Manejo de objetos

Anteriormente, he intentado mostrar las diferentes estructuras de control que nos permiten encaminar nuestros vbs, es por esto que ahora, pasaremos a los OBJETOS.

Toda interacción de vbs con el sistema se basa en el manejo de objectos (ActiveX), que no son más que librerías especializadas y a nuestra diposición, por lo que dependiendo de lo que busquemos hacer, haremos uso de una u otra (mezc landolas en nuestros scripts)

Nota: No haré uso de dim para declarar variables en los ejemplos, para así ahorrar espacio.

En el caso de copiar y pegar uno de los ejemplos al notepad, revisar las comillas, ya que al copiar desde aquí a allí, no son las mismas, y puede que de errores.

Los objetos básicos de los que se hace uso son dos:

1. FSO (FileSystemObject): manejo de unidades, archivos y carpetas.
2. Shell: acceso a información de sistema, trabajo con el registro, manejo accesos directos, ejecución de aplicaciones.

Antes de continuar debo de explicar que al trabajar con los objetos, los mismos se deben de declarar de la siguiente manera:

Set variable = createobject(objecto)

Ejemplo:

Set objfso = createobject(“scripting.filssystemobject”)

Ahora sí, pasamos directamente al análisis del objeto FSO.

FSO (File System Object)

Las distintas funciones que se pueden realizar a través de este objeto son:

- Borrar, mover y copiar archivos
- Leer y escribir en archivos de texto (Crear)
- Obtener y modificar atributos de archivos y carpetas
- Crear, borrar, mover y copiar carpetas
- Obtener propiedades de archivos y carpetas
- Listar subcarpetas
- Listar discos y particiones
- Obtener propiedades de discos y particiones
- Obtener determinadas rutas de sistema

Mover archivos

Objfso.movefile origen, destino

Ejemplo:

Set objfso = createobject("scripting.filesystemobject")

Objfso.**movefile** "C:\archivo.txt", "D:\Carpeta"

Borrar archivos

Objfso.deletefile archivo

Ejemplo:

Set objfso = createobject("scripting.filesystemobject")

Objfso.**deletefile** "C:\archivo.txt"

Copiar archivos

Objfso.copyfile origen, destino, sobreescribir

Ejemplo:

Set objfso = createobject("scripting.filesystemobject")

Objfso.**copyfile** "C:\archivo.txt", "D:\destino.txt", true

Crear carpetas

Set variable = objfso.createfolder(destino carpeta)

Ejemplo:

Set objfso = createobject("scripting.filesystemobject")

Set micarpeta = objfso.**createfolder**("C:\carpeta")

Mover carpetas

Objfso.movefolder origen, destino

Ejemplo:

Set objfso = createobject("scripting.filesystemobject")

Objfso.**movefolder** "C:\Carpeta", "D:\Destino"

Borrar carpetas

Objfso.deletefolder carpeta

Ejemplo:

Objfso.**deletefolder** "C:\Carpeta"

Copiar carpetas

Objfso.copyfolder origen, destino, sobreescribir

Ejemplo**:**

Set objfso = createobject("scripting.filesystemobject")

Objfso.**copyfolder** "C:\Carpeta", "D:\Destino", true

Leer y escribir en archivos

Para el manejo de archivos de texto, debemos de tener en cuenta, la existencia o no del archivo, y el modo en el que accedemos a él.

Obviamente, si un archivo no existe, no podrémos acceder a él, y a su vez, si abrimos un archivo en modo de léctura, nunca podremos escribir dentro.

Crear archivos de texto y escribir en ellos

Set variable = objfso.createtextfile(ruta, sobreescribir)

Ejemplo:

```
Set objfso = createobject("scripting.filesystemobject")
Set archivotexto = objfso.createtextfile("C:\archivo.txt",true)        'creamos el archivo
archivotexto.writeline "Este es el texto que estoy escribiendo"        'escribimos una línea
archivotexto.writeblanklines(2)                                        'escribimos 2 líneas en blanco
archivotexto.writeline "Aquí más texto"                                ' escribimos otra línea de texto
archivotexto.close                                                     'cerramos el archivo
```

Notese que al comenzar, hemos creado el archivo y lo hemos asignado a una variable, luego hemos utilizado el identificador de archivo (variable), para escribir dentro de él, en este caso, hemos utilizado **writeline**, que escribe una línea, y agrega un retorno de carro para que si volvemos a escribir, lo hagamos en una nueva línea, en cambio , si en su lugar, utilizamos **write,** el resultado, será que no habrá salto de línea, por lo que todas las oraciones iran quedando una detrás de la otra. Por último, hemos cerrado el archivo.

