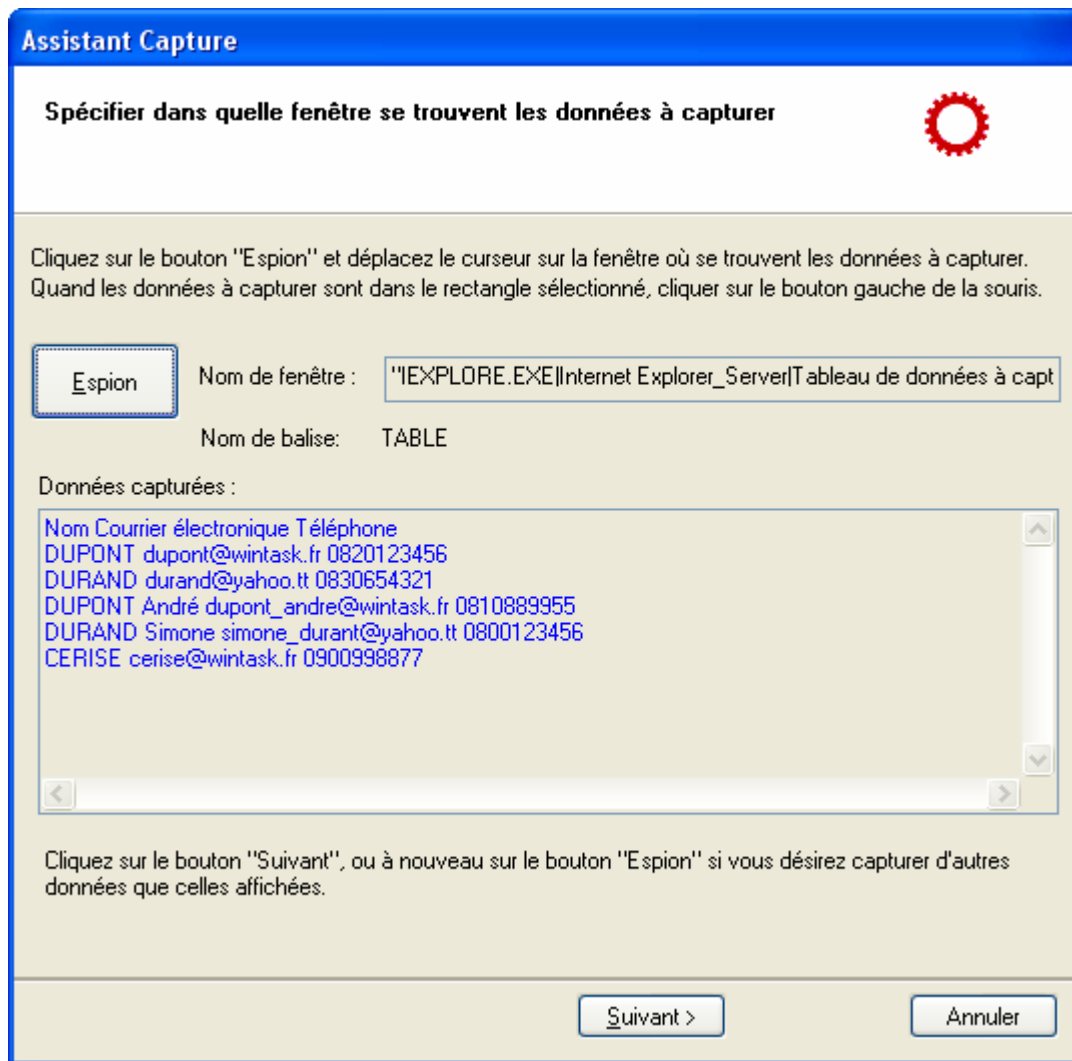
Exercice 6 : Capture de données Web

Cet exercice de programmation montre comment un script peut lire des données affichées dans une page Web et les écrire dans un fichier Excel. La solution de l'exercice 6 est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de création de script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Nouveau** de la barre d'outils pour créer un nouveau script.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** (l'Assistant de Capture ne fonctionne pas encore avec Firefox dans la version 3.8 de WinTask) et cliquez sur le bouton **OK**.
3. La boîte de dialogue **Démarrer Internet Explorer** s'affiche. Dans le champ **Adresse Web**, tapez **www.wintask.fr/demos** et cliquez sur le bouton **OK**.
4. La page titre **Pages Démonstration WinTask** s'affiche, cliquez sur le lien **Tableau de données**.
5. La page de titre **Tableau de données à capturer** s'affiche. Le but du script est de capturer le tableau de trois colonnes Nom, Courrier électronique, Téléphone et d'écrire les données capturées dans un fichier Excel comportant ces trois mêmes colonnes.
6. Dans la barre d'outils WinTask flottante, cliquez sur l'icône de Capture . Le mode Enregistrement est momentanément arrêté.



7. Cliquez sur le bouton **Espion** de l'écran Assistant Capture. Le curseur souris prend la forme d'une loupe.
8. Déplacez le curseur souris sur la table à capturer, vous devez voir un rectangle noir entourer le tableau quand le curseur souris est dessus. Cliquez et les données capturées s'affichent dans l'écran **Assistant Capture**.



9. Cliquez sur le bouton **Suivant**. Sur la boîte de dialogue **Spécifier dans quel objet HTML se trouvent les données à capturer**, cliquez simplement sur le bouton **Suivant**. La boîte de dialogue **Sélectionner les données à capturer** s'affiche.

Assistant Capture

Sélectionner les données à capturer

Dans le tableau ci-dessous, sélectionnez les cellules que vous désirez capturer. Les données extraites seront alors affichées dans le tableau "Données extraites".

	C1	C2	C3				
R1	Nom	Courrier	Téléphone				
R2	DUPONT	dupont@win	0820123456				
R3	DURAND	durand@yah	0830654321				
R4	DUPONT	dupont_andri	0810889955				
R5	DURAND	simone_dura	0800123456				
R6	CERISE	cerise@wint	0900998877				

Données extraites :

	C1	C2	C3
R1	Nom	Courrier	Téléphone
R2	DUPONT	dupont@win	0820123456
R3	DURAND	durand@yah	0830654321
R4	DUPONT	dupont_andri	0810889955
R5	DURAND	simone_dura	0800123456
R6	CERISE	cerise@wint	0900998877

< Précédent Suivant > Annuler

10. Sélectionnez les trois colonnes C1 à C3 à l'aide de la souris.

Assistant Capture

Sélectionner les données à capturer

Dans le tableau ci-dessous, sélectionnez les cellules que vous désirez capturer. Les données extraites seront alors affichées dans le tableau "Données extraites".

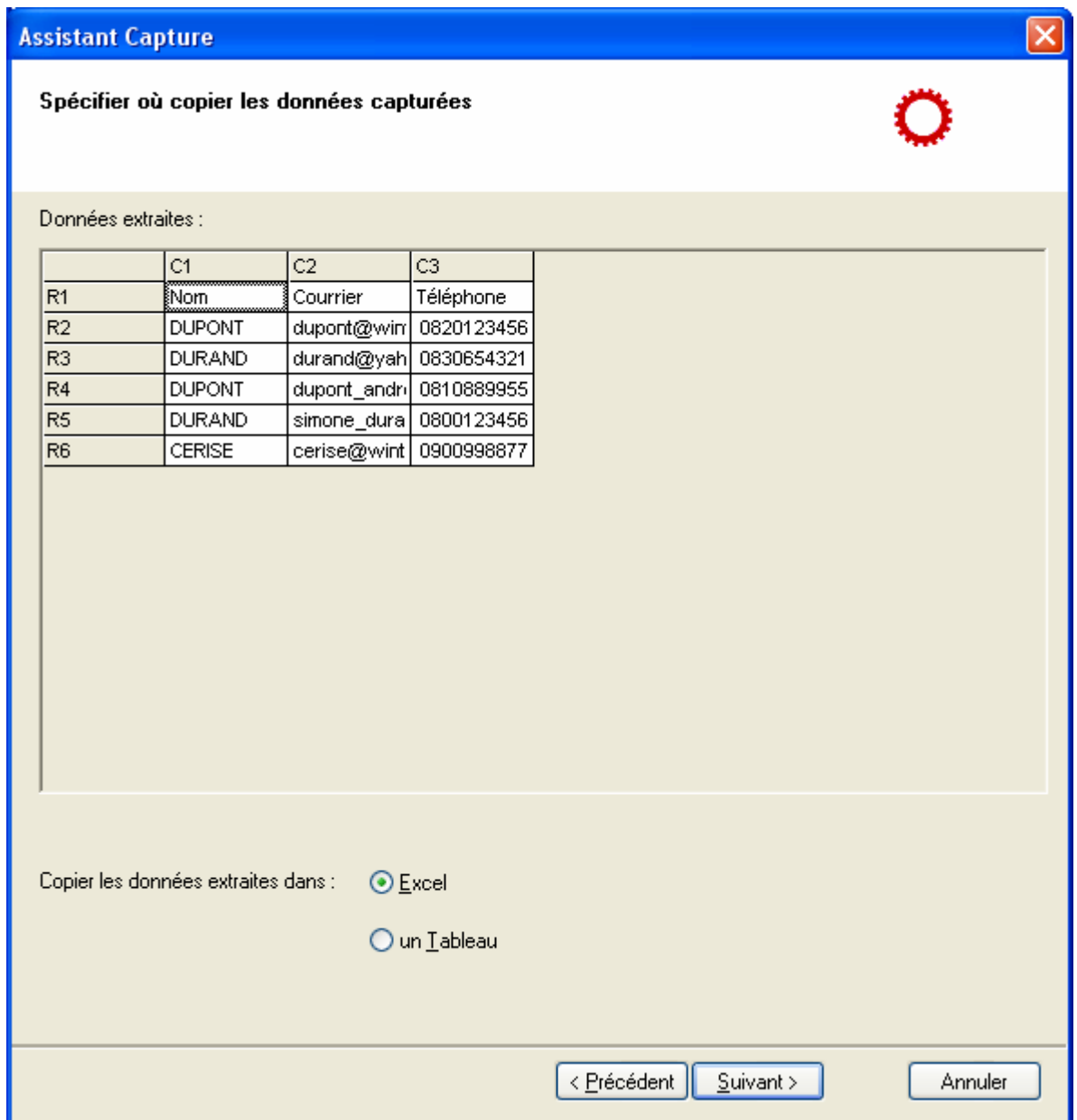
	C1	C2	C3				
R1	Nom	Courrier	Téléphone				
R2	DUPONT	dupont@win	0820123456				
R3	DURAND	durand@yah	0830654321				
R4	DUPONT	dupont_andr	0810889955				
R5	DURAND	simone_dura	0800123456				
R6	CERISE	cerise@wint	0900998877				

Données extraites :

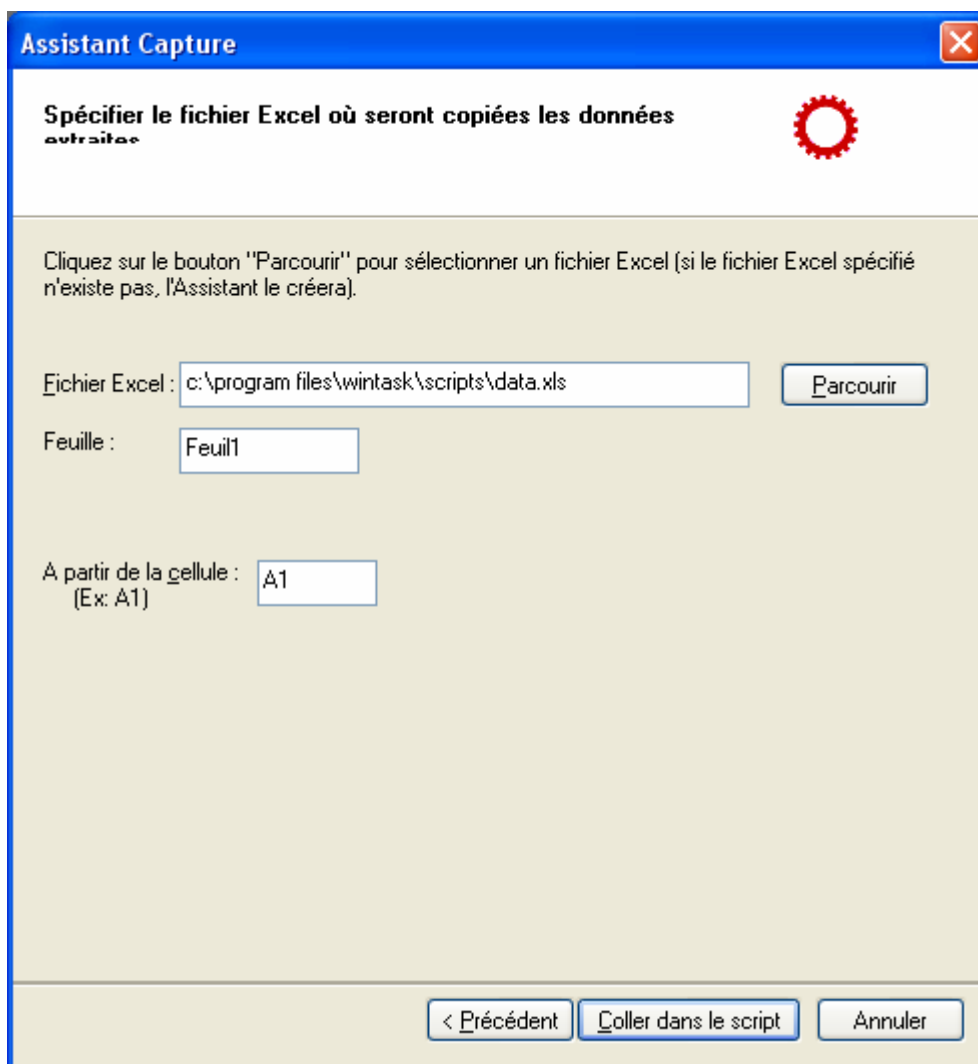
	C1	C2	C3
R1	Nom	Courrier	Téléphone
R2	DUPONT	dupont@win	0820123456
R3	DURAND	durand@yah	0830654321
R4	DUPONT	dupont_andr	0810889955
R5	DURAND	simone_dura	0800123456
R6	CERISE	cerise@wint	0900998877

< Précédent Suivant > Annuler

11. Cliquez sur le bouton **Suivant**. La boîte de dialogue **Spécifier où copier les données capturées** s'affiche. Comme par défaut, les données sont copiées dans Excel, cliquez simplement sur le bouton **Suivant**.



12. La boîte de dialogue **Spécifier le fichier Excel où seront copiées les données extraites** s'affiche. Dans le champ Fichier Excel, saisissez un nom de fichier Excel incluant le chemin complet, par exemple c:\program files\wintask\scripts\data.xls (si le fichier Excel n'existe pas, il sera créé). Avec un Windows 64 bits, le chemin complet est c:\program files (x86)\wintask\scripts\data.xls



13. Cliquez sur le bouton **Coller dans le script**.
14. Le mode Enregistrement redevient actif. Fermez la fenêtre Internet Explorer.
15. Arrêtez le mode Enregistrement en cliquant sur la première icône de la barre d'outils WinTask flottante.

L'Assistant de capture a généré 3 lignes en début de script :

```
Dim tabcell_2$(100)
Dim tabcell_1$(100)
Dim tabcell_0$(100)
```

Ces trois lignes déclarent les tableaux de chaînes de caractères qui vont être utilisés pour stocker les données capturées. WinTask fonctionne avec des tableaux à une dimension, donc comme trois colonnes doivent être capturées, il faut trois tableaux. Les noms de ces tableaux sont des noms génériques, vous pouvez modifier les noms pour refléter par exemple l'intitulé de la colonne (donc par exemple *Dim Nom\$(100)* au lieu de *Dim tabcell_0\$(100)*).

L'Assistant de capture a également généré les instructions de capture proprement dites une fois la page chargée :

```
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C1:R6C1", tabcell_0$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!A1:A6",
tabcell_0$())
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C2:R6C2", tabcell_1$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!B1:B6",
tabcell_1$())
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C3:R6C3", tabcell_2$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!C1:C6",
tabcell_2$())
```

C'est en fait 3 fois la même structure pour capturer 3 colonnes. La ligne CaptureTableHTML capture la table identifiée par le libellé de sa première colonne, soit "TABLE[CONTENT='Nom']", capture les cellules R1C1 à R6C1, soit la colonne C1 (première colonne) pour 6 lignes et stocke ces données dans un tableau de nom tabcell_0\$() qui a été déclaré auparavant par l'instruction Dim. Le début de chaque ligne *ret=* permet de stocker dans une variable résultat de type numérique le nombre de données capturées ou le nombre de données écrites dans le fichier Excel.

16. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Enregistrez le script sous le nom **scriptweb06**. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
17. Ouvrez le fichier **c:\program files\wintask\scripts\data.xls** dans Excel (ou **c:\program files (x86) \wintask\scripts\data.xls** . Vérifiez les données écrites automatiquement dans Excel.

Saisie automatique dans un formulaire Web

La saisie dans un formulaire Web s'effectue comme d'habitude en utilisant le mode Enregistrement. Les objets HTML caractérisant chacun des champs du formulaire sont spécifiés par leur descripteur HTML généré automatiquement par le mode Enregistrement. Si vous désirez savoir comment est construit un descripteur HTML, vous pouvez consulter l'aide de WinTask en recherchant le mot *Descripteur HTML*.

La saisie dans un tel formulaire ne devient intéressante que couplée avec une itération et un fichier externe de données afin de saisir automatiquement n fois le formulaire avec les données issues du fichier externe. Ce fichier est le plus souvent un fichier Excel.

Les deux exercices suivants illustrent la façon de procéder, d'abord en constituant à l'aide du mode Enregistrement la saisie de constantes une fois dans le formulaire, puis en modifiant la trame ainsi obtenue pour ajouter la boucle et la saisie de tableaux de données au lieu de constantes.

Exercice 7 : Saisie de constantes dans un formulaire

Cet exercice de programmation montre comment utiliser le mode Enregistrement pour saisir des constantes dans un formulaire. La solution de l'exercice 7 est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Nouveau** de la barre d'outils pour créer un nouveau script.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** ou **Mozilla Firefox** suivant le navigateur que vous désirez utiliser et cliquez sur le bouton **OK**.
3. La boîte de dialogue **Démarez Internet Explorer** ou **Démarez Mozilla Firefox** s'affiche. Dans le champ **Adresse Web**, tapez **www.wintask.fr/demos** et cliquez sur le bouton **OK**.
4. La page de titre **Pages Démonstration WinTask** s'affiche. Cliquez sur le lien **Formulaire**.
5. La page de titre **Formulaire** s'affiche. Dans le champ Nom, saisissez **DUPONT** ; dans le champ E-mail, saisissez **dupont@wintask.fr** ; dans le champ Téléphone, saisissez **820123456** et enfin cliquez sur le bouton **Effacer**.
6. Fermez la fenêtre du navigateur.
7. Arrêtez le mode Enregistrement en cliquant sur la première icône de la barre d'outils WinTask flottante.

Le script généré même s'il est très simple comporte en fait trois parties qui se retrouvent dans tout script de saisie automatique de formulaire. Le début permet de lancer le site et de naviguer dans le site pour aller sur la page du formulaire à remplir, la deuxième partie remplit le formulaire, la troisième partie sort du site. Dans cet exercice, la deuxième partie utilise l'instruction WriteHTML pour saisir des données constante :

```
WriteHTML("INPUT TEXT[NAME= 'nom']", "DUPONT")
WriteHTML("INPUT TEXT[NAME= 'email']", "dupont@wintask.fr")
WriteHTML("INPUT TEXT[NAME= 'tel']", "820123456")
```

Ces trois lignes ont la même structure, le premier paramètre de l'instruction WriteHTML spécifie le descripteur HTML du champ dans lequel écrire, donc par exemple "INPUT TEXT[NAME= 'nom']" pour le champ dont le nom interne HTML est *nom*. Le deuxième paramètre est la donnée constante à écrire dans le champ, par exemple "DUPONT".

