Integrar FileMaker Pro y AppleScript

FileMaker Pro ha sido una de las primeras aplicaciones en soportar Apple Events. Está provista de un completo conjunto de eventos y objetos que se pueden controlar para dar funcionalidad añadida a las bases de datos de FileMaker de forma externa a la aplicación. Aquí se exponen algunas ideas para el uso en conjunto de FileMaker y Apple event, con algunos trucos sobre su sintaxis y la programación de los eventos y objetos más comúnmente utilizados. Igualmente se incluye el diccionario de eventos y objetos, que se puede consultar asímismo en los archivos que instala FileMaker por defecto en una instalación normal. En este caso se ha creido necesaria su inclusión porque toda referencia que hace FileMaker a AppleScript sigue estando en inglés, incluyendo la actual versión FileMaker 7. Finalmente se incluyen ejemplos de forma más extensiva para poder ver el proceso de comunicación entre aplicaciones que se está exponiendo de forma muy clara y concisa.

Aunque este libro tiene la estructura expuesta anteriormente, no se ha pretendido que el lector la siga tal cual está. Por el contrario, el autor del mismo, al tratarse de una persona muy autodidacta, hubiera preferido desarrollar el texto únicamente con ejemplos, lo que nos llevaría a una monotonía en algunos casos, y a una falta de uso de eventos y/o objetos que pudieran ser de interés para el lector actual. Por ello, y en dependencia de cómo sea el lector, podrá empezar por la primera parte y luego ir a los ejemplos recurriendo al diccionario de eventos y objetos a modo que vaya teniendo dudas acerca del uso de cada uno de estos, o se podrá seguir tal cual como se ha expuesto inicialmente. En la mayoría de los casos, se incluyen ejemplos de desarrollo, estos, salvo en la última parte del libro, tienen el único sentido de poder seguir el tema expuesto de una forma más visual y para mejorar su comprensión.

Por último, añadir que, aunque el libro está basado fundamentalmente en el uso de FileMaker y AppleScript, la idea aprendida será fácilmente extrapolable a otras aplicaciones.

Herramientas a utilizar

Se necesita tener instalado el sistema operativo MacOS 7.5 como mínimo para poder utilizar los Apple Events. Asimismo, será conveniente tener instalada la última versión de FileMaker Pro, la cual, a fecha de publicación de este documento, es FileMaker 7. Muchos de los eventos "Apple-event" solicitados por los usuarios de FileMaker han sido incorporados desde versiones anteriores por lo que se recomienda revisar el diccionario de la versión de FileMaker que tenga instalada.

Herramientas de Sript. Utilidades como la de UserLand Frontier, y AppleScript 1.X, ofrecen el entorno necesario para crear scripts auto-ejecutables que pueden comunicarse con otros programas, scripts y archivos.

[Nota: En la versión de sistema operativo MacOS 9, se deberán tener instalados igualmente los 5 módulos de seguridad “Security Cert Module”, “Security Library”, “Security Manager”, “Security Policy Module”, y “Security Storage Module” dentro de la carpeta de extensiones ya que si no es así, se nos generarán errores ajenos a AppleScript evitando que nuestro código se ejecute como teníamos previsto. Igualmente, indicar que, si se ha iniciado el”Classic” desde cualquier versión de MacOS 10, no funcionarán los Scripts generados para este entorno, por lo que, si va a utili-

zar guiones de FileMaker que contengan código AppleScript, se recomienda generar dos guiones distintos donde se incluya el mismo código, pero con la salvedad de “compilarlos” en su entorno]