Abrir archivos de texto y escribir en ellos

Set variable = objfso.opentextfile(ruta, modo, creación)

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set archivotexto = objfso.opentextfile("C:\archivo.txt",8,true)        'abrimos el archivo
archivotexto.writeline "Este es el texto que estoy escribiendo"        'escribimos una línea
archivotexto.close                                                     'cerramos el archivo
```

Como se puede ver, al abrir el archivo, hemos indicado la ruta, el modo 8 que se utiliza para appending o escritura al final de archivo, y true, que quiere decir que en caso de que no exista el archivo se cree, es decir que de este modo, no solo abrimos el archivo, sino que de no existir, dicho archivo será creado en el proceso.

Para tener en cuenta, los modos en los que se puede abrir un archivo son:

1- Modo LECTURA

2- Modo ESCRITURA (escribe al principio)

8- Modo APPENDING (escribe al final)

Como se puede ver, al abrir el archivo, hemos indicado la ruta, el modo 8 que se utiliza para appending o escritura al final de archivo, y true, que quiere decir que en caso de que no exista el archivo se cree, es decir que de este modo, no solo abrimos el archivo, sino que de no existir, dicho archivo será creado en el proceso.

Leer desde archivos de texto

Así como abrimos archivos y podemos escribir en ellos, también existe la posibilidad de leer desde ellos, para lo cual utilizaremos **readline**, y **readall.**

Como se puede imaginar, con **readline** iremos leyendo una a una las líneas del archivo (cada vez que pongamos readline leeremos solo una), con este, leemos una línea, y el puntero se situa al final de la línea, para que a la próxima ejecución de esta función, sea la línea siguiente la que sea leída. Con **readall** en cambio, leeremos el total de archivo.

Existe además una función llamada **skipline,** con la cual saltaremos la lectura de una línea.

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set archivotexto = objfso.opentextfile("C:\archivo.txt",1)        'abrimos el archivo
msgbox archivotexto.readline                                      'leemos una linea, la primera
archivotexto.skipline                                             'saltamos una línea
msgbox archivotexto.readline                                      'leemos una línea, la tercera
archivotexto.close                                                'cerramos el archivo
```

Atributos de archivos y carpetas

A continuación explicaré el método mediante el cual, se puede obtener, o bien cambiar, los atribuos de archivos y carpetas (es prácticamente igual para ambas cosas)

Obtener atributos

Set variable = objfso.getfile(ruta)

variable.attributes

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set archivo = objfso.getfile("C:\tutorial.pdf")        'obtenemos el control sobre el archivo pdf
Msgbox archivo.attributes                              'mensaje con los attributos del archivo
```

En este ejemplo vemos que luego de declarar el objeto, lo que hacemos es obtener el control del el archivo **tutorial.pdf,** para lo cual utilizamos **getfile**, y asignamos el archivo a la variable **archivo**. Luego, y ya con el archivo en la variable, podemos utilizarlo directamente para mostrar sus atributos, que no será más que un número que englobará todas las constantes de los atributos.

Las constantes que hacen referencia a los atributos de archivo son:

Valor	Atributo
0	Normal
1	Solo Lectura
2	Oculto
4	Sistema
8	Letra de disco
16	Carpeta/directorio
32	Archivo
64	Link o acceso directo
128	Comprimido

Como comentaba antes, **attributes** devolverá un valor único que será la sumatoria de cada uno de los valores para cada atributo del archivo.

Como ejemplo:

Un archivo que tenga atributos de; solo lectura, oculto, de sistema, y de archivo, tendrá un valor de 1+2+4+32= **39**

Cambiar atributos

Set variable = objfso.getfile(ruta)

variable.attributes = sumaatributos

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set archivo = objfso.getfile("C:\tutorial.pdf")          'obtenemos el control sobre el archivo pdf
archivo.attributes = 34                                  'attributo dearchivo y oculto
```

El cambiar atributos consiste simplemente en asignar un valor a attributes.

En el ejemplo anterior he puesto como valor **34** que representa atributos de archivo y oculto.

Propiedades de archivos y carpetas

Así como podemos obtener y modificar los atributos de los archivos y carpetas, podemos acceder a determinadas propiedades de los mismos, entre ellas:

- Nombre; name
- Nombre corto; shortname
- Tamaño; Size
- Ruta completa; path
- Ruta corta; shortpath
- Fecha de creación/modificación/último acceso; datecreated, datelastmodified, datelastaccessed
- Tipo de archivo; type
- Carpeta contenedora; parentfolder

Estas son las principales propiedades a las que podemos acceder, lo cual se hará de la siguiente forma:

Set variable = objfso.getfile(ruta)