La ligne juste après :

```
ClickHTMLElement("INPUT RESET[VALUE= 'Effacer']")
```

automatise le clic sur le bouton Effacer.

Une nouvelle fois, toutes ces instructions sont générées automatiquement par le mode Enregistrement et il n'est pas besoin d'apprendre la syntaxe précise des descripteurs HTML.

8. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Enregistrez le script sous le nom **scriptweb07**. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
9. Le site est lancé, la page du formulaire s'affiche et la saisie s'effectue très rapidement, le bouton Effacer est cliqué et la fenêtre du navigateur est fermée.

Exercice 8 : Saisie de données issues d'Excel dans un formulaire

Cet exercice de programmation montre comment modifier le script de l'exercice 7 pour saisir dans le formulaire les données lues dans le fichier Excel créé par la capture Web de l'exercice 6. Des tableaux, des variables et une itération sont utilisés, comme d'habitude, la solution de l'exercice 8 est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Si le script **scriptweb07.src** est affiché,

prenez directement à l'étape 3, sinon sélectionnez l'option **Fichier/Fermer tout**.

2. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **scriptweb07.src**.
3. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **scriptweb08a.src**. Cette étape permet de conserver le script **scriptweb07.src** tel qu'il était à l'exercice 7.
4. Avant de lancer le site, il faut lire les données dans le fichier Excel et remplir trois tableaux avec ces données, le premier portera le nom Nom\$, le deuxième portera le nom Email\$ et le dernier Telephone\$.
5. En tout début de script, ajoutez les instructions *DIM* pour déclarer ces trois tableaux.
6. Juste après ces trois lignes DIM, ajoutez la déclaration d'une variable nomfichier\$ qui contient le nom du fichier Excel (si vous utilisez Excel 2007/2010, l'extension est xlsx au lieu de xls):

```
nomfichier$= "c:\program files\wintask\scripts\data.xls"
```

Pour un Windows 64 bits :

```
nomfichier$= "c:\program files (x86)\wintask\scripts\data.xls"
```

7. Ajoutez ensuite les trois lignes ReadExcel lisant chaque colonne du fichier Excel et remplissant le tableau correspondant. Attention : la première ligne du fichier Excel contient les intitulés de colonnes qui ne doivent pas être saisis dans le formulaire, donc ne prenez les données qu'à partir de la cellule A2. La première ligne ReadExcel s'écrit donc ainsi :

```
ReadExcel(nomfichier$, "A2:A6", nom$())
```

8. Une fois les trois lignes ReadExcel ajoutées, il faut insérer la boucle et remplacer dans les lignes WriteHTML le deuxième paramètre, constante, par l'élément du tableau à écrire.
9. Le premier élément d'un tableau commence à l'index 0, donc la boucle commence à l'index $i=0$.
10. Commencez la boucle par la ligne *while* $i < 5$ puisqu'il y a 5 éléments de 0 à 4 dans les tableaux. Une condition pour terminer la boucle indépendante du nombre d'éléments dans les tableaux est étudiée à la fin de cet exercice.
11. Puis remplacez dans les lignes WriteHTML la constante par l'élément i du tableau. Par exemple,

```
WriteHTML("INPUT TEXT[NAME= 'nom']", "DUPONT")
```

devient

```
WriteHTML("INPUT TEXT[NAME= 'nom']", nom$(i))
```

12. Une fois les trois lignes WriteHTML modifiées, ajoutez après la ligne ClickHTMLItem l'incrémentation de l'index i , soit $i = i + 1$
13. Terminez la boucle par *Wend*. Cette instruction doit se trouver au-dessus de la ligne CloseWindow qui ferme la fenêtre Internet Explorer.
14. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
15. Le site est lancé, la page du formulaire s'affiche et la saisie s'effectue très rapidement, le bouton Effacer est cliqué et la fenêtre du navigateur est fermée.
16. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **scriptweb08b.src**
17. Remplacez la condition while $i < 5$ par une condition terminant la boucle si nom\$(i) est vide (une chaîne vide est "").
18. Lancez l'exécution pour tester cette modification.

Autres instructions pour les formulaires

Les formulaires incluent souvent des listes et le script d'automatisation doit sélectionner un élément dans cette liste. WinTask inclut des instructions de gestion de telles listes.

L'instruction **SelectHTMLItem** sélectionne un élément d'une liste affichée dans une page Web. Voici un exemple sur la page Formulaire, de la sélection d'un élément dans la liste :

```
SelectHTMLItem("SELECT[NAME= 'sujet']", "Améliorations souhaitées")
```

Cette ligne sélectionne l'élément "Améliorations souhaitées" dans la liste de nom HTML *sujet*.

L'instruction **SelectedHTMLItem\$** renvoie l'élément de la liste qui est sélectionné dans la liste spécifiée. Voici un exemple sur la page Formulaire, pour retourner quel élément est sélectionné :

```
valeur_selectionnee$=SelectedHTMLItem$("SELECT[NAME= 'sujet']")
```

La variable *valeur_selectionnee\$* renvoie l'élément qui a été sélectionné dans la liste de nom HTML *sujet*.

L'instruction **ListHTMLItem\$** renvoie le n ième élément de la liste dans la liste spécifiée. La numérotation des éléments commencent en 0. Voici un exemple sur la page Formulaire, pour retourner l'élément de la liste de rang 3 (donc le quatrième) :

```
valeur_liste$=ListHTMLItem$("SELECT[NAME= 'sujet']",3)
```

La variable *valeur_liste\$* renvoie le contenu du quatrième élément de la liste de nom HTML *sujet*.

L'instruction **CheckedHTML** donne le statut du bouton radio ou de la case à cocher spécifié dans un formulaire Web. Voici un exemple sur la page Formulaire, pour savoir si la case à cocher Requête urgente est cochée ou pas.

```
statut=CheckedHTML("INPUT CHECKBOX[NAME= 'contactsoon']")
```

La variable numérique *statut* vaut 1 si la case à cocher de nom HTML *contactsoon* (nom HTML de la case à cocher Requête urgente) est cochée, elle vaut 0 si la case n'est pas cochée.

L'instruction **GetHTMLEditText** capture le contenu du champ de formulaire Web spécifié. Voici un exemple sur la page Formulaire, pour capturer le contenu du champ Nom.

```
GetHTMLEditText("INPUT TEXT[NAME= 'nom']", a$)
```

La variable chaîne de caractères *a\$* renvoie le contenu du champ *nom*.

L'instruction **WriteHTMLPaste** écrit du texte dans un champ de formulaire en simulant un Copier/Coller (alors que **WriteHTML** écrit caractère par caractère). Cette instruction est à utiliser pour écrire du texte formaté, par exemple un champ de formulaire style applet Word. Voici un exemple sur la page Formulaire, pour écrire d'un seul coup dans le champ Message.

```
WriteHTMLPaste("TEXTAREA[NAME= 'message']", "Bla bla")
```


Fonctions et Sous-Programmes (Sub)

Ce chapitre nécessite plus de connaissances en développement. Il indique comment créer des scripts d'automatisation structurés.

Le langage WinTask inclut de nombreuses instructions qui rendent l'automatisation de tâches aisée. Un même traitement doit parfois s'effectuer à différents endroits du script, ou un même processus d'automatisation est à mettre en œuvre dans différents scripts. Avec la notion de Sous-Programme incluse dans WinTask, il est possible d'écrire des Fonctions ou des Sub qui peuvent être appelées n'importe où dans un script, permettant ainsi d'écrire des scripts structurés et facilement maintenables.

Sub...EndSub

La syntaxe pour écrire un sous-programme est **Sub... EndSub** :

```
Sub <nom de la sub>([<paramètre1>[, <paramètre2>] ...])  
    <instructions>  
EndSub
```

Le nom de la Sub est spécifié dans <nom de la sub> et doit être unique dans l'ensemble du script. Une Sub peut avoir des paramètres, la liste est mise entre parenthèses après le nom de la Sub et les paramètres sont séparés par des virgules. Si un paramètre est de type chaîne de caractères, son nom doit se terminer par le caractère \$. Pour sortir abruptement d'une Sub par exemple dans une condition, l'instruction à utiliser est **ExitSub**.

ASTUCE : Comme les variables sont valides pour l'ensemble du script (sauf à utiliser le mot-clé **LOCAL** non traité dans ce manuel), utilisez des noms uniques pour l'ensemble du script.

ASTUCE : Veillez à ce que la Sub ait été déclarée avant de l'utiliser (le bloc Sub...EndSub doit être écrit en début de script, juste après les lignes DIM, et l'appel à la Sub ne peut avoir lieu qu'après cette déclaration).

L'autre type de sous-programme disponible dans WinTask est Fonction.

Function...EndFunction

La syntaxe pour écrire une fonction est **Function...EndFunction** :

Function <nom de fonction>([<paramètre1>[, <paramètre2>] ...])

<instructions>

<nom de fonction> = <valeur>

EndFunction

Une fonction est identique à une Sub, sauf que la fonction retourne un résultat. Le nom de la fonction est spécifié dans <nom de fonction> et doit être unique dans l'ensemble du script. Une fonction peut avoir des paramètres, la liste est mise entre parenthèses après le nom de la fonction et les paramètres sont séparés par des virgules. Si un paramètre est de type chaîne de caractères, son nom doit se terminer par le caractère \$. Pour sortir abruptement d'une fonction par exemple dans une condition, l'instruction à utiliser est **ExitFunction**. L'assignation du résultat à la fonction doit se faire avant de sortir de la fonction. Si la valeur retournée par la fonction est une chaîne de caractères, le caractère \$ doit être le dernier caractère du nom de la fonction.

ASTUCE : Comme les variables sont valides pour l'ensemble du script (sauf à utiliser le mot-clé **LOCAL** non traité dans ce manuel), utilisez des noms uniques pour l'ensemble du script.

ASTUCE : Veillez à ce que la fonction ait été déclarée avant de l'utiliser (le bloc Function...EndFunction doit être écrit en début de script, juste après les lignes DIM, et l'appel à la fonction ne peut avoir lieu qu'après cette déclaration).

ASTUCE : Si la fonction n'a pas assigné une valeur à la variable résultat qu'elle doit retourner, le résultat est 0 ou une chaîne vide si le nom de la fonction se termine par le caractère \$.

Voici un exemple de Function et de son appel :

```
Function Absolute_diff(entier1, entier2)
```

```
  If (entier1 = entier2) Then
```

```
    Absolute_diff = 0
```

```
    ExitFunction
```

```
  Endif
```

```
  If (entier1 > entier2) Then
```

```
    Absolute_diff = entier1 - entier2
```

```
  Else
```

```
    Absolute_diff = entier2 - entier1
```

```
  Endif
```

```
EndFunction
```

```
' Programme principal
```

```
Nombre = 89
```

```
Resultat = Absolute_diff(Nombre, 123)
```

```
Msgbox(Resultat)
```

Script complet de capture sur un site "réel"

L'objet de ce chapitre est "d'assembler les morceaux" vus tout au long de ce manuel afin d'écrire un script complexe de capture sur un site de production. La démarche décrite s'applique à n'importe quel site sur lequel vous avez des données à capturer, données se trouvant sur plusieurs pages. Le site Web utilisé est un site en anglais mais il comporte peu de mots en anglais et cela ne gêne pas le propos.

Avant de commencer à écrire quelque code que ce soit, vous devez comprendre la navigation sur le site et visualiser les données à capturer. Donc manuellement, chargez la page Web d'adresse :

<http://www.donaldson.com/en/index.html>

Cliquez sur le lien **Part Search/Cross Reference**. La page Cross Reference Search se charge et dans le premier champ **Item #**, saisissez **145*** puis cliquez sur le bouton **Search**. La page suivante affiche un premier tableau de résultats, les données des cinq colonnes sont à capturer. Le script écrit les données à capturer dans un fichier Excel, qui comportera donc 5 colonnes.

Il faut maintenant réfléchir à la structure du script : cinq tableaux sont nécessaires, donc cinq lignes Dim en début de script pour les déclarer. Le programme principal commence par le lancement de l'adresse Web, puis la saisie de la requête (saisie de 145* et clic sur le bouton Search). Bien sûr ce 145* devrait être une variable afin de pouvoir exécuter le script avec n'importe quel nombre à rechercher. Deux étapes sont ensuite répétitives : la capture des données d'une page et l'écriture de ces données dans Excel.

Comme l'écriture de ce script est assez longue, le code généré au fur et à mesure de cet exercice sera inclus au niveau de chacune des étapes (texte en italiques) et vous pouvez consulter le script (Exercice 9) dans sa version définitive à l'appendice C.

Exercice 9 : Script complet de capture

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Nouveau** de la barre d'outils WinTask.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** et cliquez sur le bouton **OK**.
3. La boîte de dialogue **Démarez Internet Explorer** s'affiche. Dans le champ **Adresse Web**, tapez **<http://www.donaldson.com/en/index.html>** et cliquez sur le bouton **OK**.

4. La page de titre **Donaldson Company** s'affiche. Cliquez sur le lien **Part Search/Cross Reference**. La page de titre Cross Reference Search s'affiche. Saisissez **145*** dans le premier champ **Item #** et cliquez sur le bouton **Search**.


5. Arrêtez le mode Enregistrement, le script généré doit ressembler à

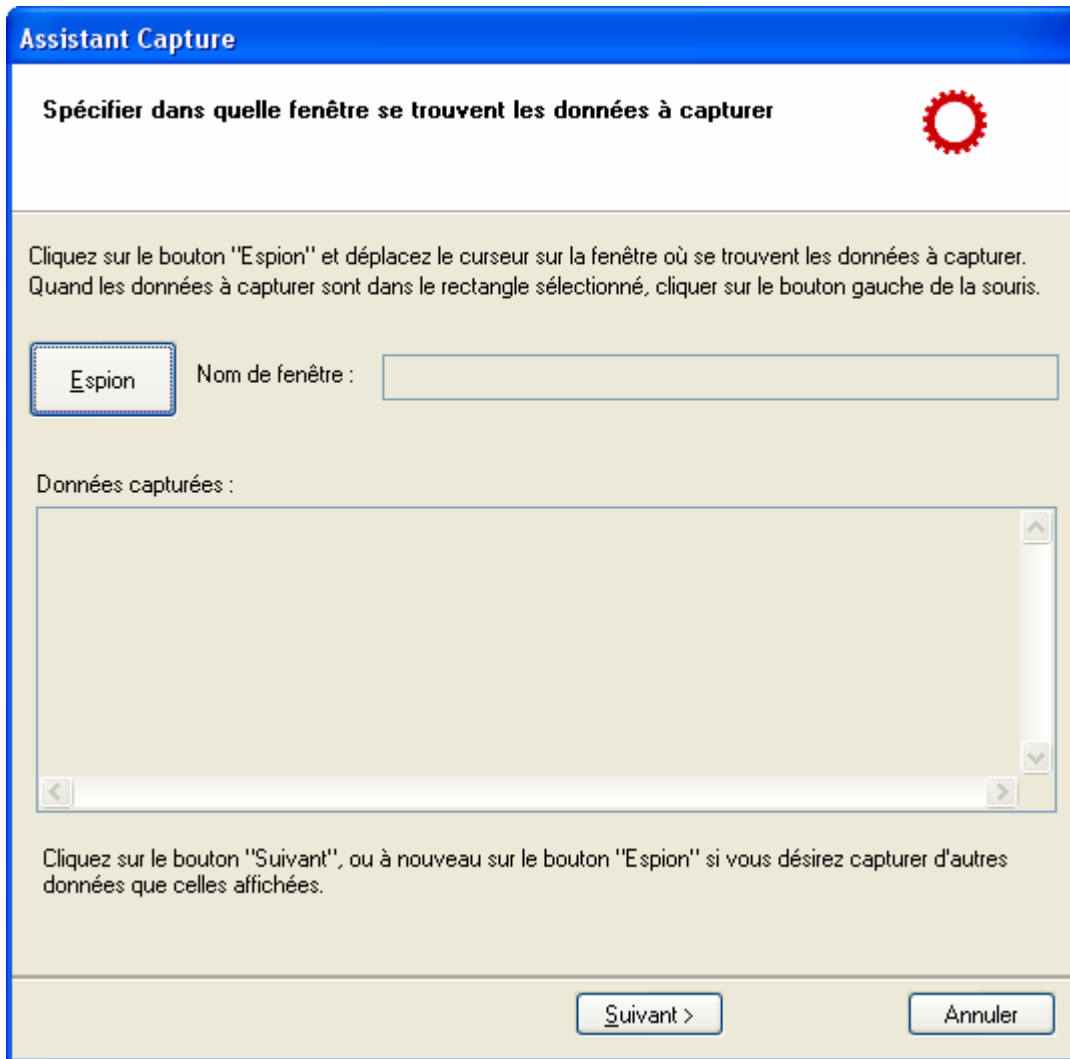
```
StartBrowser("IE", "http://www.donaldson.com/en/index.html", 3 )
```

```
UsePage("Donaldson Company, Inc. - Americas Region - English")  
ClickHTMLElement("A[INNERTEXT= 'Part Search/Cross Re']")
```

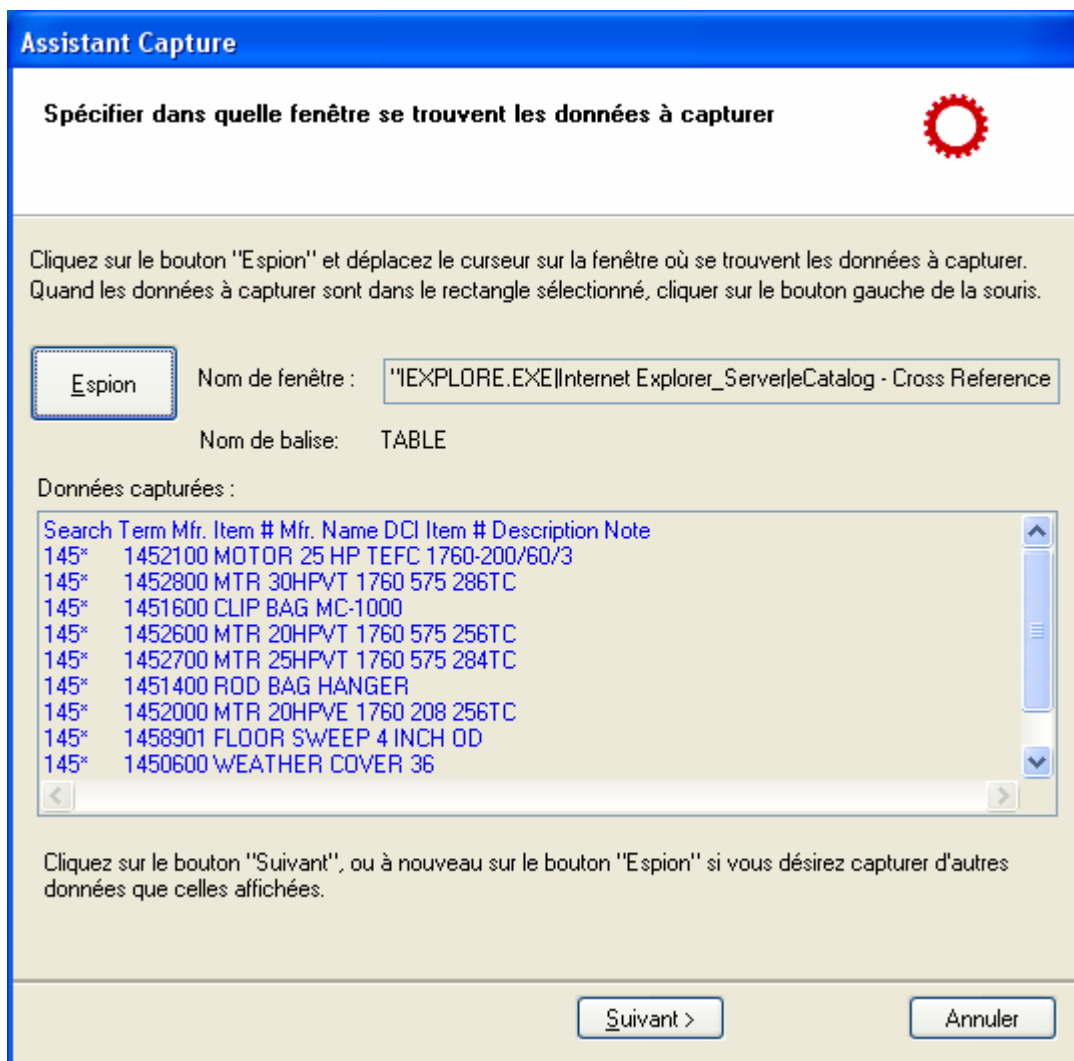
```
UsePage("eCatalog - Cross Reference Search")
```

```
WriteHTML("INPUT TEXT[NAME= 'part_0']", "145*")  
ClickHTMLElement("INPUT SUBMIT[VALUE= 'Search']")
```

6. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **scriptweb09.src**
7. Vérifiez que le curseur texte dans la fenêtre de l'Editeur est après la dernière ligne de code générée et démarrez à nouveau le mode Enregistrement en sélectionnant le menu **Démarrer/Enregistrement**. Dans la barre d'outils WinTask flottante, cliquez sur l'icône de Capture . Le mode Enregistrement est momentanément arrêté.