Referencias. Todo programador de Apple Event debe tener al menos una sólida guía de referencia como "The Tao of AppleScript", "The Complete AppleScript Handbook" de Danny Goodman, "AppleScript in a Nutshell" de Bruce W. Perry, o el "AppleScript Developer's Kit" de Apple Computer que se puede descargar de forma gratuita de la web de Apple. También se podrá usar el Editor de Scripts para acceder al diccionario de eventos y objetos de FileMaker Pro. Asimismo, se puede consultar la base de datos de eventos y objetos de FileMaker en la carpeta "FileMaker y AppleEvents" que se instala de forma genérica con FileMaker Pro al realizar una instalación completa y que se incluye una traducción en este libro. Como se ha comentado anteriormente, a lo largo de este documento se incluirán scripts preparados con el "Editor de Scripts" por su carácter gratuito, que se incluye en sistemas superiores al 7.5, con lo que se podrán copiar dentro de la ventana de texto de este para no tener que teclearlos de nuevo. Asimismo, se ha utilizado el entorno Classic para una mayor compatibilidad. Esto se podrá ejecutar igualmente en entorno X sin que sea necesario su adaptación, ya que el mismo código valdrá para uno y otro entorno. Si está utilizando MacOS X se podrá beneficiar también del uso del AppleScript Studio, que se puede instalar con las herramientas de desarrollo incluidas con este sistema operativo.

[Nota: FileMaker, a partir de la versión 2.1v2 incluye una base de datos de eventos y objetos FileMaker que ofrece documentación más amplia y ejemplos de sintaxis de los eventos soportados.]

Soporte. Foros "online" son los medios más comunes de intercambiar noticias, consultas prácticas, scripts, y preguntas. Se puede conectar a los foros de Frontier y AppleScript que se pueden consultar a través de internet. Asimismo, Apple España realiza soporte técnico de AppleScript. Para realizar consultas acerca de las distintas versiones de AppleScript se puede hacer a través del teléfono 902 151 827

Entender SciptMaker®

ScriptMaker® es una de las herramientas más fáciles y potentes que se pueden encontrar entre las distintas aplicaciones de desarrollo para sistemas MacOS. No es difícil llegar a crear guiones con ScriptMaker® ya que, a medida que se van creando estos, ScriptMaker® se encarga de autocompletarlos y de depurarlos para evitar que haya errores durante su ejecución. A todo ello, le añadimos que ScriptMaker® también puede ejecutar AppleScript (así como Apple Events), por lo que podremos llegar a realizar las tareas más comunes no sólo desde dentro de nestra base de datos, sino con otros programas que necesitemos ejecutar para cumplimentar nuestra tarea, por ejemplo, crear un diagrama en base a los datos contenidos en unos registros, escanear un documento para incluirlo en nuestra base de datos, preparar y enviar un fax, etc...

Para poder incluir y ejecutar un AppleScript desde dentro de FileMaker tendremos que:

Seleccionar el menu Guion
Seleccionar ScriptMaker®
Indicar un nombre y crear un nuevo guión
(Borrar los pasos que se crean por defecto) - Opcional y dependiente de la versión
Incluir el paso de guión "Ejecutar AppleScript" (Al final de la lista de pasos disponibles)

Hacer Click sobre el botón "Especificar" bajo el cuadro donde se visualizan los pasos de guión.

Nuevos conceptos de lo antiguo

Programar significa pensar en la interfase de Macintosh como el medio para comunicar a un sujeto una serie de frases verbo-sustantivo. Así, tendremos que:

El sujeto es el programa al cual le queremos transmitir (tell) nuestro mensaje, es decir, las acciones que queremos que realice.

Cuando se hace doble-click, arrastra, o selecciona un comando ("verbo") desde un menú, se dice a una aplicación, ya sea el Finder o FileMaker Pro, que haga algo con ese objeto. Comandos como, Cerrar, Cortar, Copiar, Pegar, y Borrar son "events".

Piense por un momento acerca de todos los tipos de objetos ("sustantivo") que se seleccionan: archivos, carpetas, palabras, párrafos, menús, aplicaciones. En la terminología de eventos Apple, estos "seleccionables" se denominan "objetos". Programar es sencillamente una forma de decir a las aplicaciones cómo actuar sobre lo que está seleccionado, por ejemplo, cuando queremos automatizar el doble click sobre una carpeta deberemos decir 'Abrir carpeta'.