Msgbox **variable.propiedad**

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set archivo = objfso.getfile("C:\tutorial.pdf")          'obtenemos el control sobre el archivo pdf
Msgbox archivo.size                                      'tamaño del archivo en bytes
```

Para obtener otras propiedades, simplemente sería cuestión de cambiar "**size"** por alguna de las otras propiedades a las que tenemos acceso (les recomiendo probar con cada una para ver los resultados), y de igual manera, se haría con carpetas en lugar de archivos, para lo que solamente tendríamos que cambiar **getfile,** por **getfolder**.

Listar subcarpetas

Con listar subcarpetas, nos referimos a, acceder a la **colección** de subcarpetas de una carpeta "X", para trabajar con cada una de ellas de manera independiente.

Cuando hablamos de colección, hacemos referencia a un array, en el que están todos los elementos contenidos en un determinado "listado".

Set variable = objfso.getfolder(ruta)

Set subvariable = variable.subfolders

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set micarpeta = objfso.getfolder("C:\Carpeta")        'obtenemos el control sobre la carpeta
Set subcarpetas = micarpeta.subfolders                'obtenemos la colección de subcarpetas
    For each s in subcarpetas                         'por cada carpeta(s) en la colección(subcarpetas)
    Msgbox s.name                                     'mensaje con el nombre
Next                                                  'pasamos a la siguiente subcarpeta
```

Bueno, como se ve en este caso, el acceder a las subcarpetas no es totalmente directo, sino que, en un principio, obtenemos la colección (array) de carpetas, para luego trabajar con cada una de ellas por medio de un **for** que recorre toda la colección.

Al listar subcarpetas, se debe de terner en cuenta además, que la colección, solamente contiene, las carpetas del primer nivel, y no las que se encuentran en los niveles consiguientes, para que quede más claro:

C:\carpeta

C:\carpeta\nivel1

C:\carpeta\nivel1\nivel2

Si listamos la colección de subcarpetas en la carpeta "**C:\carpeta**", obtendremos todas las del ***nivel 1***, pero no las del nivel 2, para eso, deberíamos de implementar otro **for** que haga referencia a estas.

Listar discos/particiones

Al igual que para listar subcarpetas, para listar discos y particiones lo haremos a traves de una colección, por lo que en realidad, será muy similar al punto anterior.

Set variable = objfso.getfolder(ruta)

Set discos = variable.drives

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set discos = objfso.drives                    'obtenemos la colección de discos
    For each d in discos                      'por cada disco(d) en la colección(discos)
    Msgbox d.driveletter                      'mensaje con la letra de disco
Next                                          'pasamos al siguiente disco
```

Como ya se había dicho, el método de listar los discos y particiones, es el mismo que para obtener las subcarpetas de una carpeta, por lo que no debería de presentar ningún problema el hacerlo.

También aclarar, que en el anterior ejemplo, solo he incluído una propiedad que no se había visto anteriormente, y esta es la de **driveletter**, y que como ya se habrán dado cuenta, hace referencia a la letra que tiene asignado el disco o partición en el sitema, por lo que una vez mencionado esto, pasaremos justamente, a identificar, cuales son las propiedades de disco a las que tendremos acceso.

Propiedades de disco

Como hemos visto en el anterior punto, a traves del objecto FSO es posible acceder a la colección de discos, así como a sus propiedades, siendo estas propiedades, las siguientes:

- Letra; driveletter
- Nombre del disco; volumename
- Espacio disponible; availablespace
- Espacio libre; freespace
- Espacio total; totalsize
- Disponibilidad; isready
- Ruta; path
- Sistema de ficheros (NTFS, FAT, CDFS); filesystem
- Carpeta principal; rootfolder
- Número de serie; serialnumber
- Nombre compartido; sharename
- Tipo de disco; drivetype

Ejemplo, no pondré en este caso, ya que es sería igual al que he puesto anteriormente.

Por otra parte, solamente profundizaré en una de las propiedades de los discos (creo que las otras son facilmente identificables), y es justamente, la última que he enumerado, **drivetype**

El tipo de disco de un disco (valga la redundancia), puede estar entre los siguientes:

Valor	Tipo
0	Unknown/Desconocido
1	Removable/Removible
2	Fixed/Rígido
3	Network/Red
4	CD-Rom
5	RAM Disk

También he de acotar, que la disquetera (Unidad A), es reconocida como disco extraíble, así como también ocurre, con las unidades virtuales, ej; aquellas carpetas montadas con el comando SUBST de ms-dos.

Rutas de carpetas

Por último, pero no menos importante, debo de comentar que el objecto FSO, permite obtener la ruta de tres de las carpetas más importantes del sistema, como lo son:

- Windows (0)
- System32 (1)
- Temp (2)

Para acceder a ellas, haremos uso de la función **getspecialfolder**.

Set variable = objfso.getspecialfolder(constante)

Ejemplo