8. Cliquez sur le bouton **Espion** de l'écran Assistant Capture. Le curseur souris prend la forme d'une loupe.
9. Déplacez le curseur souris sur la table à capturer : placez la loupe sur l'intitulé de la première colonne du tableau (**Search Term**), vous devez voir un rectangle noir entourer tout le tableau. Cliquez une fois sur le bouton gauche de la souris, les données capturées s'affichent dans l'écran **Assistant Capture**.



10. Cliquez sur le bouton **Suivant**. Sur la boîte de dialogue **Spécifier dans quel objet HTML se trouvent les données à capturer**, cliquez simplement sur le bouton **Suivant**. La boîte de dialogue **Sélectionner les données à capturer** s'affiche.
11. Comme cette capture devra être répétée pour chaque page et qu'il ne faut pas écrire dans le fichier Excel à chaque fois l'intitulé des colonnes, sélectionnez les cinq colonnes C1 à C5 à l'aide de la souris, mais sans prendre la ligne R1.

Assistant Capture

Sélectionner les données à capturer

Dans le tableau ci-dessous, sélectionnez les cellules que vous désirez capturer. Les données extraites seront alors affichées dans le tableau "Données extraites".

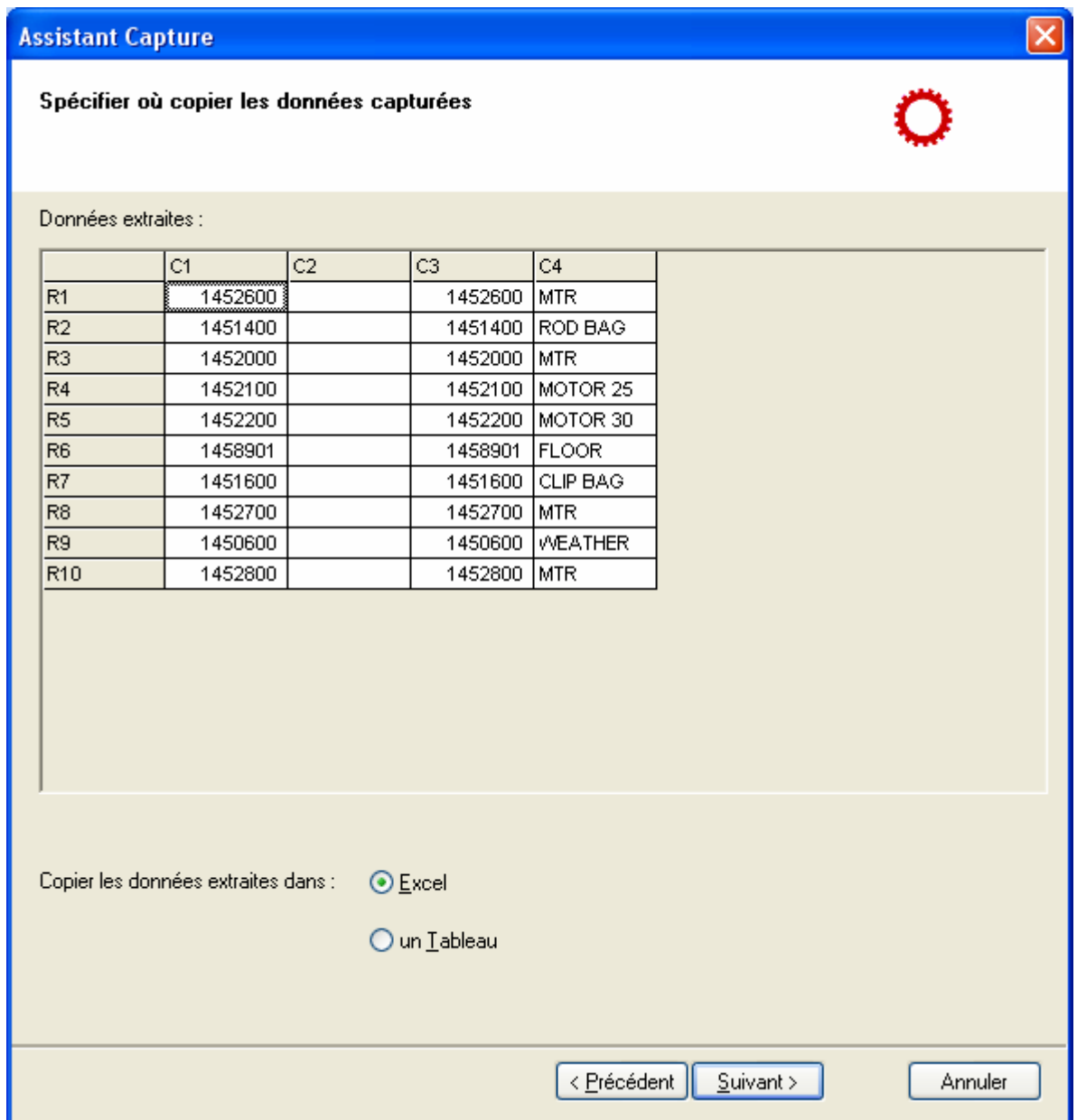
	C1	C2	C3	C4	C5	C6
R1	Search Term	Mfr. Item #	Mfr. Name	DCI Item #	Description	Note
R2	145*			1452100	MOTOR 25	
R3	145*			1452800	MTR	
R4	145*			1451600	CLIP BAG	
R5	145*			1452600	MTR	
R6	145*			1452700	MTR	
R7	145*			1451400	ROD BAG	
R8	145*			1452000	MTR	
R9	145*			1458901	FLOOR	
R10	145*			1450600	WEATHER	
R11	145*			1452200	MOTOR 30	

Données extraites :

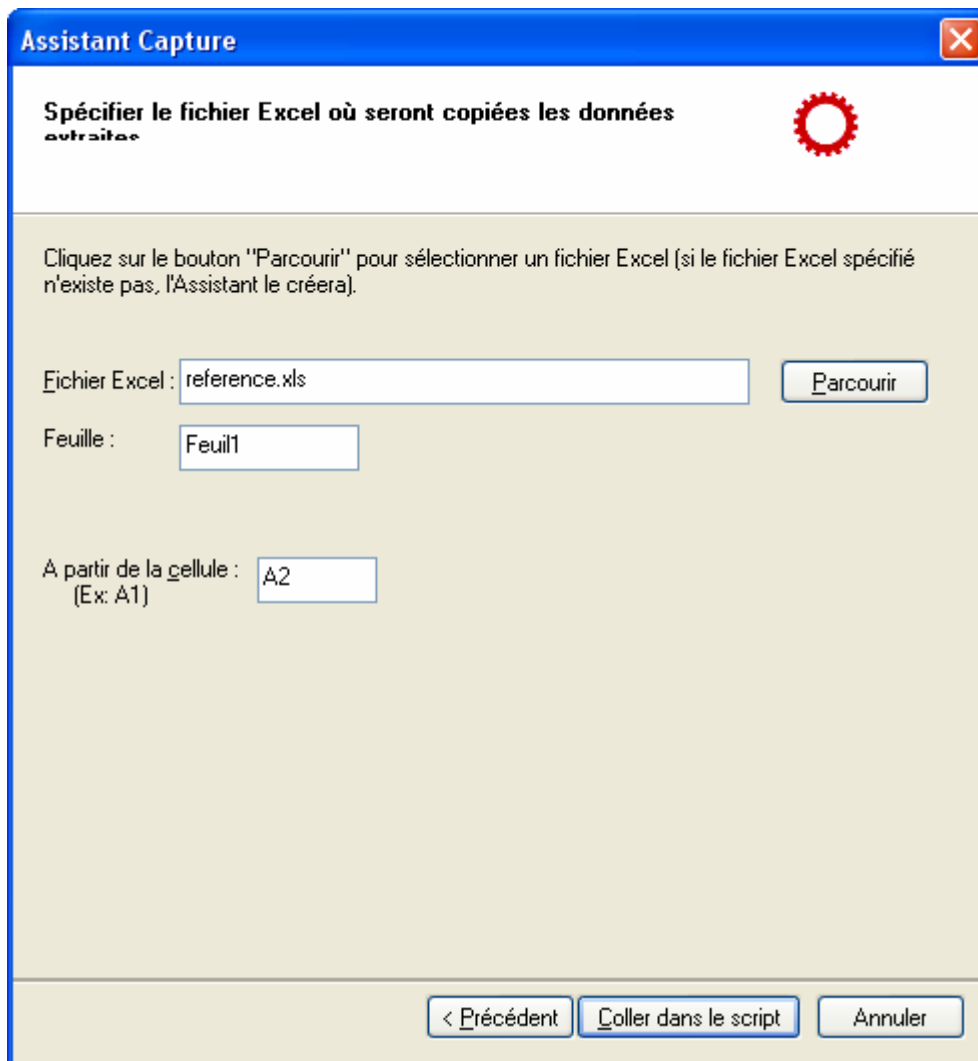
	C1	C2	C3	C4	C5
R1	145*			1452100	MOTOR 25
R2	145*			1452800	MTR
R3	145*			1451600	CLIP BAG
R4	145*			1452600	MTR
R5	145*			1452700	MTR
R6	145*			1451400	ROD BAG
R7	145*			1452000	MTR
R8	145*			1458901	FLOOR
R9	145*			1450600	WEATHER
R10	145*			1452200	MOTOR 30

< Précédent Suivant > Annuler

12. Cliquez sur le bouton **Suivant**. La boîte de dialogue **Spécifier où copier les données capturées** s'affiche. Comme par défaut, les données sont copiées dans Excel, cliquez simplement sur le bouton **Suivant**.



13. La boîte de dialogue **Spécifier le fichier Excel où seront copiées les données extraites** s'affiche. Dans le champ Fichier Excel, saisissez un nom de fichier Excel, par exemple **reference.xls** (ou **reference.xlsx** si vous utilisez Excel 2007/2010). Ce fichier sera créé dans le répertoire où est enregistré le script. Ou sinon incluez le chemin complet, par exemple **c:\program files\wintask\scripts\reference.xls** (si le fichier Excel n'existe pas, il sera créé). Comme la première ligne du fichier Excel contiendra les intitulés de colonnes, dans le champ **A partir de la cellule**, tapez **A2** pour écrire les données à capturer à partir de la colonne A et ligne 2.



14. Cliquez sur le bouton **Coller dans le script**.
15. Le mode Enregistrement redevient actif. Arrêtez le mode Enregistrement en cliquant sur la première icône de la barre d'outils WinTask flottante.
16. Le mode Enregistrement a généré toutes ces lignes :

```

Dim tabcell_4$(100)
Dim tabcell_3$(100)
Dim tabcell_2$(100)
Dim tabcell_1$(100)
Dim tabcell_0$(100)
StartBrowser("IE", "http://www.donaldson.com/en/index.html")
UsePage("Donaldson Company, Inc. - Americas Region - English")
    ClickHTMLElement("A[INNERTEXT= 'Part Search/Cross Re']")
UsePage("eCatalog - Cross Reference Search")
WriteHTML("INPUT TEXT[NAME= 'part_0']", "145*")
ClickHTMLElement("INPUT SUBMIT[VALUE= 'Search']")

UsePage("eCatalog - Cross Reference Search")
    ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C1:R11C1",
tabcell_0$())
    ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!A2:A11", tabcell_0$())
    ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C2:R11C2",
tabcell_1$())
    ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!B2:B11", tabcell_1$())
    ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C3:R11C3",
tabcell_2$())
    ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!C2:C11", tabcell_2$())
    ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C4:R11C4",
tabcell_3$())
    ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!D2:D11", tabcell_3$())
    ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C5:R11C5",
tabcell_4$())
    ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!E2:E11", tabcell_4$())

```

Cinq tableaux ont été déclarés en début de script pour recevoir chaque colonne des données capturées.

Puis chaque colonne est capturée (ligne *CaptureTableHTML*) et écrite dans la colonne correspondante du fichier Excel (ligne *WriteExcel*).

17. Fermez manuellement le site Web puis lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau. Ouvrez le fichier *reference.xls* pour vérifier que la capture s'est bien faite.
18. Simplifiez le script en supprimant les lignes *UsePage* inutiles et rendez-le plus lisible en changeant les variables *tabcell_x\$* pour des noms significatifs.

```
Dim searchterm$(100)
Dim description$(100)
Dim mfrname$(100)
Dim mfritem$(100)
Dim dciitem$(100)
```

```
StartBrowser("IE", "http://www.donaldson.com/en/index.html", 3)
```

```
UsePage("Donaldson Company, Inc. - Americas Region - English")
```

```
ClickHTMLElement("A[INNERTEXT= 'Part Search/Cross Re']")
```

```
UsePage("eCatalog - Cross Reference Search")
```

```
WriteHTML("INPUT TEXT[NAME= 'part_0']", "145*")
```

```
ClickHTMLElement("INPUT SUBMIT[VALUE= 'Search']")
```

```
UsePage("eCatalog - Cross Reference Search")
```

```
ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C1:R21C1",
searchterm$())
```

```
ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls", "Feuil1!A2:A21",
searchterm$())
```

```
ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C2:R21C2",
mfritem$())
```

```
ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!B2:B21", mfritem$())
```

```
ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C3:R21C3",
mfrname$())
```

```
ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!C2:C21", mfrname$())
```

```
ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C4:R21C4",
dciitem$())
```

```
ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!D2:D21", dciitem$())
```

```
ret = CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C5:R21C5",
description$())
```

```
ret = WriteExcel("C:\Program Files\WinTask\Scripts\reference.xls",
"Feuil1!E2:E21", description$())
```

19. Vérifiez que vos modifications fonctionnent : lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau. Ouvrez le fichier reference.xls pour vérifier la capture.
20. Si les résultats sont sur plusieurs pages, le script doit répéter cette capture et cette écriture dans Excel un certain nombre de fois. Transformez la partie Capture en une Sub, de nom capture_une_page, et la partie écriture en une

autre Sub, de nom escrit_resultat. Utilisez un nom de variable pour le fichier Excel, par exemple fichierexcel\$.

```
Dim searchterm$(100)
```

```
Dim description$(100)
```

```
Dim mfrname$(100)
```

```
Dim mfritem$(100)
```

```
Dim dciitem$(100)
```

```
Function capture_une_page()
```

```
UsePage("eCatalog - Cross Reference Search")
```

```
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C1:R21C1",  
searchterm$())
```

```
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C2:R21C2",  
mfritem$())
```

```
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C3:R21C3",  
mfrname$())
```

```
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C4:R21C4", dciitem$())
```

```
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C5:R21C5",  
description$())
```

```
EndFunction
```

```
Function escrit_resultat()
```

```
WriteExcel(fichierexcel$, "A2:A21", searchterm$())
```

```
WriteExcel(fichierexcel$, "B2:B21", mfritem$())
```

```
WriteExcel(fichierexcel$, "C2:C21", mfrname$())
```

```
WriteExcel(fichierexcel$, "D2:D21", dciitem$())
```

```
WriteExcel(fichierexcel$, "E2:E21", description$())
```

```
EndFunction
```

```
fichierexcel$="c:\program files\wintask\scripts\reference.xls"
```

```
StartBrowser("IE", "http://www.donaldson.com/en/index.html", 3)
```

```
UsePage("Donaldson Company, Inc. - Americas Region - English")
```

```
ClickHTMLElement("A[INNERTEXT= 'Part Search/Cross Re']")
```

```
UsePage("eCatalog - Cross Reference Search")
```

```
WriteHTML("INPUT TEXT[NAME= 'part_0']", "145*")
```

```
ClickHTMLElement("INPUT SUBMIT[VALUE= 'Search']")
```

```
capture_une_page()
```

```
escrit_resultat()
```

'Si il y a un bouton Next page sur la page de resultants, ajoutez cette ligne
'ClickHTMLElement("INPUT SUBMIT[VALUE= 'Next page']")

21. Comme d'habitude, vérifiez que vos modifications fonctionnent : lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la

fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau. Ouvrez le fichier reference.xls pour vérifier la capture.

22. Si les résultats sont affichés sur plusieurs pages, vous devez ajouter une boucle pour capturer les données sur les n pages de résultats. La condition de sortie de la boucle est par exemple que le bouton **Next page** n'est plus affiché sur la page en cours. Le test de l'existence de ce bouton s'effectue par l'instruction `ExistHTML`Element. Si `ExistHTML`Element("INPUT SUBMIT[VALUE='Next page']") vaut 1, le bouton est là. Incluez donc la boucle `repeat...until` sur cette condition. Comme il faut cliquer sur le bouton Next page si le bouton est là, une bonne technique de programmation consiste à utiliser une variable de nom par exemple `exit` qui prend la valeur 0 au début et est mise à 1 si la condition de sortie est vue. Cela permet d'éviter l'instruction `Goto`. Voyez ci-dessous le code.

Dim searchterm\$(100)

Dim description\$(100)

Dim mfrname\$(100)

Dim mfritem\$(100)

Dim dciitem\$(100)

Function capture_une_page()

UsePage("eCatalog - Cross Reference Search")

CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C1:R21C1", searchterm\$())

CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C2:R21C2", mfritem\$())

CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C3:R21C3", mfrname\$())

CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C4:R21C4", dciitem\$())

CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C5:R21C5", description\$())

EndFunction

Function escrit_resultat()

WriteExcel(fichierexcel\$, "A2:A21", searchterm\$())

WriteExcel(fichierexcel\$, "B2:B21", mfritem\$())

WriteExcel(fichierexcel\$, "C2:C21", mfrname\$())

WriteExcel(fichierexcel\$, "D2:D21", dciitem\$())

WriteExcel(fichierexcel\$, "E2:E21", description\$())

EndFunction

fichierexcel\$="c:\program files\wintask\scripts\reference.xls"

StartBrowser("IE", "http://www.donaldson.com/en/index.html", 3)

UsePage("Donaldson Company, Inc. - Americas Region - English")

ClickHTMLElement("A[INNERTEXT= 'Part Search/Cross Re']")

UsePage("eCatalog - Cross Reference Search")

WriteHTML("INPUT TEXT[NAME= 'part_0']", "145")*

ClickHTMLElement("INPUT SUBMIT[VALUE= 'Search']")

exit=0

repeat

capture_une_page()

ecrit_resultat()

If existhtmllement("INPUT SUBMIT[VALUE= 'Next page']") = 1 then

ClickHTMLElement("INPUT SUBMIT[VALUE= 'Next page']")

Else

Exit=1

EndIf

until exit = 1

23. Comme d'habitude, vérifiez que vos modifications fonctionnent : lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau. Ouvrez le fichier reference.xls pour vérifier la capture. Vous constatez qu'il y a une erreur.
24. En effet, à chaque itération, dans la fonction `ecrit_resultat`, les lignes `WriteExcel` écrivent les données toujours de la ligne 2 à la ligne 21 et donc écrasent les données capturées de la page précédente. Il faut donc garder en mémoire le numéro de ligne à partir duquel il faut écrire les nouvelles données capturées. Modifiez la fonction en utilisant la variable de nom `numero_ligne`. Consultez la solution à l'appendice C.