```
Set objfso = createobject("scripting.filesystemobject")
Set micarpeta = objfso.getspecialfolder(0)          'obtenemos el control sobre la carpeta
Msgbox micarpeta.path                               'mensaje con la ruta de la carpeta
```

Como se ve en ejemplo, en este caso, nos hacemos con el control de la carpeta de windows, y luego mostramos su ruta, para poder accesar a las carpetas, las constantes son las que he indicado en un entre parentesis en un principio, al nombrar las carpetas

Bueno, con esto hemos terminado con el objeto FSO, pasemos entonces al segundo y principal objeto, la shell.

Shell (wscript.shell)

A través de este objeto podemos entre otras cosas:

- Mostrar mensajes temporizados
- Leer, borrar, y escribir en el registro de windows
- Ejecutar aplicaciones (dos métodos)
- Obtener el foco de una ventana
- Enviar pulsaciones de teclado
- Acceder multiples carpetas de sistema
- Obtener variables del sistema
- Crear accesos directos

Mostrar mensajes temporizados

Esta es la menos importante de todas las funciones de este objeto, pero no deja de ser útil en algunas ocasiones, sobre todo, si no queremos que un proceso se bloquee, solo por intentar mostrar un mensaje. Para que quede más claro, a lo largo de todos los ejemplo que he puesto, he ido mostrando determinados mensajes a traves de la función de **msgbox** (opción básica)**.** Los mensajes generados con dicha función, no salen de pantalla, hasta bien el usuario, no da click en el/los botones que tiene el cuadro de mensaje, por lo que en caso de utilizar un mensaje de este tipo, la ejecución de nuestro script, se parará hasta bien este no salga de pantalla.

Pero en lugar de utilizar este tipo de mensaje, podemos hacer uso de una ventana **popup**, la cual desaparecerá sola al cabo del tiempo que hayamos especificado, luego de lo cual, continuará la ejecución del script.

Modo en el que se emplea esta ventana:

Variableventana = variable.popup(texto, tsegundos, título, botones)

Ejemplo:

```
Set objshell = createobject("Wscript.shell")

rventana = objshell.popup("Este es un mensaje de prueba",3,"Mensaje Popup",64)
```

El resultado;

Para empezar, se puede apreciar que el objeto shell, se declara con **Wscript.shell,** y bueno, lo referente al mensaje, creo que una imagen dice más que mil palabras, hemos mostrado un mensaje en pantalla, el cual al cabo de 3 segundos, desaparecera y dará paso al resto de nuestro script.

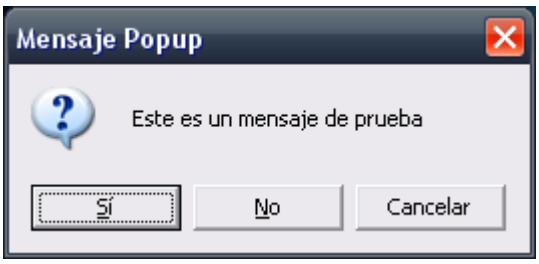
Notese algo, y es que para los "botones" he puesto el valor **64**, este valor, hace referencia a la sumatoria de los botones e iconos que aparecen en la ventana de mensaje, siendo los mismos:

Botones	
Valor	**Descripción**
0	OK
1	OK y Cancelar
2	Abortar, Reintentar e Ignorar
3	Si, No y Cancelar
4	Si y No
5	Reintentar y Cancelar

Iconos	
Valor	**Descripción**
16	Parar/Error
32	Pregunta
48	Exclamación
64	Información

En el caso de mi mensaje, el 64 corresponde a la suma del botón de ok (cero) y el icono de información (64)

Si quisieramos mostrar un mensaje de *Si, No y Cancelar*, con un icono de pregunta, deberíamos de poner como valor el número **35** (3+32).



Por otro lado, también vemos en el ejemplo como el popup, se guarda en una variable de nombre **rventana**. En dicha variable, se guarda el valor de la respuesta dada al mensaje, es decir, para el caso de un mensaje con los botones de Si, No y Cancelar, la variable **rventana** guardaría el valor de cual de esos botones fue presionado por el usuario, siendo estas las posibilidades:

Valor	Descripción
1	OK
2	Cancelar
3	Abortar
4	Reintentar
5	Ignorar
6	Si
7	No

Es decir, que en caso de que la persona presione **Si**, el resultado obtenido será **6**, esto puede servirnos de mucho si queremos darle un poco de interactividad a nuestros scripts y tomar diferentes caminos en virtud de las respuestas de los usuarios, lo cual puede ser utilizado junto a las funciones de **msgbox** (para esta también valen los botones e

iconos) e **inputbox,** aunque a pesar de la mención las veremos en detalle, ya que no dependen de ningún objeto, y son de fácil comprensión.

Leer, borrar y modificar el registro de windows

Sin lugar a dudas, es una de las funciones más atractivas de este objeto, y su implementación no implica ninguna dificultad, y aún así, la implementación de esta función, no presenta ninguna dificultad.

Leer valores

variable.regread(llave de registro)

Ejemplo:

```
Set objshell = createobject("wscript.shell")

Msgbox objshell.regread("HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\My Pictures")
                                                    'va todo en una misma línea, la llave es muy larga
```

Sencillo de entender, a través del objeto shell llamamos a la función **regread**, encargada de leer las llaves del registro, en este caso, he optado por una llave un tanto larga, pero que contiene la ruta de la carpeta de **"mis imágenes",** al igual que en otros ejemplos, muestro el valor por medio de un **msgbox**, pero como en todos los casos, es posible guardar este valor en una variable para luego trabajar con ella.

Borrar valores

variable.regdelete(llave de registro)

Ejemplo:

```
Set objshell = createobject("wscript.shell")

objshell.regdelete("HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\My Pictures")
```

Al igual que para leer, el borrar cadenas no tiene nada de especial, sino que simplemente debemos de llamar a la función **regdelete** para deshacernos de la cadena en cuestión.

Escribir/modificar el registro

Algo que de seguro le interesará a varios.

variable.regwrite(llave de registro, valor, tipo de dato)

Ejemplo:

```
Set objshell = createobject("wscript.shell")

objshell.regwrite("HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr",1,"REG_D
WORD")                                              'va todo en una misma línea, la llave es muy larga
```

En este ejemplo, estaríamos modificando la llave del registro que deshabilita el **administrador de tareas** (taskmgr).

Los datos que debemos de pasarle a esta función son; la llave a modificar/crear claro esta, el valor que le daremos, en mi ejemplo ha sido **1**, y el tipo de datos que estamos ingresando.

Como muchos sabrán, el registro de windows soporta ciertos tipos de datos en las diferentes claves de las que hace uso, a saber:

Tipo	Descripción
REG_SZ	Cadena
REG_DWORD	Numérico
REG_BINARY	Binario
REG_EXPAND_SZ	Una cadena expandible

Existe un quinto tipo de datos, el **REG_MULTI_SZ**, pero no es soportado por esta función.

Tener en cuenta que **regwrite** puede tener problemas con cadenas demaciado largas, y considerar también, que tal como he puesto en el título, regwrite creará una llave en caso de no existir, y la modificará en caso de que ya esté presente.

También he de recomendar que en caso de trabajar con el registro se ha de tener mucho cuidado, y que quererse dejar el code más legible, se utilicen las siguientes abreviaciones (lo he hecho en mi ejemplo):

Llave principal	Abreviación
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_CLASSES_ROOT	HKCR
HKEY_USERS	HKEY_USERS
HKEY_CURRENT_CONFIG	HKEY_CURRENT_CONFIG

Ejecutar aplicaciones

Como he comentado antes, a traves de este objeto existen dos métodos diferentes de ejecutar aplicaciones, los cuales además, tendrán diferentes opciones y resultados.

Método **Run**

Este es el más simple y usado de los dos métodos.

variable.run rutaprograma, estadoventana, espera

Ejemplo:

```
Set objshell = createobject("wscript.shell")

Objshell.run "notepad", 1, true
```

En este caso ejecutamos el bloc de notas, en modo normal, y establecemos, que el script se pause (true) hasta que se cierre la aplicación (notepad).

Los estados en los que se puede abrir la ventana, van del 0 al 10, pasando por oculto, maximizado y minimizado, entre otros, pero pasaré de explicar cada uno de ellos ya que resulta mejor opción probar y ver el resultado, que lo que podría ser la explicación.

Lo que si cabe destacar, es el estado 0 (cero) o vbhide, que ejecuta una aplicación en modo **oculto**, sin mostrar ningún tipo de ventana. Probar por ejemplo sustituyendo el número **1** por vbhide en el ejemplo que he dejado, para

ver (en realidad no se verá nada) como se ejecuta el notepad sin mostrar ninguna ventana (si se verá el proceso en el administrador de tareas).

Tener en cuenta además, que el estado de la ventana, y la espera del programa, son totalmente opcionales , así que con poner simplemente objshell.run “notepad” hubiese alcanzado.

Método **Exec**

Este método es muy similar al anterior, solamente que nos permite un mayor control sobre la aplicación que estamos corriendo, con la excepción de que no nos permite seleccionar estado de la ventana.

variable.exec(rutaprograma)

Ejemplo:

```
Set objshell = createobject(“wscript.shell”)