Cet exercice a illustré comment un tableau peut être utilisé dans un script structuré. Vous savez maintenant écrire des scripts complexes, partant du mode Enregistrement pour générer un canevas et utilisant le langage pour transformer les actions à répéter en Sub.

Débogage

Erreurs de compilation

Quand vous lancez l'exécution d'un script (fichier SRC), ce dernier est d'abord compilé. Les erreurs de compilation éventuelles sont affichées dans la fenêtre Résultats de l'Editeur (si la fenêtre Résultats n'est pas affichée, sélectionnez **Affichage/Fenêtre Résultats** pour la faire afficher).

En double-cliquant sur une ligne rapportant une erreur, le curseur texte se met dans la fenêtre principale de l'Editeur, sur la ligne où l'erreur se produit.

Souvenez-vous de la structure d'un script :

Les instructions Dim déclarant des tableaux en premier, les déclarations des Sub et Fonctions en deuxième, et l'appel à une fonction à l'intérieur d'une autre nécessite que la fonction appelée ait été déclarée avant, le programme principal en dernier.

Un script comportant des scripts inclus doit respecter cet ordre une fois toutes les inclusions effectuées à la compilation

Un exemple d'instruction Include : Include "ma_boite_outils.src"

Erreurs d'exécution

Trace en utilisant MsgBox ou MsgFrame

Exercice 10 : Tracer l'exécution via MsgBox ou MsgFrame

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Sélectionnez l'option **Fichier/Fermer tout**.
2. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **scriptweb05a.src**.
3. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **scriptweb10.src**. Cette étape permet de conserver le script **scriptweb05a.src** tel qu'il était à l'exercice 5.

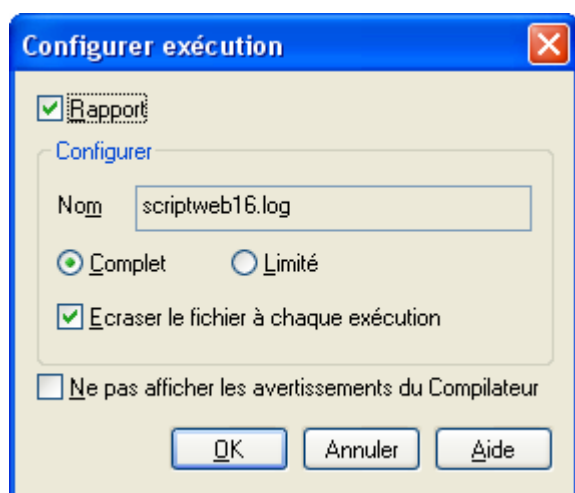
En imaginant que l'index *i* est à surveiller car une erreur se produit, insérez dans le script une instruction *MsgBox* pour faire afficher la valeur de l'index à chaque itération. Lancez l'exécution. Remplacez la ligne *MsgBox* par *MsgFrame* et rejouez.

ASTUCE : Notez la grande différence entre *MsgBox* et *MsgFrame*. *MsgBox* prend le focus et l'utilisateur doit cliquer sur OK après l'affichage pour que le script continue son exécution. *MsgFrame* par contre ne modifie aucunement le comportement applicatif.

Fichier Rapport

Il est possible de paramétrer l'exécution pour que chaque ligne soit écrite dans un fichier rapport (fichier d'extension .LOG).

Sélectionnez le menu **Paramétrer/Exécution** :



Si la case **Complet** est cochée, toutes les lignes exécutées sont écrites dans le fichier rapport. Si **Limité** est coché, seules les instructions *Comment* sont écrites dans le rapport.

Exercice 11 : Tracer l'exécution via un fichier Rapport (fichier log)

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Si le script **scriptweb10.src** n'est pas le script chargé dans la fenêtre de l'Editeur, cliquez sur l'icône **Ouvrir** de la barre d'outils de l'Editeur et ouvrez le script **scriptweb10.src**.
2. Paramétrez l'exécution pour générer un rapport complet (sélectionnez le menu **Paramétrer/Exécution**) et lancez l'exécution (icône **Exéc** de la barre d'outils).
3. Ouvrez le fichier rapport généré (sélectionnez le menu **Fichier/Ouvrir Rapport**) – étudiez les lignes générées. Fermez le fichier log.

4. Remplacez la ligne `msgbox` ou `msgframe` par la ligne `Comment` ("l'index i vaut :"+str\$(i)).
5. Ajoutez une autre ligne `Comment` pour écrire dans le fichier rapport la valeur de l'index i.
6. Enregistrez le script modifié sous le nom **scriptweb11a** (menu Fichier/Enregistrer sous).
7. Paramétrez maintenant l'exécution pour générer un rapport limité.
8. Cliquez sur l'icône **Exéc** de la barre d'outils de l'Editeur pour lancer l'exécution. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
9. Ouvrez le fichier rapport (menu **Fichier/Ouvrir Rapport**) et étudiez les lignes générées.
10. Dans l'Editeur, sélectionnez à nouveau le menu **Paramétrer/Exécution** et décochez la case à cocher **Rapport** pour que les exécutions des scripts qui suivent dans ce manuel ne génèrent pas de rapport.

Création de votre propre fichier log :

Vous pouvez écrire vous-même une Sub écrivant les informations que vous désirez dans un fichier, voici un exemple :

```
sub log(msg_log$)
  local buffer$
  buffer$=Date$()+", "+hour$()+": "+min$()+": "+sec$()+ " --> "+msg_log$
  write(fichier_log$,buffer$,CRLF)
endsub
```

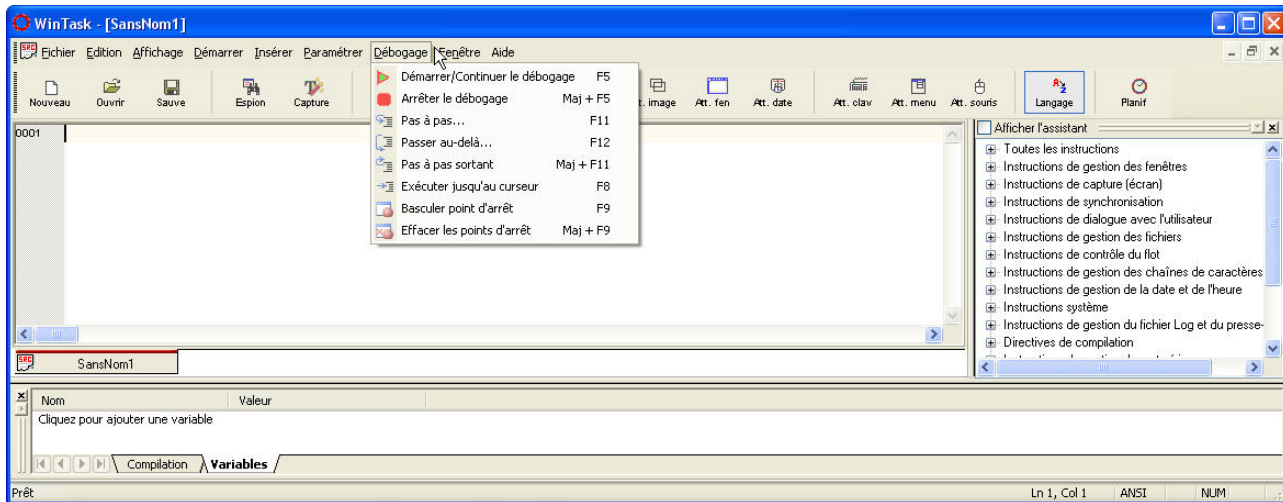
En reprenant le script écrit dans l'exercice 11, intégrez cette Sub dans le script (en début de script) et remplacez les deux lignes COMMENT par deux lignes appelant cette Sub. Ajoutez une ligne pour assigner la variable `fichier_log$` à un nom de fichier d'extension `.txt` et enregistrez le script ainsi modifié sous le nom **scriptweb11b**. Lancez l'exécution et ouvrez le fichier généré.

Exécution en mode Débogage

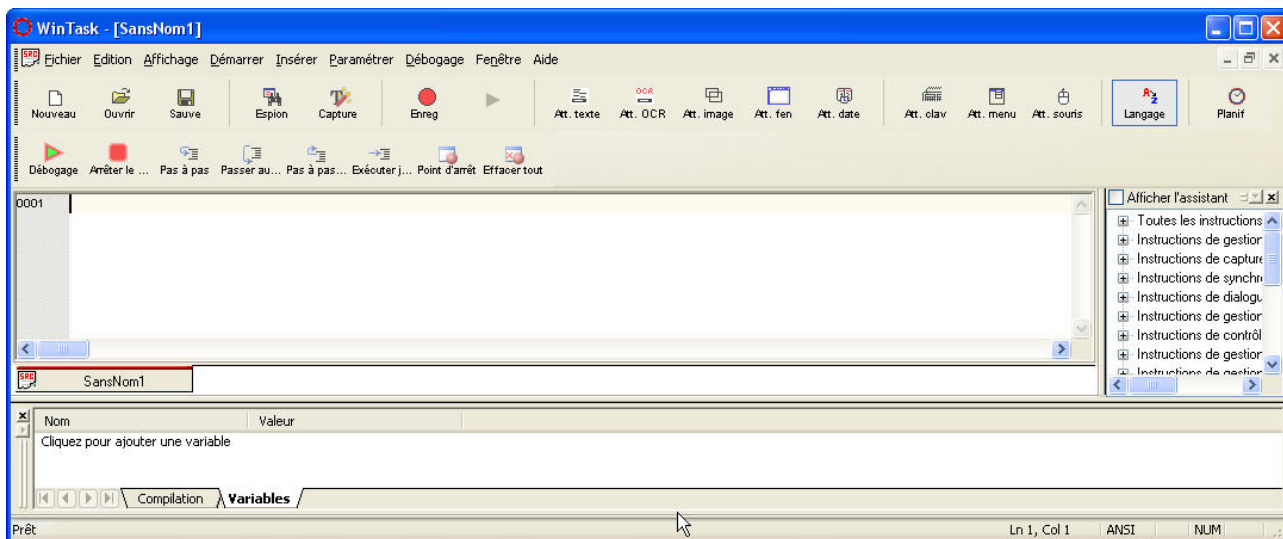
WinTask inclut un débogueur de scripts. En exécutant un script en mode débogage, vous pouvez arrêter l'exécution du script à une certaine ligne, afficher le contenu des variables à ce point d'arrêt, puis reprendre l'exécution jusqu'au point d'arrêt suivant que vous avez spécifié.

Toutes les fonctionnalités du mode Débogage sont accessibles depuis le menu **Débogage** dans la fenêtre de l'Editeur WinTask. Vous pouvez également faire afficher

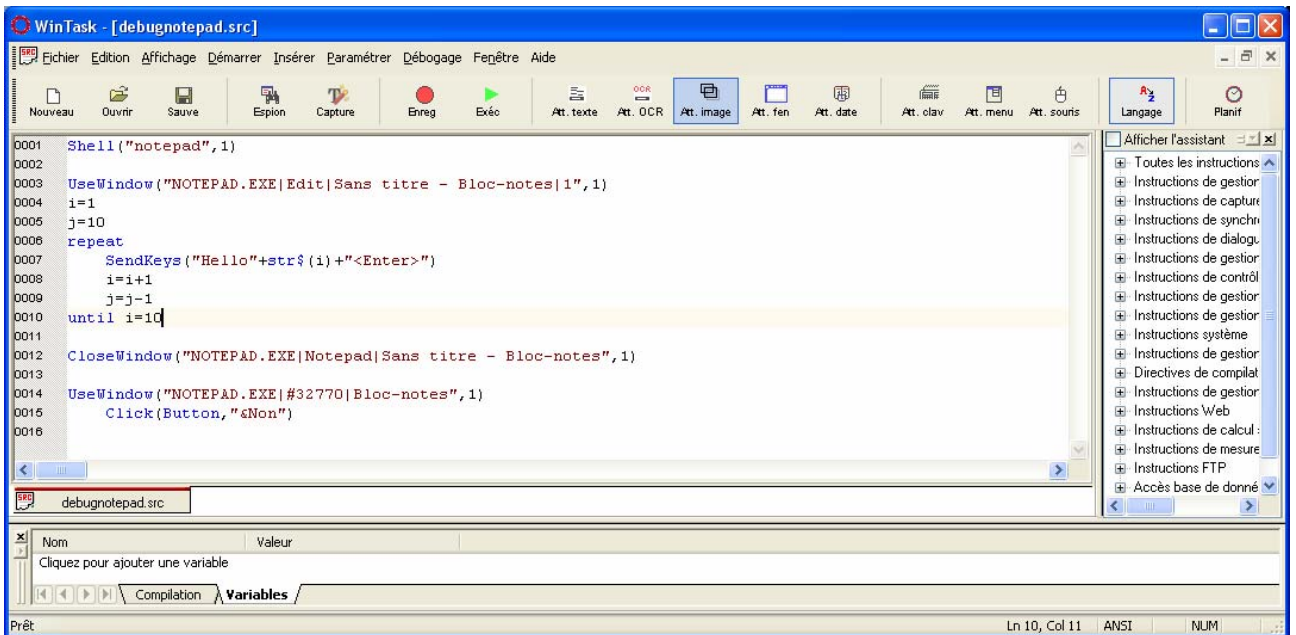
la barre d'outils Débogage en sélectionnant le menu Affichage/Barre d'outils Débogage.



La fenêtre de l'Editeur avec la barre d'outils Débogage :



Pour utiliser sur un exemple le mode Débogage, créez le petit script tel qu'affiché ci-dessous :

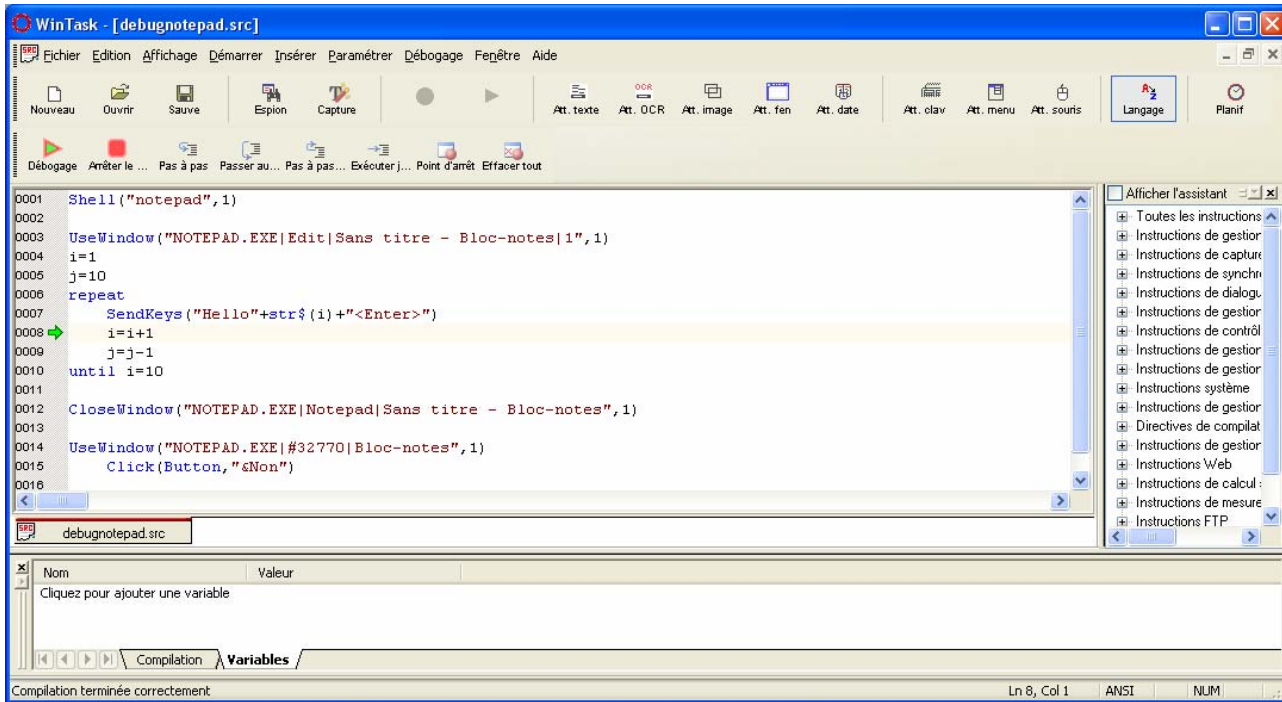


Ce script enregistré sous le nom "debugnotepad" lance notepad. Puis dans une boucle, il saisit dans la fenêtre notepad Hello1, puis Hello2, etc... jusqu'à ce que l'index i vaille 10. L'index j n'est pas utilisé, il est là juste pour illustrer le fonctionnement du mode Débogage. Notez que si vous utilisez Vista ou Windows 7, la fermeture de notepad ligne 14 utilise un nom de fenêtre différent et le bouton est Ne pas enregistrer au lieu de Non.

La manière la plus simple d'utiliser le mode Débogage est de mettre le curseur devant la ligne où l'exécution doit être suspendue, puis de visualiser le contenu des variables désirées à ce point d'arrêt. Donc mettez le curseur en début de ligne 8 et sélectionnez le menu **Débogage/Exécuter jusqu'au curseur**, ou cliquez l'icône **Exécuter**

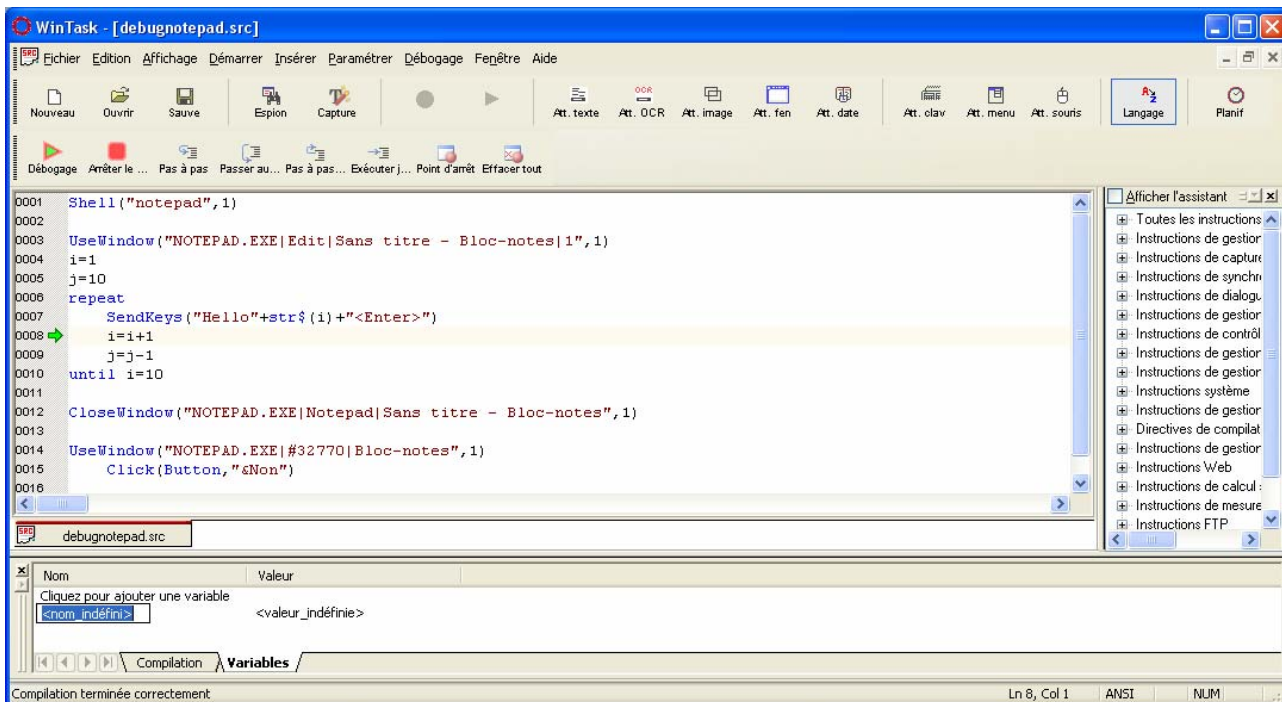
jusqu'au curseur dans la barre d'outils Débogage.