Objshell.Exec(“notepad”)
```

Como vemos, el método en el que se ejecuta la aplicación es practicamente igual al anterior, solamente hemos sutituído **run** por **exec** y no hemos pasado ningún parámetro a la función, pero ahora veamos que es lo que o hace diferente del otro método.

- Status
- ProcessID
- Terminate
- Stdin, Stdout, Stderr

Estas serían las funciones que diferencian el **exec** del **run,** todas estas se aplican sobre la aplicación/comando que estamos ejcutando, por lo que, para hacer uso de esta funcionalidad, es necesario ejecutar la aplicación, y asignar la misma a una variable para luego trabajar a traves de esta.

Veamos esto, junto a la explicación de **status** y **processid**

El **status** identifica el estado de la aplicación ejecutada, 0 (cero) representa una aplicación en ejecución, y 1 (uno), indentifica que la aplicación fue cerrada y su proceso ya no se encuentra presente.

Con respecto al **processid,** creo que salta a la vista la función del mismo, no es más que el identificador de proceso (PID).

Ejemplo:

```
Set objshell = createobject(“wscript.shell”)

Set bloc = Objshell.Exec(“notepad”)

Msgbox bloc.status

Msgbox bloc.processid
```

En este ejemplo, ejecutamos el bloc de notas, asignando su proceso a la variable **“bloc”**, y luego mostramos dos mensajes, uno con el status (debería de ser 0), y otro con el número de proceso.

La función status nos servirá en definitiva, para poder monitorear nuestro proceso, y realizar una determinada acción en función de si permanece abierto, o bien, lo han cerrado.

Ejemplo:

```
Set objshell = createobject("wscript.shell")

Set bloc = objshell.Exec("notepad")

Do while bloc.status = 0

        Wscript.sleep 200                    'sirve para hacer una pausa de x milésimas de segundo

loop

msgbox "Se ha cerrado el bloc de notas"
```

Pasemos entonces a **terminate**

Esta función, permite terminar el proceso que habíamos iniciado, por lo que por su nombre era fácil predecirlo

Ejemplo:

```
Set objshell = createobject("wscript.shell")

Set bloc = objshell.Exec("notepad")

Wscript.sleep 5000                           'Pausa de 5000 milésimas, o 5 segundos

bloc.terminate
```

Abrimos el bloc de notas, realizamos una espera de 5 segundos, y cerramos el bloc de notas, fácil.

Pasando ahora a las últimas funciones de exec; stdin, stdout, stderr, debo de aclarar que estas funciones pueden ser utiles a la hora de trabajar en línea de comandos, pero dado que en línea de comandos hay mejores maneras (a mi criterio) de hacerlo, obviaré la explicación de dos de estas funciones, y solo le daré importancia a **stdout** que si puede facilitarnos la tarea en la ejecución de comandos de ms-dos.

Stdout nos muestra la salida de un los comandos ejecutados a traves de **exec**, y para que se entienda mejor, pasare directamente a un ejemplo;

```
Set objshell = createobject("wscript.shell")

Set ping = objshell.exec("cmd /c ping www.google.com")

Msgbox ping.stdout.readall
```

El resultado:

En este ejemplo, hemos ejecutado un ping a google a traves del cmd, y luego por medio de **stdout** y su **readall** hemos leído el contenido completo de la salida del comando y lo hemos mostrado en un msgbox.

También, en lugar de **readall** podemos utilizar **read(x)**, donde x es el número de caracteres que queremos leer.

En un ejemplo un poco mas "elaborado", haremos lo mismo que antes, pero esta vez haciendo uso de **read**, leeyendo el texto completo, pero carácter a carácter.

```
Set objshell = createobject("wscript.shell")

Set ping = objshell.exec("cmd /c ping www.google.com")

do

    if not ping.stdout.atendofstream then

        respuesta = resultado & ping.stdout.read(1)

    else

        exit do

    end if

loop

msgbox respuesta
```

Este ejemplo, es un poco más dificil de explicar, así como también lo es entender, por lo que quien quiera entenderlo deberá prestar atención y visualizarlo detenidamente.

En resumen, ejecutamos el ping a google, y luego nos metemos en un bucle infinito (do), dentro de este, un análisis condicional, si **no** se alcanza el final de la salida (stdout.**atendofstream**), la variable respuesta, es igual a la propia variable y un carácter (concatenamos). Ese analisis se continuará haciendo hasta que se alcance el final de la lectura, es ahí donde entonces, pasamos al **else**, y el **else** dice que, se sale del bucle do (**exit do**).

Por lo que en definitiva, el if se ejecutará una y otra vez, e ira guardando en la variable (respuesta) letra por letra hasta formar el mensaje completo, cuando se llegue al final, se saldrá del bucle, y se mostrara el msgbox con la "respuesta". Un poco más complicado, pero espero se entienda la idea de este ejemplo.

Obtener el foco de una ventana

Variable.appactivate tituloventana

Ejemplo.

Set objshell = createobject("wscript.shell")

Objshell.appactivate "Sin título"

Con esto, obtendríamos el foco de una ventana con por ejemplo, el título "Sin título – Bloc de notas", que como se habrán dado cuenta, es el título que tiene un nuevo bloc de notas.

Las consideraciones que debemos de tener en cuenta al utilizar esta función, son las referentes al orden en el que realiza las comparaciones en busca de nuestra ventana;

- Primero busca la ventana que tiene el título exacto que hemos establecido
- Si no encuentra el título exacto, busca aquella ventana que coincida en el **principio** del título
- Si no encuentra ni el título exacto, ni tampoco encuentra título que comience como el nuestro, entonces busca aquel que **termina** como el que queremos.
- También, en el caso de haber varias ventanas con el mismo nombre, elegira una cualquiera al azar, y le dará el foco.

Enviar pulsaciones del teclado

Otra de las funciones llamativas de este objeto, es la función **sendkeys** que nos permitirá enviar pulsaciones de teclado, como si estuviesemos escribiendo, así que hay que tener cuidado con el uso que se da, ya que una vez ejecutada, enviará las pulsaciones a la ventana activa, por lo que si no es lo que deseabamos, podemos terminar presionando un montón de teclas en un programa x, con los resultados que ello puede acarrear.

Variable.sendkeys teclas

Ejemplo:

Set objshell = createobject("wscript.shell")

Objshell.sendkeys "Estas son las teclas"

Este es un ejemplo muy simple ,que además no recomiendo probar, ya que como he dicho antes, enviará las pulsaciones de las teclas a la ventana activa, y los resultados pueden no “gustarnos”.

Es por eso que a continuación muestro un ejemplo más claro de esta función, haciendo uso además, de funciones que hemos visto anteriormente.