Dés que vous avez sélectionné l'exécution jusqu'au curseur, la fenêtre de l'Editeur est mise en réduction et notepad se lance. Vous voyez Hello1 tapé dans notepad, puis le script arrête son exécution et la fenêtre de l'Editeur revient au premier-plan avec une flèche verte sur la ligne où le script s'est arrêté :

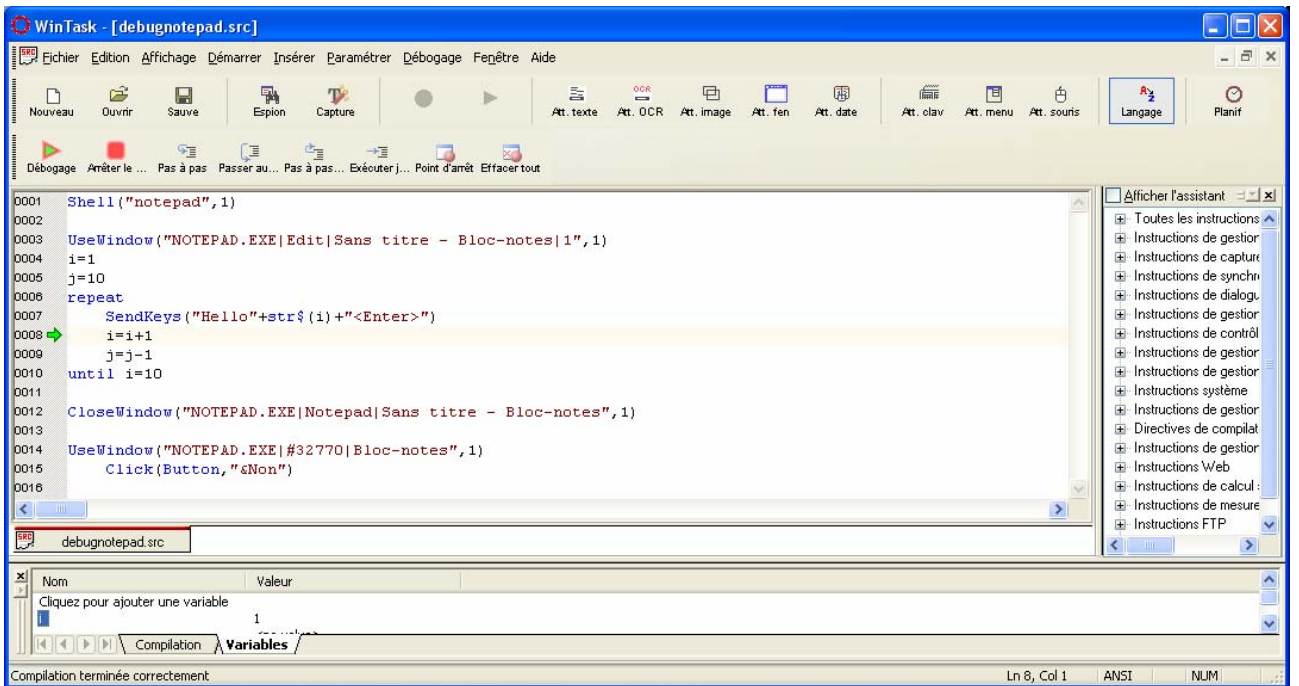


Avant de continuer l'exécution, vous voulez connaître la valeur de la variable i.

Dans la fenêtre Résultats de l'Editeur (la fenêtre en bas), cliquez sur **Cliquez pour ajouter une variable** :



Tapez **i** dans le champ <nom_undefined> et appuyez sur la touche **Entrée**, la valeur de i s'affiche :



Pour reprendre l'exécution du script, sélectionnez le menu **Débogage/Démarrer/Continuer le débogage** ou cliquez sur l'icône **Débogage**



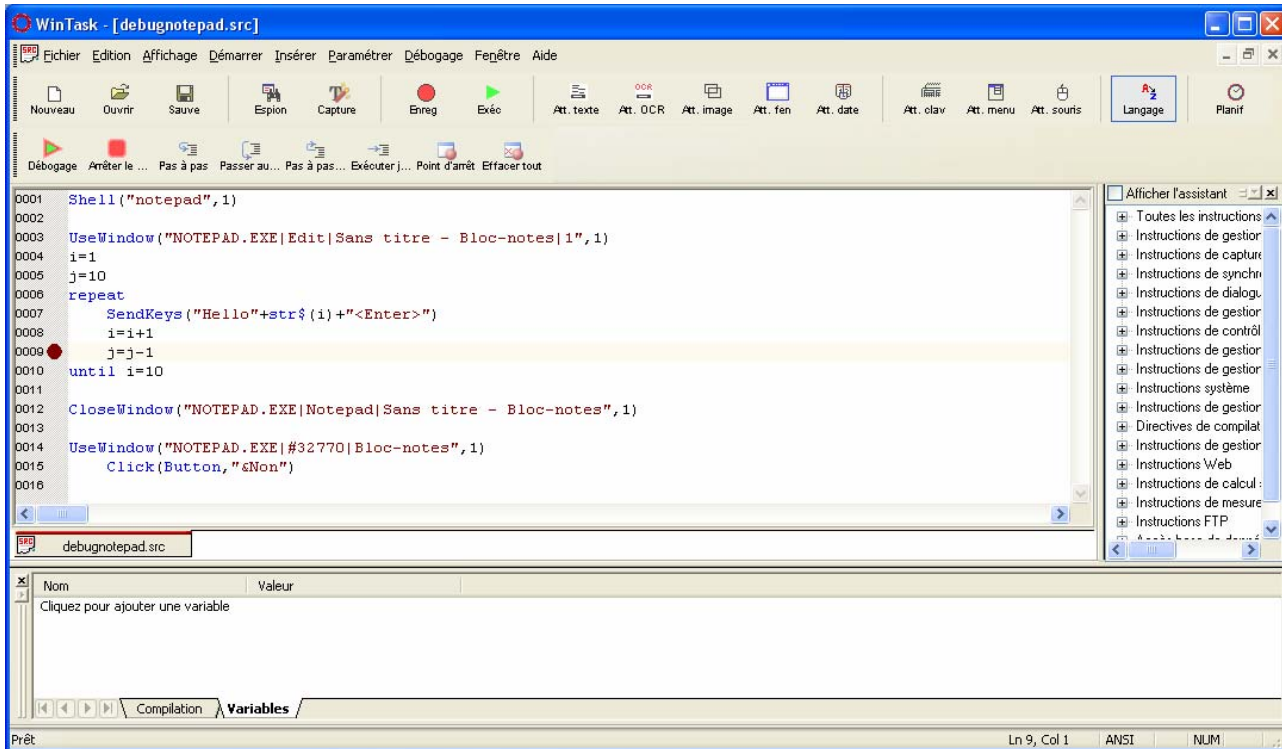
dans la barre d'outils Débogage.

Pour suspendre l'exécution plusieurs fois dans le script, utilisez les points d'arrêt. Par exemple, insérez un point d'arrêt à la ligne 9 : mettez le curseur en début de ligne 9 puis sélectionnez le menu **Débogage/Basculer point d'arrêt** ou cliquez sur l'icône



Point d'arrêt dans la barre d'outils Débogage.

La fenêtre de l'Editeur affiche un point rouge pour le point d'arrêt :

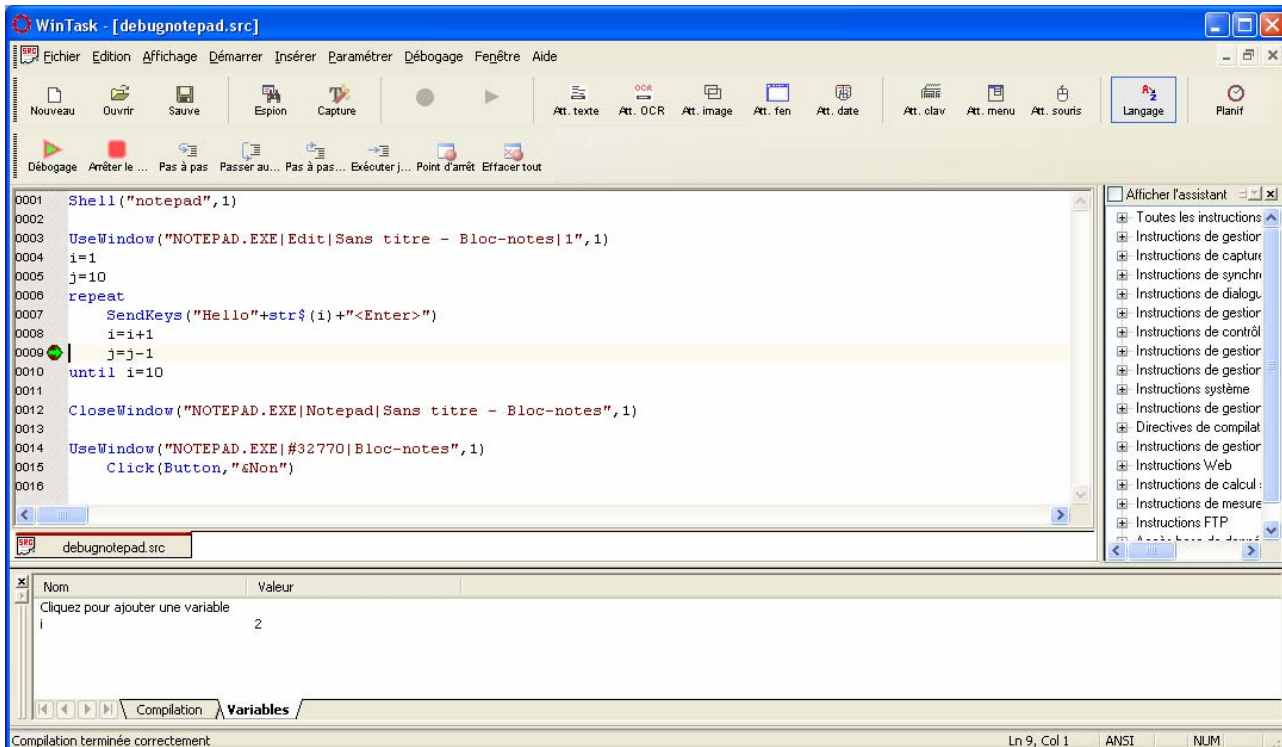


Démarrez alors l'exécution en mode Débogage : sélectionnez le menu **Débugage/Démarrer/Continuer le débogage** ou cliquez sur l'icône **Débugage**



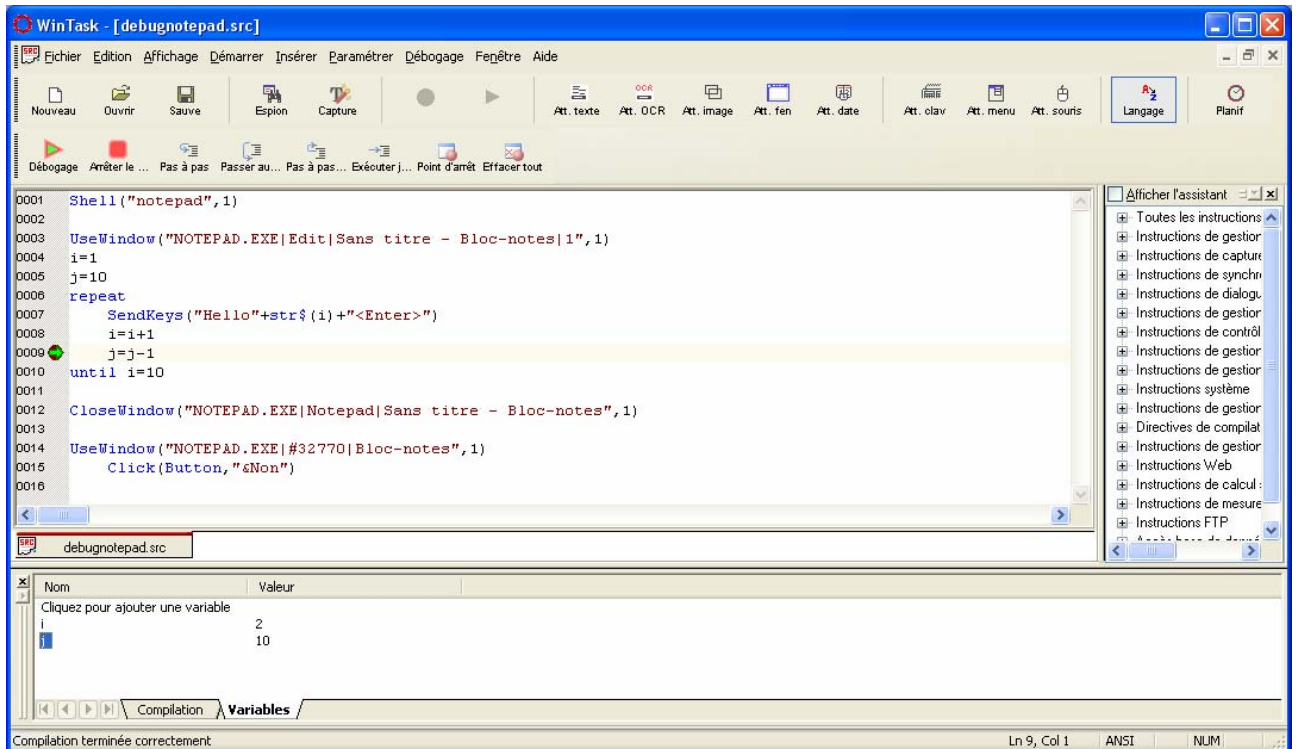
dans la barre d'outils de Débogage. Le script s'exécute jusqu'au point d'arrêt sans exécuter la ligne où est située le point d'arrêt.

Au point d'arrêt, la fenêtre de l'Editeur revient au premier-plan, la flèche verte est à la ligne 9, vous voyez la nouvelle valeur de la variable i qui est maintenant à 2 dans la fenêtre Résultats en bas.



Vous pouvez ajouter une nouvelle variable à visualiser à ce point d'arrêt en cliquant

sur Cliquez pour ajouter une variable, ajoutez la variable j :



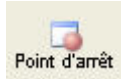
Poursuivez l'exécution avec le point d'arrêt toujours présent : cliquez sur l'icône



Débogage de la barre d'outils Débogage. A chaque itération, vous voyez la fenêtre de l'Editeur revenir au premier plan juste avant l'exécution de la ligne 9 et le contenu des variables i et j s'affiche.

Pour supprimer le point d'arrêt, mettez le curseur sur la ligne où se situe le point d'arrêt et sélectionnez le menu **Débogage/Basculer point d'arrêt** ou cliquez sur

l'icône **Point d'arrêt**



Pour arrêter l'exécution du script en mode débogage, sélectionnez le menu

Débogage/Arrêter le débogage ou cliquez sur l'icône Arrêter le débogage



Délai d'attente dépassé lors du chargement d'une page

Si ce message d'erreur est affiché lors de l'exécution d'une instruction StartBrowser, deux raisons sont possibles :

La page Web à charger met plus de temps que le délai maximum autorisé (la valeur par défaut du timeout est 30 secondes). Vous pouvez alors modifier ce timeout en mettant au-dessus de la ligne StartBrowser, l'instruction `#ActionTimeout=60` pour un délai maximum de 60 secondes. Si les autres pages du site se chargent plus rapidement, mettez l'instruction `#ActionTimeout=30` après la ligne StartBrowser pour revenir à la valeur par défaut de cette variable directive.

StartBrowser inclut un mécanisme de synchronisation automatique et donc l'instruction suivante ne peut être exécutée que si la page est complètement chargée. Si une fenêtre de type Alerte de Sécurité ou de type Saisie du mot de passe s'affiche lors du chargement de la page, la page ne peut pas se charger tant que vous n'avez pas saisi les informations demandées dans cette petite fenêtre, et donc le message d'erreur Délai d'attente dépassé s'affiche. Pour éviter ce message d'erreur, utilisez à la place de l'instruction StartBrowser, l'instruction *Shell* qui elle n'inclut pas le mécanisme de synchronisation. Vous pouvez générer automatiquement cette instruction en démarrant le mode Enregistrement, puis dans la boîte de dialogue "*Que voulez-vous démarrer avant de commencer l'enregistrement*", vous laissez cocher **Une application** et cliquez sur le bouton **OK**. Dans la boîte de dialogue suivante, vous cliquez sur le bouton **Parcourir** pour aller chercher IEXPLORE.EXE ou FIREFOX.EXE sur le disque et dans le champ Paramètres, vous spécifier l'adresse Web à lancer.

Si ce message d'erreur est affiché lors de l'exécution d'une instruction UsePage, ne fermez pas manuellement la page qui a provoqué l'erreur et comparez le titre de la page ayant provoqué l'erreur avec le titre tel que spécifié dans le script. Ajustez le titre spécifié dans le script pour refléter le titre tel qu'il est affiché lors du rejoue.

Une autre raison est que vraiment la page n'a pas pu se charger dans les 30 secondes, valeur par défaut du timeout, utilisez alors `#ActionTimeout` pour spécifier un délai d'attente plus long.

Une autre cause d'erreur possible est qu'au rejoue, la page qui aurait dû se charger ne l'a pas été car l'action précédente ne s'est pas effectuée correctement. Consultez **ClickHTMLElement ne génère pas d'erreur mais ne clique pas** ci-dessous.

ClickHTMLElement ne génère pas d'erreur mais ne clique pas

C'est une erreur "visuelle" : à l'exécution, vous voyez la souris cliquer quelque part, mais pas l'élément que vous pensiez. Et en conséquence, le UsePage suivant ce ClickHTMLElement échoue puisque le rejoue n'a pas chargé la bonne page.

Trois raisons possibles :

Le clic est fait mais pas tout à fait sur le petit lien. Utilisez les paramètres additionnels de ClickHTMLElement pour forcer un clic à la bonne place :

```
ClickHTMLElement("A[INNERTEXT= 'Télécharger version ']", left, 5, 5)
```

Pour cliquer sur l'objet HTML de nom *Télécharger version* mais avec un décalage de 5 pixels à gauche et en haut.

Le clic est fait à un tout autre endroit que prévu. C'est que le descripteur HTML spécifié dans le script n'est pas correct, modifiez-le en utilisant l'Espion et la rubrique d'aide Validation du contenu du mot clé CONTENT.

La position de l'objet HTML n'est pas complètement stabilisée même une fois la page complètement chargée. Pour forcer une synchronisation supplémentaire pour cet objet précis, utilisez la variable directive #HTMLPosRetry. Consultez **Saisie incorrecte dans un champ** ci-dessous.

Navigate, une solution de contournement

L'instruction *Navigate* peut remplacer un *ClickHTMLElement* sur un lien. A l'exécution, *Navigate* va directement à l'url spécifiée et inclut une synchronisation automatique.

Par exemple, sur www.wintask.fr, si vous enregistrez le clic sur **Téléchargez la version 30 jours**, cela génère la ligne :

```
ClickHTMLElement("A[INNERTEXT= 'Téléchargez la versi']")
```

Et ce clic envoie à la page **Wintask gratuit – Version 30 jours**.

Vous pouvez remplacer le ClickHTMLElement par une navigation directe à la page **Wintask gratuit** :

```
Navigate("http://www.wintask.fr/telecharger-wintask.php")
```

Saisie incorrecte dans un champ

Aucune erreur n'est renvoyée par le rejoue d'une instruction WriteHTML, mais vous voyez que le champ n'est pas correctement rempli. Cela se produit si la position de l'objet HTML (champ du formulaire) n'est pas complètement stabilisée même une fois la page complètement chargée.

Pour forcer une synchronisation supplémentaire pour cet objet précis, utilisez la variable directive #HTMLPosRetry. Avec cette variable à une valeur différente de 0 (sa valeur par défaut), la position de l'objet HTML est vérifiée une ou plusieurs fois avant d'écrire dedans.

Erreur élément HTML non trouvé

Ces erreurs se produisent lors de l'automatisation d'actions à l'intérieur d'une page Web affichée.

Il y a deux raisons possibles pour avoir un élément HTML non trouvé quand le script exécute une instruction dont un paramètre est un descripteur HTML (ClickHTMLÉlement, WriteHTML, CaptureHTML, CaptureTableHTML, GetHTMLÉditText, OverHTMLÉlement,...) : cela peut être dû à un problème de synchronisation, l'élément à cliquer n'est pas encore affiché - une deuxième raison peut être que l'élément cliqué lors de l'enregistrement a changé de nom lors du rejoue (son descripteur HTML a changé). Pour comprendre d'où vient l'erreur et la corriger, suivez ces étapes :

1. Lancez l'exécution du script.
2. Quand l'exécution s'arrête en affichant un message d'erreur, cliquez sur le bouton **OK** du message d'erreur. Ne fermez pas la fenêtre du navigateur.
3. Si vous voyez que la page ne s'est pas complètement chargée, l'objet HTML n'a pas été trouvé à cause d'un problème de synchronisation. Vous pouvez alors corriger en suivant les recommandations sur Délai d'attente dépassé plus haut.
4. Si la page Web est bien celle désirée et qu'elle est bien complètement chargée, le descripteur HTML spécifié dans le script n'est peut-être pas correct. Restaurez alors la fenêtre de l'Editeur WinTask et créez un nouveau script.
5. Démarrez le mode Enregistrement. Sur la boîte de dialogue *Que voulez-vous démarrer avant de commencer l'enregistrement*, cochez le bouton radio **Rien** et cliquez sur le bouton **OK**.
6. Revenez à la fenêtre du navigateur où est affichée la page avec l'objet HTML non trouvé et cliquez sur l'objet qui pose problème.
7. Arrêtez le mode Enregistrement, un petit script a été généré automatisant juste l'étape qui pose problème. Manuellement, revenez à la page où le petit script doit trouver l'objet HTML.
8. Exécutez ce petit script, la compilation se lance après avoir enregistré le script, puis l'action se rejoue.
9. Si le petit script se rejoue sans erreur, comparez le descripteur HTML tel qu'il a été enregistré dans le petit script et celui dans le script en erreur. S'il est différent, modifiez dans le script en erreur. Si l'élément est identique dans les deux scripts, insérez dans le script en erreur une synchronisation correcte afin que l'élément ait bien le temps d'être affiché avant que le script ne clique dessus.

La capture d'une Table ne renvoie pas les données attendues


Les instructions de Capture utilisent le contenu de l'objet HTML à capturer pour identifier ce qu'il faut capturer. C'est le mot-clé CONTENT, par exemple une instruction de capture du texte se trouvant dans un paragraphe s'écrit ainsi, si le premier mot du paragraphe est 2010 :

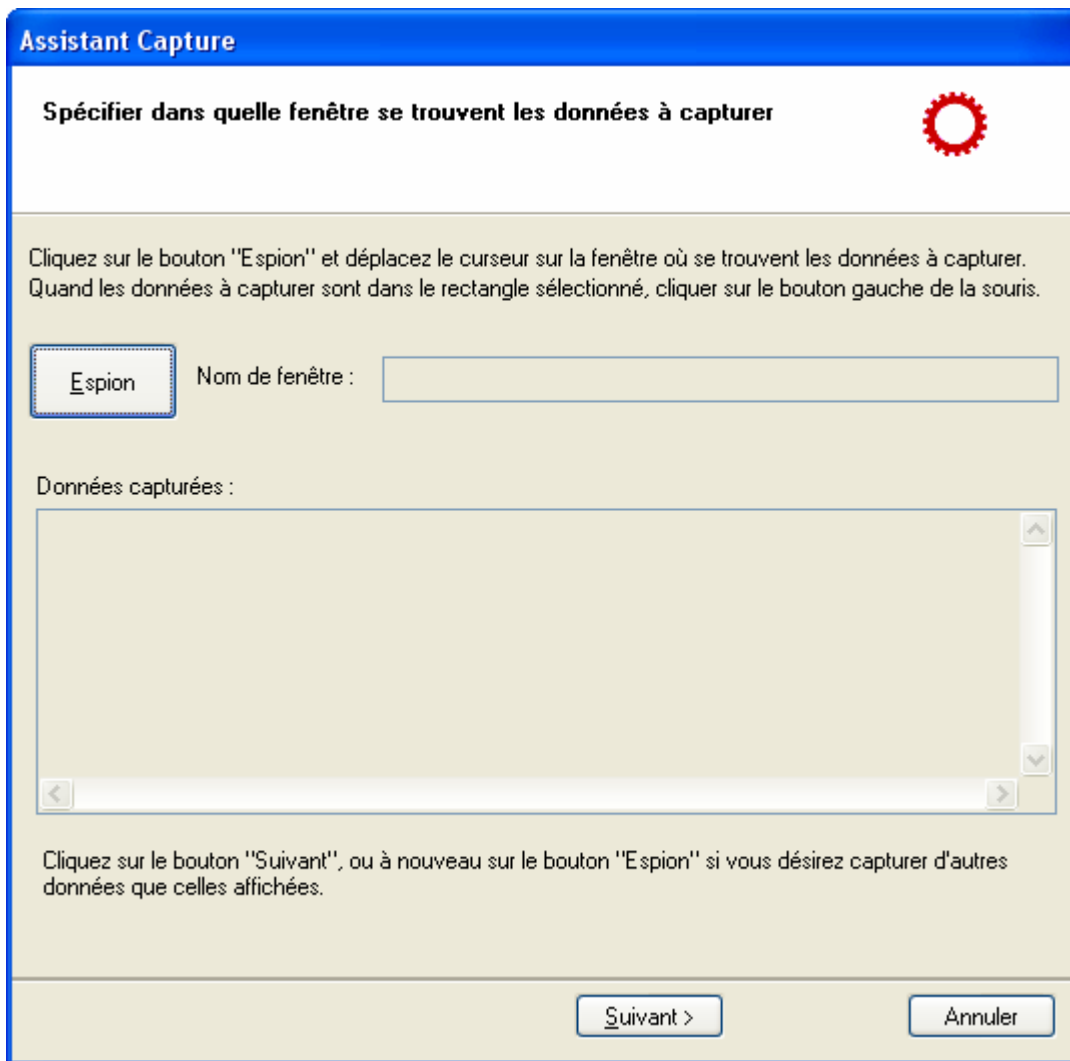
```
CaptureHTML("P[CONTENT='2010']",texte_capture$)
```

Donc le descripteur HTML de paragraphes ou de tables repose sur le premier mot contenu dans le paragraphe ou la table. Mais parfois, ce mot n'est pas suffisant pour identifier de façon unique l'objet HTML, ou ce mot varie sur la page affichée.

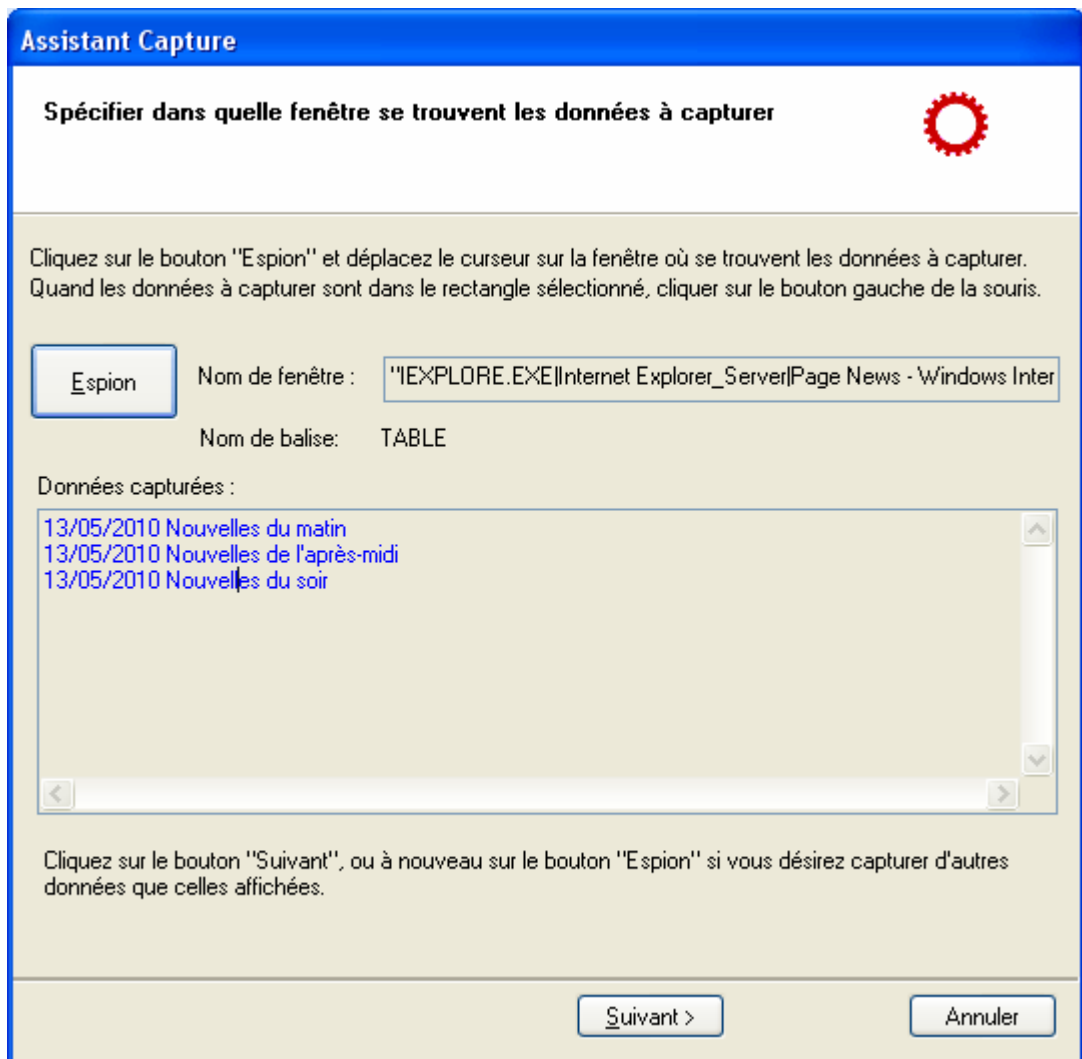
L'exercice suivant illustre comment modifier le contenu du mot-clé CONTENT pour qu'un script de capture retrouve correctement les objets à capturer. La solution de l'exercice 12 est donnée en appendice C.

Exercice 12 : Modification du mot-clé CONTENT pour une capture correcte

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Nouveau** de la barre d'outils de l'Editeur.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez la case **Internet Explorer** et cliquez sur le bouton **OK**.
3. La boîte de dialogue **Démarez Internet Explorer** s'affiche. Dans le champ **Adresse Web**, tapez **www.wintask.fr/demos/page-news.htm** et cliquez sur le bouton **OK**.
4. La page de titre **Page News** s'affiche. Cette page simule une page de nouvelles avec différents paragraphes de nouvelles et le texte de ces nouvelles varie au long de la journée. Le script doit capturer les différentes nouvelles affichées.
5. Dans la barre d'outils WinTask flottante, cliquez sur l'icône de Capture . Le mode Enregistrement est momentanément arrêté.



6. Cliquez sur le bouton **Espion** de l'écran Assistant Capture. Le curseur souris prend la forme d'une loupe.
7. Déplacez le curseur souris sur le tableau à capturer commençant à la date 13/05/2010, un rectangle noir entoure le texte du paragraphe quand le curseur souris est dessus. Cliquez et les données capturées s'affichent dans l'écran **Assistant Capture**.



8. Cliquez sur le bouton **Suivant**. La boîte de dialogue **Spécifier dans quel objet HTML se trouvent les données à capturer** s'affiche et propose un Descripteur HTML pour identifier le paragraphe à capturer, le descripteur HTML proposé est "P[CONTENT='13']".

Validation du contenu du mot-clé CONTENT

Descripteur
"TABLE[CONTENT='13']"

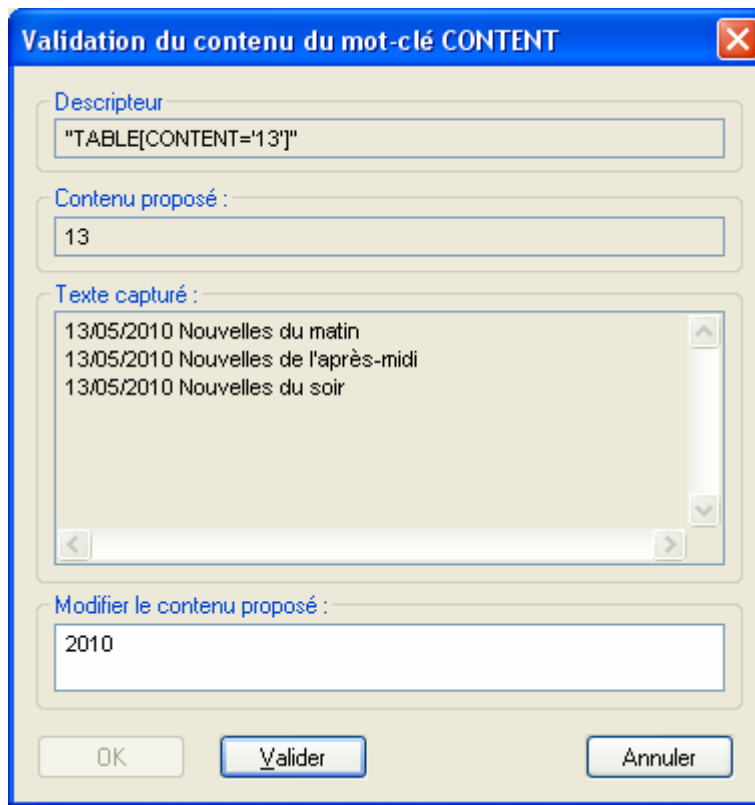
Contenu proposé :
13

Texte capturé :
13/05/2010 Nouvelles du matin
13/05/2010 Nouvelles de l'après-midi
13/05/2010 Nouvelles du soir

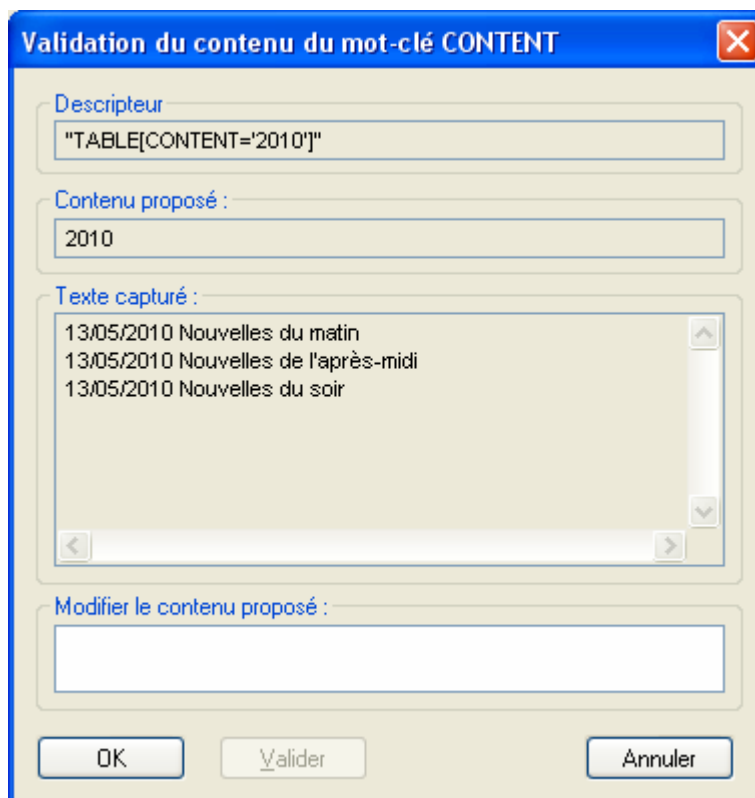
Modifier le contenu proposé :

OK Valider Annuler

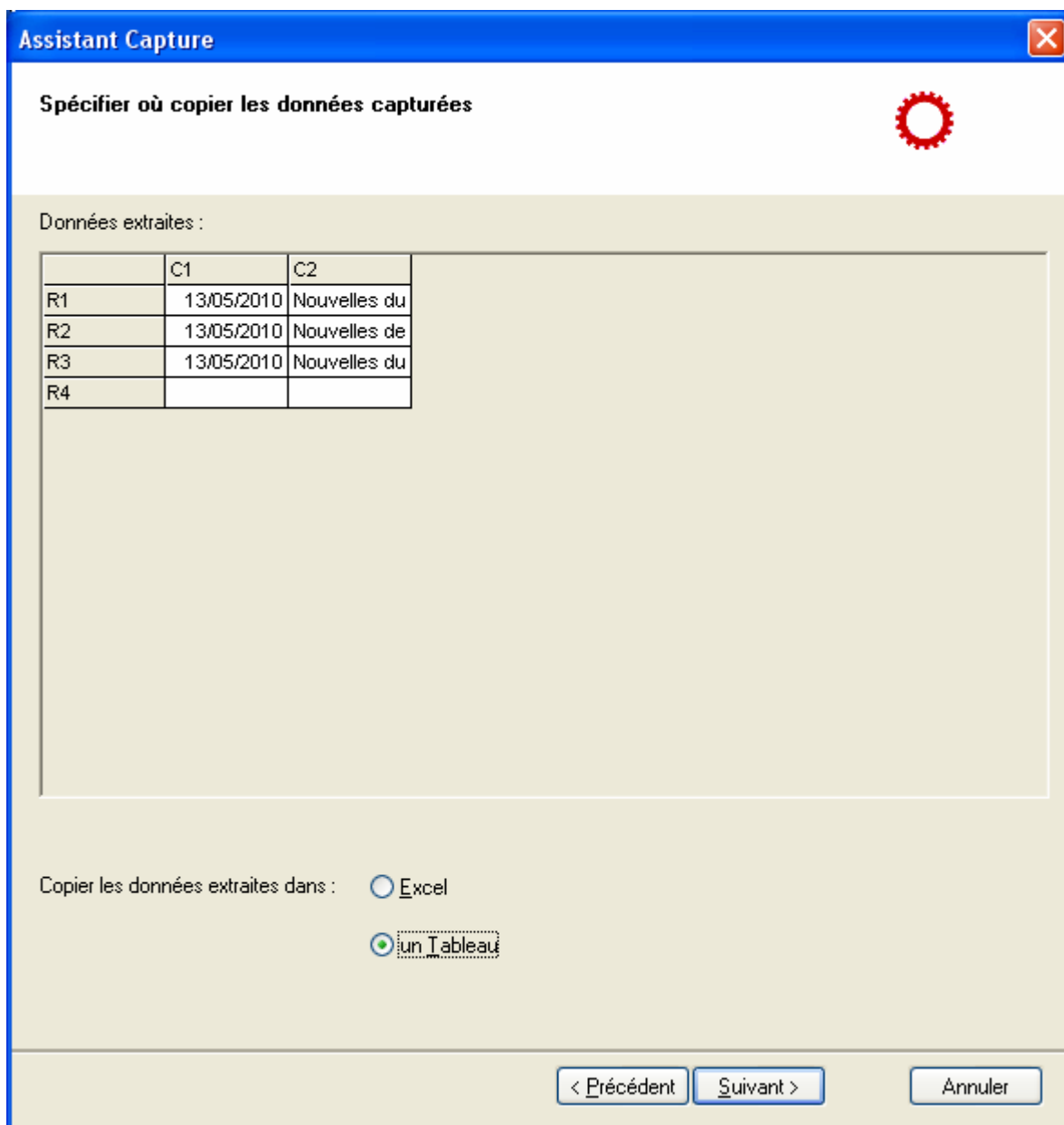
10. Vous devez choisir dans le Texte capturé une chaîne de caractères qui ne varie pas quand les nouvelles varient. La chaîne qui ne change qu'une fois par an est l'année, c'est donc la chaîne que vous allez utiliser comme contenu du mot-clé CONTENT. A l'aide de la souris, copiez **2010** du champ **Texte capturé** et collez-le dans le champ **Modifier le contenu proposé**. Le bouton **Valider** devient actif.