```
Set objshell = createobject(“wscript.shell”)

Set bloc = objshell.exec(“notepad”)                    ‘ejecutamos el bloc de notas

Wscript.sleep 2000                                     ‘espera de dos segundos

Objshell.appactivate bloc.processid                    ‘ponemos el foco en la ventana del bloc

Wscript.sleep 200                                      ‘espera de mílesimas

Objshell.sendkeys “Tutorial vbs”                       ‘enviamos un mensaje con sendkeys

Objshell.sendkeys “{ENTER}”                            ‘luego del mensaje anterior, un ENTER

Wscript.sleep 2000                                     ‘nueva espera de dos segundos

Objshell.sendkeys “Probando la funcion sendkeys”       ‘enviamos una segunda línea de mensaje
```

Como había dicho, en este ejemplo he hecho uso de algunas otras fuciones vistas anteriormente.

Para empezar he optado por ejecutar el bloc de notas a traves de la función **exec**, podría optarse por **run,** pero este ejemplo me servía para mostrarles el uso que puede hacerse del **processid**. Como vemos, a la hora de obtener el foco de la ventana con **appactivate,** he puesto como “título” el **processid**, ya que vbs nos permite hacer uso conjunto de estas dos funciones, para de este modo asegurarnos de que el foco se ponga en la ventana que corresonde y no en otra con similar nombre. Al igual que antes, podríamos poner simplemente el nombre de la ventana del bloc de notas en lugar de processid, pero igualmente creo no quedarán dudas de por que el uso de esta.

Luego, con respecto al envío de las teclas, no creo que merezca explicación alguna, ya que es simplemente el mensaje/pulsación que queremos envíar.

Igualmente habrán visto en medio del envío de teclas que también esta presente una tecla especial, el {ENTER}, es por eso que dejo a continuación, la tabla con el listado de argumentos para estas teclas:

Tecla	Argumento
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INSERT} or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
Fx	{Fx}

Para el caso de las teclas SHIFT, CTRL y ALT, los argumentos con **+**, **^** y **%** respectivamente, no siendo posible utilizar, la tecla PRTSC, o lo que es igual, la tecla de PRINT (la que se utiliza para capturar pantalla). Para utilizar estas tres letras;

```
Objshell.sendkeys "+A"

Objshell.sendkeys "^V"

Objshell.sendkeys "%{TAB}"
```

Carpetas "especiales"

Con carpetas especiales me refiero a aquellas carpetas como "Inicio", "Enviar a" y "Favoritos" entre otras. Para eso, haremos uso de la función **specialfolders**

Variable.specialfolders(carpeta)

Ejemplo:

```
Set objshell = createobject("wscript.shell")