11. Cliquez sur le bouton **Valider** pour que WinTask vérifie que cette chaîne est correcte pour identifier de manière unique ce paragraphe. Le champ Descripteur contient alors le nouveau descripteur.



12. Cliquez sur le bouton **OK**. Vous revenez à l'écran Spécifier dans quel objet HTML se trouvent les données à capturer et le champ Descripteur HTML reflète le changement effectué à l'étape précédente.



15. Cliquez sur le bouton **Coller dans le script**. Arrêtez le mode Enregistrement, la fenêtre de l'Editeur s'affiche et vous voyez que deux lignes ont été générées en début de script pour déclarer les tableaux utilisés par la capture, plus trois lignes pour effectuer la capture :

```
UsePage("Page News")
```

```
ret = CaptureTableHTML("TABLE[CONTENT='2010']", "R1C1:R4C1", tabcell_0$())
```

```
ret = CaptureTableHTML("TABLE[CONTENT='2010']", "R1C2:R4C2", tabcell_1$())
```

La première ligne UsePage permet d'attendre que la page soit bien chargée avant d'effectuer la capture. Les deux lignes suivantes correspondent à la capture proprement dite, le résultat de la capture est mis dans deux tableaux de type chaîne de caractères dont le nom par défaut est tabcell_x\$. Remplacez les noms par *datejour\$* et *textenews\$*.

16. Fermez la fenêtre Internet Explorer.
17. A la fin du script, écrivez deux lignes msgbox pour afficher le contenu de la première valeur de chacun des deux tableaux.
18. Cliquez sur l'icône **Exéc** de la barre d'outils de l'Editeur pour lancer l'exécution. Donnez comme nom au script **scriptweb12** quand la boîte de dialogue **Enregistrer sous** s'affiche. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau. Les trois paragraphes capturés s'affichent successivement.

ASTUCE : L'année 2010 est utilisée dans CONTENT, donc le script ne fonctionnera plus en 2011. Vous pouvez variabiliser le descripteur HTML en utilisant l'instruction WinTask Year\$() qui renvoie l'année en cours. Voici ce que devient la ligne CaptureHTML pour l'index 3 :

```
ret = CaptureTableHTML("TABLE[CONTENT=""+"year$()+"""], "R1C1:R4C1",
datejour$())
```

Option de Menu de Internet Explorer non sélectionnée

Au rejoue, un ChooseMenu devant sélectionner une option dans un menu, ne sélectionne pas l'option désirée. Ou le mode Enregistrement n'a pas généré l'instruction. Cela se produit notamment si Internet Explorer est en mode Protégé, ce qui interdit à WinTask l'accès aux ressources du menu.

Vous pouvez alors utiliser les touches raccourci :

Par exemple dans notepad, le menu Fichier **Fichier/Enregistrer** :

```
UseWindow("""NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes")
SendKeys("<Alt f>",Noactivate)
pause 5 ticks
SendKeys("e",NoActivate)
```

Le nom de fenêtre spécifié par l'instruction UseWindow doit être le nom de la fenêtre menu (utilisez comme d'habitude l'Espion pour trouver le nom), et notez à nouveau la ligne Pause pour que le menu Fichier ait le temps de s'ouvrir avant d'envoyer la deuxième touche de raccourci.

Option de Menu d'une page Web non sélectionnée

De plus en plus de pages Web utilisent des "over" menus : les options du menu principal ne s'affichent que lorsque la souris passe au-dessus du menu principal.

L'instruction *OverHTMLElement* de WinTask permet d'automatiser ce passage de la souris au-dessus d'une option de menu. En mode Enregistrement, mettez la souris sur l'option de menu, appuyez un bref instant sur la touche Majuscules puis cliquez sur l'option du sous-menu que vous devez sélectionner.

Voici un exemple :

```
StartBrowser("IE", "www.wintask.fr", 3)
UsePage("Macro et Capture de données avec WinTask, le logiciel d'automatisation
pour Windows et Internet")
OverHTMLElement("A[INNERTEXT= 'Produit']")
ClickHTMLElement("A[INNERTEXT= 'Fonctions OCR']")artBrowser("IE", "")
```

Le mode Enregistrement peut générer plusieurs lignes *OverHTMLElement* identiques si vous appuyez trop longtemps sur la touche Majuscules. Supprimez alors les lignes en trop pour n'en garder qu'une.

La sélection d'une option d'un menu contextuel ne se rejoue pas

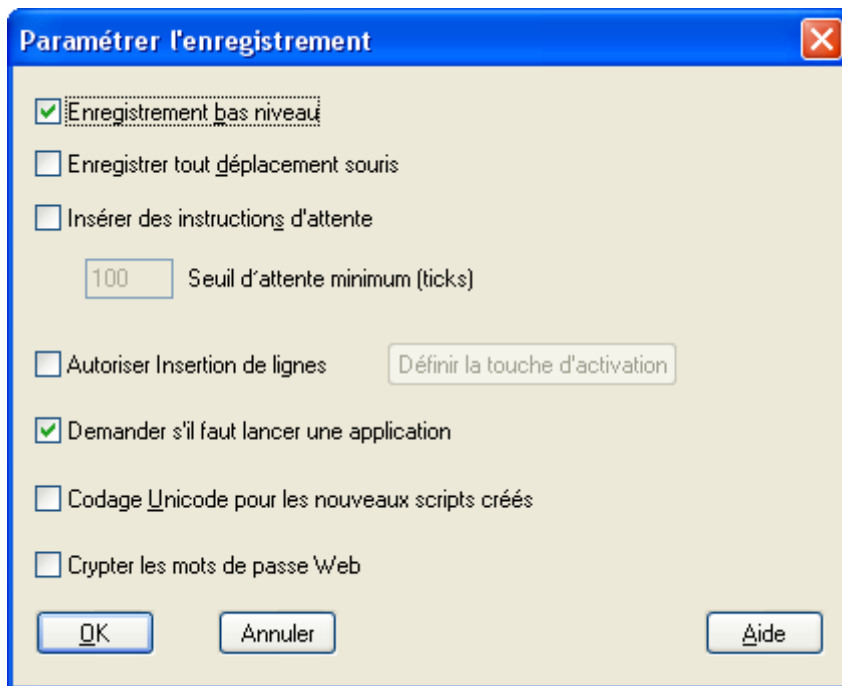
WinTask inclut deux instructions *SaveTargetAs* et *SavePictureAs* qui sont à utiliser plutôt que les menus contextuels **Enregistrer la cible sous** et **Enregistrer l'image sous**.

Enregistrement Bas Niveau quand rien d'autre ne marche

Avec cet autre mode d'Enregistrement, les actions sont enregistrés par des clics souris et donc l'enregistrement n'est plus orienté objet.

N'utilisez ce mode que pour enregistrer les quelques actions qui ne sont pas rejouées correctement si elles ont été enregistrées en mode objet.

Configurez l'Enregistrement en bas niveau en sélectionnant le menu **Paramétrer/Enregistrement**.



Par exemple, voici ce que génère l'enregistrement du menu Fichier/Ouvrir de notepad, d'abord en normal, puis en bas niveau :

```
Shell("notepad", 1)
```

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
  ChooseMenu(Normal, "&Fichier/&Ouvrir... Ctrl+O")
```

```
UseWindow("NOTEPAD.EXE/#32770/Ouvrir", 1)
  Click(Button, "Annuler")
```

```
CloseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
```

En bas niveau :

```
Shell("notepad", 1)
```

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1, NoActivate)
  ClickMouse(Left, Down, 5, 38)
  ClickMouse(Left, Up, 5, 38)
  ClickMouse(Left, Down, 33, 81)
  ClickMouse(Left, Up, 33, 81)
```

```
UseWindow("NOTEPAD.EXE/Button/Annuler!", 1, NoActivate)
  ClickMouse(Left, Down, 50, 14)
  ClickMouse(Left, Up, 50, 14)
```

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
  ClickMouse(Left, Down, 1082, 6)
  ClickMouse(Left, Up, 1082, 6)
```

Les coordonnées souris générées sont différentes d'un PC à un autre.

Le rejoue est lent

Un script écrit en utilisant Internet Explorer 6 se rejoue plus lentement sous Internet Explorer 8 ou 9. En effet les noms de fenêtre Internet Explorer ne sont pas les mêmes dans ces deux versions d'Internet Explorer.

Par exemple, la plupart des scripts écrits dans ce manuel se terminent par une instruction de fermeture de la fenêtre Internet Explorer. Sous IE6, l'instruction est :

```
CloseWindow("IEXPLORE.EXE/IEFrame/WinTask - Microsoft Internet Explorer", 1)
```

Sous IE8/IE9, l'instruction est :

```
CloseWindow("IEXPLORE.EXE/IEFrame/WinTask - Windows Internet Explorer", 1)
```

Au rejoue, WinTask recherche d'abord une fenêtre de nom exactement celui spécifié dans CloseWindow, puis au bout d'environ 3 secondes (valeur par défaut issue de la valeur par défaut de la variable directive #ActionTimeout), commence à tronquer le nom de fenêtre en commençant par la droite.

Si vous tronquez manuellement dans le script le nom de fenêtre pour ne pas tenir compte de la fin, la fenêtre sera trouvée immédiatement :

```
CloseWindow("IEXPLORE.EXE/IEFrame/WinTask", 1)
```

Vous pouvez appliquer la même technique si une instruction *UseWindow* se rejoue lentement.

Deux autres variables directives influent sur la vitesse d'exécution :

```
#HTMLPosRetry
```

```
#ExecuteDelay
```

Consultez l'aide sur ces deux variables.

L'Assistant de Capture ne permet pas de capturer les données désirées

Si pour une raison quelconque, l'Assistant de Capture ne répond pas à votre besoin, vous pouvez utiliser l'instruction *GetPageSource\$* pour récupérer la page HTML source et extraire du code HTML les données que vous désirez. L'instruction *ExtractBetween\$* permet facilement cette extraction.

Si la page contient des frames, vous pouvez utiliser l'instruction *GetFramesource\$*.

Conclusion

Avec ces quelques exemples, vous avez pu constater l'extraordinaire flexibilité de WinTask pour automatiser tout type de tâches sur des sites Web.

Nous avons également indiqué nos astuces les plus courantes pour développer des scripts qui se jouent de manière fiable. Le point le plus important est de rendre le script le plus indépendant possible de l'environnement, si le fichier existe ou pas, si le répertoire courant a changé, ...

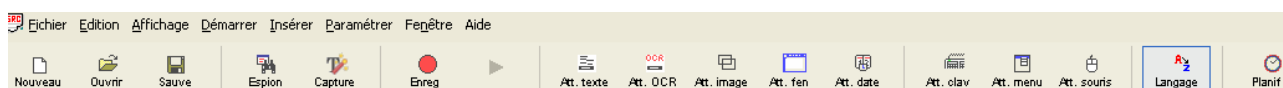
Nous n'avons pas traité les mêmes aspects pour l'automatisation de applications Windows, vous pouvez consulter le manuel dédié aux applications Windows à l'adresse <http://www.wintask.fr/manuels-wintask.php>








Nous vous remercions de nous envoyer vos remarques sur ce manuel à info@wintask.fr



APPENDICE A

Barre d'outils de l'Editeur WinTask

La barre d'outils de l'Editeur WinTask permet d'accéder aux fonctionnalités listées ci-dessous. Les icônes présentées sont les grandes icônes (menu **Affichage/Grandes icônes**).



 Nouveau	Crée un nouveau script. Une nouvelle fenêtre s'ouvre.
 Ouvrir	Ouvre un script existant. Le script est ouvert dans une nouvelle fenêtre de l'Editeur.
 Sauve	Enregistre le script. Le script affiché dans la fenêtre courante est sauvé.
 Espion	Démarre l'outil Espion. L'Espion permet de trouver les noms des fenêtres se trouvant sur le bureau Windows.
 Capture	Démarre l'Assistant de Capture. Il permet de capturer le texte affiché dans une fenêtre ou le contenu d'objets HTML.
 Enreg	Démarre le mode Enregistrement. Toutes les actions utilisateurs seront désormais enregistrées afin de générer un script WinTask dans la fenêtre courante de l'Editeur.
 Exéc	Compile et lance l'exécution. Le script affiché dans la fenêtre courante est sauvegardé, puis compilé et si aucune erreur de compilation n'est trouvée, l'exécution est lancée.
 Att. texte	Synchronisation Texte. Appelle l'assistant de synchronisation Texte. Les lignes générées par cet assistant sont insérées dans le script courant.
 Att. OCR	Synchronisation Texte OCR. Appelle l'assistant de synchronisation Texte OCR. Les lignes générées par cet assistant sont insérées dans le script courant.
 Att. image	Synchronisation Image. Appelle l'assistant de synchronisation Image. Les lignes générées par cet assistant sont insérées dans le script courant.
 Att. fen	Synchronisation Fenêtre. Appelle l'assistant de synchronisation Fenêtre. Les lignes générées par cet assistant sont insérées dans le script courant.

	<p>Synchronisation Date/Heure. Appelle l'assistant de synchronisation Date/Heure. Les lignes générées par cet assistant sont insérées dans le script courant.</p>
	<p>Attente action Touche. Appelle l'assistant de l'attente clavier. Les lignes générées par cet assistant sont insérées dans le script courant.</p>
	<p>Attente action Menu. Appelle l'assistant de l'attente menu. Les lignes générées par cet assistant sont insérées dans le script courant.</p>
	<p>Attente action Souris. Appelle l'assistant de l'attente souris. Les lignes générées par cet assistant sont insérées dans le script courant.</p>
	<p>Affiche la liste des instructions du langage. La liste est affichée dans une fenêtre à droite dans l'Editeur.</p>
	<p>Démarre le Planificateur WinTask. Le Planificateur permet de lancer l'exécution de scripts à une certaine date et heure. Le Planificateur est capable d'ouvrir un bureau Windows, de lancer l'exécution et de fermer le bureau (le Planificateur n'est pas disponible sous Vista/Windows 7/2008).</p>




APPENDICE B

Barre d'outils WinTask flottante

En mode Enregistrement, l'Editeur WinTask est mis en réduction et une barre d'outils flottante s'affiche. La barre d'outils WinTask flottante permet d'accéder aux fonctionnalités listées ci-dessous :



	Arrête le mode Enregistrement. Et la fenêtre de l'Editeur est restaurée.
	Démarre l'outil Espion. Lance l'outil Espion sans quitter le mode Enregistrement.
	Démarre l'Assistant de Capture. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation Texte. Appelle l'assistant de synchronisation Texte. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation Texte OCR. Appelle les différentes instructions OCR incluses dans WinTask.
	Synchronisation Image. Appelle l'assistant de synchronisation Image. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation Fenêtre. Appelle l'assistant de synchronisation Fenêtre. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation sur Durée. Insère une pause d'une certaine durée. Cette instruction Pause est collée dans le script en cours.
	Synchronisation Date/Heure. Appelle l'assistant de synchronisation sur Date/Heure. Les lignes générées par cet assistant sont collées dans le script en cours.
	Attente clavier. Appelle l'assistant d'attente clavier. Les lignes générées par cet assistant sont collées dans le script en cours.

	Attente menu. Appelle l'assistant d'attente menu. Les lignes générées par cet assistant sont collées dans le script en cours.
	Attente souris. Appelle l'assistant d'attente souris. Les lignes générées par cet assistant sont collées dans le script en cours.
	Insérer des lignes. Appelle la boîte de dialogue Insérer des lignes dans le script qui permet d'insérer pendant l'Enregistrement des instructions autres que celles générées automatiquement.

APPENDICE C

Solutions des exercices

Les solutions aux exercices de programmation sont fournies ci-dessous.

Exercice 3, Scriptweb03.src

```
'=====
' Scriptweb03
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 3
'
' Les instructions UsePage sont ajoutées manuellement pour assurer une synchronisation correcte
' alors que toutes les pages ont le même titre.
'=====

StartBrowser("IE", " www.wintask.fr/demos/pageidentique1.htm", 3)

UsePage("Page Titre identique")
    ClickHTMLElement("A[INNERTEXT= 'ici']")

' UsePage ajouté ici pour attendre que la nouvelle page de titre Page Titre identique soit chargée.
UsePage("Page Titre identique")
    ClickHTMLElement("A[INNERTEXT= 'page précédente']")

' UsePage ajouté ici pour attendre que la nouvelle page de titre Page Titre identique soit chargée.
UsePage("Page Titre identique")
    WriteHTML("INPUT TEXT[NAME= 'nom']", "Durand")

CloseWindow("IEXPLORE.EXE|IEFrame|Page Titre identique - Windows Internet Explorer",1)
```

Exercice 4, Scriptweb04.src

```
=====
' Scriptweb04
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 4
'
' Utilisation d'une synchronisation Texte OCR quand toutes les pages ont le même titre,
' et que la page à attendre ne contient pas d'objet HTML sur lequel se synchroniser.
=====
```

```
StartBrowser("IE", "www.wintask.fr/demos/pageidentique1.htm")
```

```
UsePage("Page Titre identique")
    ClickHTMLElement("A[INNERTEXT= 'ici']")
```

```
Ret=UseOCREngine(2)
```

```
Pause 20 secs until
```

```
    TextOCR("cette")
    InWindow("IEXPLORE.EXE|Internet Explorer_Server| Page Titre identique - Windows Internet
Explorer|1",1)
    InArea(387,181,23,229)
```

```
PauseFalse
```

```
    MsgBox("La page ne s'est pas chargée en moins de 20 secondes !")
    End
```

```
EndPause
```

```
msgbox("le texte de la page est maintenant affiché")
```

```
    ClickHTMLElement("A[INNERTEXT= 'page précédente']")
```

```
' Pour fermer la fenêtre du navigateur quelle que soit la version d'Internet Explorer ou de Firefox,
' remplacez le CloseWindow généré par le mode Enregistrement par l'instruction CloseBrowser
CloseBrowser()
```

Exercice 5, Scriptweb05a.src

```
=====
' Scriptweb05a
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 5, partie 1
'
' Ce script met en œuvre une boucle pour cliquer sur les numéros de page
' représentant chacun un lien pour aller à la page considérée.
=====

StartBrowser("IE", "www.wintask.fr/demos/page-iteration1.htm", 3)

i = 2

While i <= 7
    UsePage("Page Itération 1")
    ' La syntaxe de la chaîne de caractères représentant le descripteur HTML
    ' pour chacun des numéros de page doit être respectée :
    ' le début de la chaîne est A[INNERTEXT= ' mis entre " puisque c'est une chaîne
    ' puis on concatène str$(i), l'index i étant transformé en chaîne via l'instruction str$
    ' et la fin de chaîne est ']' ces deux caractères étant mis entre ", puisque c'est une chaîne.
        ClickHTMLElement("A[INNERTEXT= '"+str$(i)+"']")