Msgbox objshell.specialfolders("Desktop")
```

Con lo que obtendríamos un msgbox, con la ruta del escritorio, algo como "C:\Documents and Settings\Novlucker\Escritorio", de igual modo y como he mencionado antes, podemos acceder a las rutas de otras carpetas, para lo que, contamos con la siguiente tabla.

Carpeta	Identificador
Escritorio	AllUsersDesktop
Menú Inicio	AllUsersStartMenu
Programas	AllUsersPrograms
Inicio	AllUsersStartup
Escritorio	Desktop
Favoritos	Favorites
Fuentes	Fonts
Mis documentos	MyDocuments
Entorno de red	NetHood
Impresoras	PrintHood
Programas	Programs
Reciente	Recent
Enviar a	SendTo
Menú Inicio	StartMenu
Inicio	Startup
Plantillas	Templates

Con esta tabla, solo bastaría cambiar "desktop" en nuestro ejemplo, para ver las rutas de otras de las carpetas disponibles, y nótese también que si bien, algunas carpetas parecen estar repetidas, algunos identificadores hacen referencia a las carpetas de **todos** los usuarios, y otras a la del usuario activo.

Obtener variables de sistema

Bueno, habrán visto entonces, que a pesar de la cantidad de carpetas de las que se puede obtener la ruta, hay algunas muy interesantes que no aparecen con el **specialfolders,** como ser "Archivos de programa", la carpeta de usuario, etc. , es por eso que entonces, haremos uso de la función **expandenvironmentstrings.**

Este función, y como se ha adelantado en el título, permite acceder a todas las variables del sistema, como ser, USERNAME, USERPROFILE, COMPUTERNAME, y todas aquellas que aparecen a través del comando SET de ms-dos.

Variable.expandenvironmentstrings(%variablesistema%)

Ejemplo:

```
Set objshell = createobject("wscript.shell")

Msgbox objshell.expandenvironmentstrings("%Programfiles%")
```

Como es de esperarse, en este caso, nos aparecerá un mensaje con la ruta de la carpeta de "Archivos de programa".

Crear accesos directos

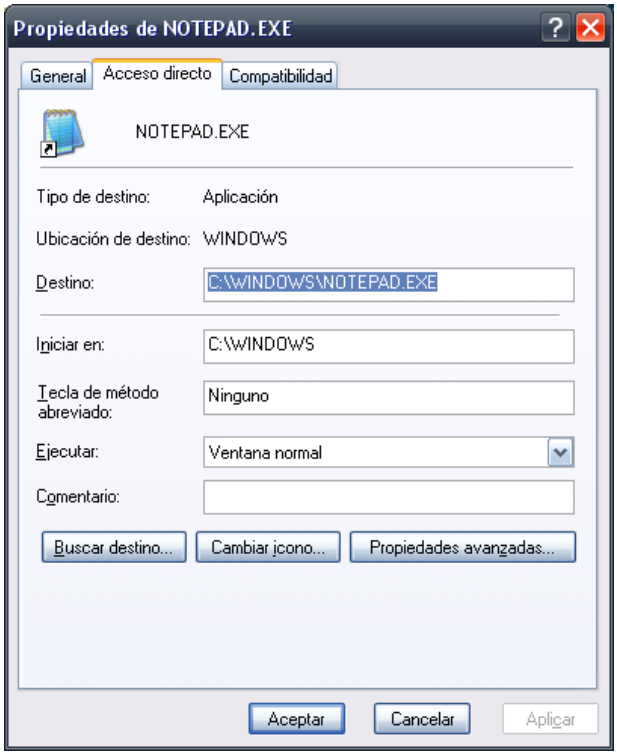
Bueno, luego de haber visto ya, varias funciones, solo queda por ver, la de **createshortcut**, la cual permite crear accesos directos a cualquier aplicación/archivo, quizás parezca una función un poco "insulsa", pero quizás alguien quiera por ejemplo, agregar un acceso directo en la carpeta de **Inicio**, y será aquí cuando recurramos a esta función.

En esta función es necesario definir varios valores, así que en lugar de explicarla, pasaré directamente a un ejemplo:

```
Set objshell = createobject("wscript.shell")

Set ellink = objshell.createshortcut("C:\Acceso directo.lnk")        'creamos el link

Ellink.targetpath = "C:\windows\notepad.exe"                        'completamos los valores

Ellink.windowstyle = 1

Ellink.hotkey = "CTRL+SHIFT+N"

Ellink.iconlocation = "C:\windows\notepad.exe,0"

Ellink.description = "Acceso directo a notepad"

Ellink.workingdirectory ="C:\"

Ellink.save                                                         'guardamos el link
```

Creo que este ejemplo alcanza para entender sin problemas esta función, no hacemos más que crear el link, completamos todos los datos que lleva el link, y salvamos.

Los campos a completar son los que aparecen en cualquier link (imagen adjunta), teniendo en cuenta demás que no todos son indispensables.



Falta agregar, que pueden crearse links a páginas web's del siguiente modo.

Ejemplo:

Set objshell = createobject("wscript.shell")

Set weburl = objshell.createshortcut("C:\google.url")

Weburl.targetpath = http://www.google.com

Weburl.save

Simplemente hemos cambiado la extensión del link a **url**, en lugar de **lnk** como tenía antes, y la ruta del link pasa a ser la dirección de la web a la que queremos acceder.

Una vez más ,así termina la segunda parte de este tutorial, en donde he intentado mostrar a quienes no tienen conocimientos, los dos principales objetos de la "librería" de vbs.