    ' Le titre de la page se construit également par concaténation de chaînes.
    UsePage("Page Itération "+str$(i))
        ClickHTMLElement("A[INNERTEXT= 'ici']")

    i = i + 1
Wend

' Pour fermer la fenêtre du navigateur quelle que soit la version d'Internet Explorer ou de Firefox,
' remplacez le CloseWindow généré par le mode Enregistrement par l'instruction CloseBrowser
CloseBrowser()
```

Exercice 5, Scriptweb05b.src

```
=====
' Scriptweb05b
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 5, partie 2
'
' Ce script met en œuvre une boucle pour cliquer sur les numéros de page
' représentant chacun un lien pour aller à la page considérée.
=====

' Comme toutes les pages commencent par le même titre
' Page Itération,
' il vaut mieux utiliser #UsePageExact=1 pour forcer la reconnaissance exacte des titres de page.
#UsePageExact = 1

StartBrowser("IE", " www.wintask.fr/demos/page-iteration1.htm" ,3)

i = 2

repeat
  UsePage("Page Itération 1")
  ' La syntaxe de la chaîne de caractères représentant le descripteur HTML
  ' pour chacun des numéros de page doit être respectée :
  ' le début de la chaîne est A[INNERTEXT= ' mis entre " puisque c'est une chaîne
  ' puis on concatène str$(i), l'index i étant transformé en chaîne via l'instruction str$
  ' et la fin de chaîne est ' ] ces deux caractères étant mis entre ", puisque c'est une chaîne.
  ClickHTMLElement("A[INNERTEXT= '"+str$(i)+"']")

  ' Le titre de la page se construit également par concaténation de chaînes.
  UsePage("Page Itération "+str$(i))
  ClickHTMLElement("A[INNERTEXT= 'ici']")

  i = i + 1
until i = 8

' Pour fermer la fenêtre du navigateur quelle que soit la version d'Internet Explorer ou de Firefox,
' remplacez le CloseWindow généré par le mode Enregistrement par l'instruction CloseBrowser
CloseBrowser()
```

Exercice 6, Scriptweb06.src

```
=====
' Scriptweb06
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 6
'
' Ce script capture des données affichées dans un tableau sur une page Web.
=====

Dim tabcell_2$(100)
Dim tabcell_1$(100)
Dim tabcell_0$(100)

StartBrowser("IE", "www.wintask.fr/demos", 3)

UsePage("Pages Démonstration WinTask")
    ClickHTMLElement("A[INNERTEXT= 'Tableau de données']")

UsePage("Tableau de données à capturer")
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C1:R6C1", tabcell_0$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!A1:A6", tabcell_0$())
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C2:R6C2", tabcell_1$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!B1:B6", tabcell_1$())
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C3:R6C3", tabcell_2$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!C1:C6", tabcell_2$())

CloseBrowser()

=====
' Le même script mais en utilisant des noms de tableaux reprenant les intitulés de colonnes
=====

Dim telephone$(100)
Dim email$(100)
Dim nom$(100)

StartBrowser("IE", "www.wintask.fr/demos", 3)

UsePage("Pages Démonstration WinTask")
    ClickHTMLElement("A[INNERTEXT= 'Tableau de données']")

UsePage("Tableau de données à capturer")
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C1:R6C1", nom$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!A1:A6", nom$())
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C2:R6C2", email$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!B1:B6", email$())
ret = CaptureTableHTML("TABLE[CONTENT='Nom']", "R1C3:R6C3", telephone$())
ret = WriteExcel("c:\program files\wintask\scripts\data.xls", "Feuil1!C1:C6", telephone$())

CloseBrowser()
```

Exercice 7, Scriptweb07.src

```
=====
' Scriptweb07
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 7
'
' Ce script saisit des données constantes dans un formulaire Web.
=====
```

```
StartBrowser("IE", "www.wintask.fr/demos" , 3)
'Avec Firefox
'StartBrowser("FF", "www.wintask.fr/demos" , 3)
```

```
UsePage("Pages Démonstration WinTask")
    ClickHTMLElement("A[INNERTEXT= 'Formulaire']")
```

```
UsePage("Formulaire")
    WriteHTML("INPUT TEXT[NAME= 'nom']", "DUPONT")
    WriteHTML("INPUT TEXT[NAME= 'email']", "dupont@wintask.fr")
    WriteHTML("INPUT TEXT[NAME= 'tel']", "820123456")
    ClickHTMLElement("INPUT RESET[VALUE= 'Effacer']")
```

```
CloseBrowser()
```

Exercice 8, Scriptweb08a.src

```
=====
' Scriptweb08a
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 8, partie 1
'
' Ce script lit des données dans un fichier Excel et
' les saisit dans un formulaire Web en utilisant une boucle.
=====

Dim nom$(10)
Dim email$(10)
Dim telephone$(10)

nomfichier$="c:\program files\wintask\scripts\data.xls"

ReadExcel(nomfichier$,"A2:A6",nom$())
ReadExcel(nomfichier$,"B2:B6",email$())
ReadExcel(nomfichier$,"C2:C6",telephone$())

StartBrowser("IE", "www.wintask.fr/demos" , 3)
'Avec Firefox
'StartBrowser("FF", "www.wintask.fr/demos" , 3)

UsePage("Pages Démonstration WinTask")
    ClickHTMLElement("A[INNERTEXT= 'Formulaire']")

i=0
While i < 5
UsePage("Formulaire")
    WriteHTML("INPUT TEXT[NAME= 'nom']", nom$(i))
    WriteHTML("INPUT TEXT[NAME= 'email']", email$(i))
    WriteHTML("INPUT TEXT[NAME= 'tel']", telephone$(i))
    ClickHTMLElement("INPUT RESET[VALUE= 'Effacer']")
i= i + 1
Wend

CloseBrowser()

=====
' Scriptweb08b
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 8, partie 2
'
' Ce script lit des données dans un fichier Excel et
' les saisit dans un formulaire Web en utilisant une boucle tant qu'il y a des données Excel.
=====

' Pour pouvoir traiter un tableau Excel avec un maximum de 100 lignes
' les tailles des tableaux sont augmentés par rapport au script de la partie 1
```

```
Dim nom$(100)
Dim email$(100)
Dim telephone$(100)
```

```
nomfichier$="c:\program files\wintask\scripts\data.xls"
```

```
' Lecture de 100 lignes
ReadExcel(nomfichier$, "A2:A100", nom$())
ReadExcel(nomfichier$, "B2:B100", email$())
ReadExcel(nomfichier$, "C2:C100", telephone$())
```

```
StartBrowser("IE", "www.wintask.fr/demos")
'Avec Firefox
'StartBrowser("FF", "www.wintask.fr/demos", 3)
```

```
UsePage("Pages Démonstration WinTask")
ClickHTML_Element("A[INNERTEXT= 'Formulaire']")
```

```
i=0
```

```
' Tant que nom$(i) n'est pas vide, la boucle continue.
While nom$(i) <> ""
UsePage("Formulaire")
WriteHTML("INPUT TEXT[NAME= 'nom']", nom$(i))
WriteHTML("INPUT TEXT[NAME= 'email']", email$(i))
WriteHTML("INPUT TEXT[NAME= 'tel']", telephone$(i))
ClickHTML_Element("INPUT RESET[VALUE= 'Effacer']")
```

```
i= i + 1
Wend
```

```
CloseBrowser()
```

Exercice 9, Scriptweb09.src

```
=====
' Script09
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 9
'
' Ce script assemble les morceaux dans un projet complexe de capture de données Web.
=====

Dim searchterm$(100)
Dim mfrname$(100)
Dim mfritem$(100)
Dim dciitem$(100)
Dim description$(100)

=====

function capture_une_page()
'Reinitialisation des tableaux pour avoir des tableaux vides à chaque itération.
searchterm$=""
mfrname$=""
mfritem$=""
description$=""
dciitem$=""

UsePage("eCatalog - Cross Reference Search")
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C1:R21C1", searchterm$())
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C2:R21C2", mfritem$())
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C3:R21C3", mfrname$())
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C4:R21C4", dciitem$())
CaptureTableHTML("TABLE[CONTENT='Search Term']", "R2C5:R21C5", description$())
EndFunction

Function ecrit_resultat()
'Les données sont écrites 20 par 20 car les tableaux capturés ont 20 lignes.
'Il est possible d'ajouter un paramètre à la fonction pour écrire n lignes et non 20.
'L'instruction str$ transforme l'entier numero_ligne en chaîne qui est concaténée à l'aide du signe +
WriteExcel(fichierexcel$, "A"+str$(numero_ligne)+":A"+str$(numero_ligne+20), searchterm$())
WriteExcel(fichierexcel$, "B"+str$(numero_ligne)+":B"+str$(numero_ligne+20), mfritem$())
WriteExcel(fichierexcel$, "C"+str$(numero_ligne)+":C"+str$(numero_ligne+20), mfrname$())
WriteExcel(fichierexcel$, "D"+str$(numero_ligne)+":D"+str$(numero_ligne+20), dciitem$())
WriteExcel(fichierexcel$, "E"+str$(numero_ligne)+":E"+str$(numero_ligne+20),description$())

numero_ligne = numero_ligne+ 20
EndFunction

=====
' Programme principal
=====
'Initialisation des variables.
numero_ligne=2
fichierexcel$="c:\program files\wintask\scripts\reference.xls"

StartBrowser("IE", "http://www.donaldson.com/en/index.html", 3)
```

```
UsePage("Donaldson Company, Inc. - Americas Region - English")
```

```
ClickHTMLElement("A[INNERTEXT= 'Part Search/Cross Re']")
```

```
UsePage("eCatalog - Cross Reference Search")
```

```
WriteHTML("INPUT TEXT[NAME= 'part_0']", "145*")
```

```
ClickHTMLElement("INPUT SUBMIT[VALUE= 'Search']")
```

```
exit=0
```

```
repeat
```

```
capture_une_page()
```

```
ecrit_resultat()
```

```
If existhtmlElement("INPUT SUBMIT[VALUE= 'Next page']") = 1 then
```

```
ClickHTMLElement("INPUT SUBMIT[VALUE= 'Next page']")
```

```
Else
```

```
exit=1
```

```
Endif
```

```
until exit = 1
```

Exercice 10, Scriptweb10.src

```
'=====
' Scriptweb10
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script illustre la différence entre msgbox et msgframe pour faire afficher une variable.
'=====
```

```
StartBrowser("IE", "www.wintask.fr/demos/page-iteration1.htm", 3)
```

```
i = 2
```

```
While i <= 7
```

```
    UsePage("Page Itération 1")
    MsgBox("l'index i vaut : "+str$(i))
    ClickHTMLElement("A[INNERTEXT= '"+str$(i)+"']")
```

```
    UsePage("Page Itération "+str$(i))
    ClickHTMLElement("A[INNERTEXT= 'ici']")
```

```
    i = i + 1
```

```
Wend
```

```
CloseWindow("IEXPLORE.EXE|IEFrame|Page Itération 1 - Windows Internet Explorer",1)
```

```
'=====
' Le même script mais avec msgframe.
'=====
```

```
StartBrowser("IE", "www.wintask.fr/demos/page-iteration1.htm", 3)
```

```
i = 2
```

```
While i <= 7
```

```
    UsePage("Page Itération 1")
    MsgBox("l'index i vaut : "+str$(i),1)
    ClickHTMLElement("A[INNERTEXT= '"+str$(i)+"']")
```

```
    UsePage("Page Itération "+str$(i))
    ClickHTMLElement("A[INNERTEXT= 'ici']")
```

```
    i = i + 1
```

```
Wend
```

```
CloseWindow("IEXPLORE.EXE|IEFrame|Page Itération 1 - Windows Internet Explorer",1)
```

Exercice 11, Scriptweb11a.src

```
=====
' Scriptweb11a
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script illustre la génération d'un fichier rapport de type limité où
' seules les lignes Comment sont écrites dans le fichier rapport lors de l'exécution.
=====

StartBrowser("IE", "www.wintask.fr/demos/page-iteration1.htm", 3)

i = 2

While i <= 7
    UsePage("Page Itération 1")
    Comment("l'index i vaut : "+str$(i))
    ClickHTMLElement("A[INNERTEXT= '"+str$(i)+"']")

    UsePage("Page Itération "+str$(i))
    ClickHTMLElement("A[INNERTEXT= 'ici']")

    i = i + 1
Wend

CloseWindow("IEXPLORE.EXE|IEFrame|Page Itération 1 - Windows Internet Explorer",1)
```

```
=====
' Scriptweb11b
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script génère un fichier log en appelant une Sub définie par le programmeur.
=====

sub log(msg_log$)
local buffer$
buffer$=Date$()+", "+hour$()+": "+min$()+": "+sec$()+ " --> "+msg_log$
write(fichier_log$,buffer$,CRLF)
endsub

fichier_log$="C:\test\script17blog.txt"

StartBrowser("IE", "www.wintask.fr/demos/page-iteration1.htm", 3)

i = 2

While i <= 7
    UsePage("Page Itération 1")
    log("l'index i vaut : "+str$(i))
    ClickHTMLElement("A[INNERTEXT= '"+str$(i)+"']")

    UsePage("Page Itération "+str$(i))
    ClickHTMLElement("A[INNERTEXT= 'ici']")

    i = i + 1
```

Wend

CloseWindow("IEXPLORE.EXE|IEFrame|Page Itération 1 - Windows Internet Explorer",1)

Exercice 12, Scriptweb18.src

```
=====
' Scriptweb12
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script illustre l'utilisation du mot-clé CONTENT et du mot-clé INDEX
' pour identifier de manière unique un objet HTML via son descripteur HTML.
=====
```

```
Dim datejour$(100)
Dim textenews$(100)
```

```
StartBrowser("IE", "www.wintask.fr/demos/page-news.htm", 3)
```

```
UsePage("Page News")
```

```
ret = CaptureTableHTML("TABLE[CONTENT="+year$()+"]", "R1C1:R4C1", datejour$())
```

```
ret = CaptureTableHTML("TABLE[CONTENT="+year$()+"]", "R1C2:R4C2", textenews$())
```

```
msgbox(datejour$(0))
```

```
msgbox(textenews$(0))
```

GLOSSAIRE

Barre d'outils WinTask flottante : Barre d'outils affichée pendant le mode Enregistrement. Ses icônes permettent d'accéder aux principaux outils de WinTask sans sortir du mode Enregistrement (l'Espion, les assistants de synchronisation, l'assistant de capture et le bouton Stop).

Boîte de dialogue : Fenêtre affichée par un programme où l'utilisateur saisit des informations utilisées dans la suite du programme. Les boîtes de dialogue peuvent également afficher des données.

Chaîne de caractères : Texte alphanumérique mis entre double-quotes. Par exemple "Voici le message numéro 1 pour WinTask". Une variable de type chaîne de caractères se termine toujours par le caractère \$. Par exemple montexte\$ = "Voici le message numéro 1 pour WinTask".

Compilateur : Programme WinTask qui lit un fichier script source et le transforme en fichier exécutable. Les scripts WinTask peuvent être compilés en sélectionnant le menu **Démarrer/Compilation seule** (ou touche Ctrl+F7), en sélectionnant le menu **Démarrer/Exécution** ou en cliquant sur l'icône **Exéc** de la barre d'outils. Dans les deux derniers cas, si aucune erreur de compilation n'est détectée, l'exécution est lancée une fois la compilation terminée. Notez que le lancement de la compilation enregistre d'abord le fichier script source.

Descripteur HTML : Identifiant unique d'un objet HTML se trouvant sur une page Web. Ce descripteur HTML est généré automatiquement en mode Enregistrement.

Editeur : Programme WinTask permettant la réalisation des scripts d'automatisation. De la fenêtre de l'Editeur sont accessibles toutes les fonctions nécessaires à l'écriture de scripts : mode Enregistrement, Compilation, Exécution, Assistants de synchronisation, Assistant de capture, Insertion d'instructions.

Espion : Outil WinTask permettant de trouver le nom Windows d'un objet (d'une fenêtre, d'un bouton, ...).

Log ou fichier rapport : Fichier contenant les instructions exécutées lors du rejoue. Un fichier rapport sert à comprendre pourquoi le rejoue n'a pas donné les résultats escomptés.

Macro : Voir Script.

Mode Enregistrement : Fonctionnalité de l'Editeur WinTask permettant de transformer les actions effectuées par l'utilisateur en instructions WinTask et ainsi créer le script d'automatisation.

Nom de fenêtre: Spécifie de manière unique le nom d'une fenêtre présente sur le bureau Windows. Ce nom de fenêtre est généré automatiquement par le mode Enregistrement ou peut être trouvé en utilisant l'outil Espion. Au rejoue, TaskExec utilise le nom de fenêtre pour envoyer les actions suivantes à la fenêtre spécifiée.

Runtime : Logiciel WinTask permettant seulement d'exécuter des scripts compilés réalisés avec le logiciel WinTask complet, appelé WinTask-pack de développement. Le runtime WinTask est un sous-ensemble du pack de développement, il ne contient pas l'Editeur, le Compilateur ni le Planificateur de tâches.

.ROB : Fichiers issu de la compilation d'un script source. TaskExec lit les instructions contenues dans un .ROB pour exécuter les différentes lignes et ainsi rejouer les actions enregistrées.

Script : Ensemble d'instructions du langage WinTask organisées en fichier de type texte et décrivant les différentes actions à automatiser. Les fichiers Script de WinTask portent l'extension .SRC.

.SRC : Fichiers de type texte contenant les instructions du langage WinTask décrivant les différentes actions à automatiser. L'Editeur WinTask permet de créer et de modifier les fichiers .SRC. Le Compilateur WinTask lit les fichiers .SRC et les transforme en fichiers .ROB, fichiers exécutable par le programme TaskExec de WinTask.

TaskExec : Programme WinTask qui exécute toutes les instructions auparavant compilées dans un fichier d'extension .ROB. Un double-clic sur un .ROB lance l'exécution de ce .ROB.

Index

#ActionTimeout.....	28	Instructions du langage.....	22
#IgnoreErrors.....	28	Itération (Boucles).....	31
#UseExact.....	29	Kill().....	33
Attente action.....	16	Menu contextuel dans une page Web..	90
Augmenter la vitesse du rejoue.....	92	Menus Internet Explorer.....	89
Autres instructions pour les formulaires	48	Menus Web Over.....	89
.....	48	Navigate.....	79
Capture ne renvoie pas les données		Nombres réels.....	25
désirées.....	92	Opérateurs.....	25
Chaînes de caractères.....	25	Read().....	34
ClickHTMLElement ne clique pas.....	78	ReadExcel.....	34
Compilateur		Repeat/Until.....	31
TaskComp.....	6	Saisie trop rapide dans un formulaire..	79
Délai d'attente dépassé lors du		Stopper l'exécution d'un script.....	12
chargement d'une page.....	78	Sub/ExitSub/EndSub.....	51
Editeur		Synchronisation.....	7
TaskEdit.....	6	Synchronisation – Date/Heure.....	16
Editeur boîtes de dialogue.....	7	Synchronisation - Durée.....	16
Enregistrement Bas Niveau.....	90	Synchronisation - Fenêtre.....	15
Entiers.....	24	Synchronisation - Image.....	15
Erreur élément HTML non trouvé.....	80	Synchronisation - Texte.....	15
Erreurs de compilation.....	69	Synchronisation – Texte OCR.....	15
Erreurs d'exécution.....	69	Table, capture erronée.....	81
Espion.....	7	Tableaux.....	25
Exécution		Trace en utilisant MsgBox ou MsgFrame	69
TaskExec.....	7	69
Exécution en mode Débogage.....	71	Variables.....	24
Exist().....	33	Variables directives.....	24
Fichier Rapport.....	70	While/Wend.....	31
Function/ExitFunction/EndFunction.....	51	WriteExcel.....	34
Include.....	69		