Nuestro primer ejemplo, en vez de ser el tan conocido "Hello World", será comunicarle a nuestro Finder que abra una carpeta nueva (que llamamos "carpeta sin título" por ser de reciente creación) por nosotros:

```
tell application "Finder" --Indicamos a nuestro Mac que queremos decirle algo al Finder
        open folder "carpeta sin título" --El susodicho mensaje
end tell -- Terminamos nuestra comunicación abierta anteriormente
```

Siempre se dice algo al sujeto

A medida que vayamos avanzando, se observará que existen una serie de eventos que sólo lo entienden las aplicaciones para las cuales están diseñados y otros que son genéricos. Por ejemplo, se puede decir '**open**' tanto a FileMaker como al Finder. Por esta razón siempre se deberá indicar a quién se quiere enviar el evento, como se ha visto en el ejemplo anterior.

Por otra parte, los eventos que son propios de una aplicación, no será necesario "encerrarlos" en una estructura del tipo '**tell application**' '**end tell**' siempre que el AppleScript se ejecute desde dentro de esta. Por ejemplo, si queremos ir a un registro determinado, dentro de FileMaker le podremos indicar únicamente '**go to record 1**', pero si esto mismo se lo queremos indicar a FileMaker desde fuera de la aplicación, entonces nos tendremos que dirigir a la aplicación con la estructura anterior:

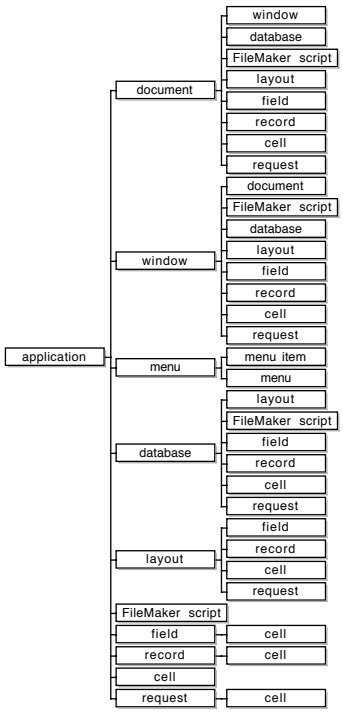
```
tell application "FileMaker Pro"
        go to record 1
end tell
```

Nota: Hay ciertos eventos que igualmente son entendidos sin estar incluidos dentro de la estructura '**tell application**' ' **end tell**'. Por ello, es recomendable que se incluya para evitar enviar un evento a la aplicación equicovada.

Entender los Objetos

Las aplicaciones están diseñadas para manejar algunos objetos mejor que otros. El Finder, por ejemplo, lo hace con archivos, carpetas, y aplicaciones. Documentos, párrafos, frases, palabras, y caracteres son el terreno preferido de los procesadores de texto. En el caso de FileMaker, se tiene un conjunto de objetos que es familiar a los usuarios de bases de datos: campos, registros, presentaciones, y bases de datos. La jerarquía contenida en cada aplicación describe las relaciones entre objetos mostrando qué objetos están contenidos por otros. Aunque no se ha indicado anteriormente, pero igualmente importante, son los menús, opciones de menú, y guiones de FileMaker.

Los objetos se clasifican en un orden jerárquico llamado "*object containment hierarchy*." Los objetos más grandes contienen otros más pequeños. La jerarquía de contenido define qué objetos están contenidos por otros objetos. En los procesadores de texto, un documento contiene párrafos; los párrafos contienen frases; las frases contienen palabras, las cuales contienen letras. Es como una muñeca Matriuska rusa (que literalmente significa "pequeña madre"). Abres la muñeca grande, y contiene una más pequeña, que contiene otra más pequeña, y así. Al principio de la jerarquía de contenido de objetos de FileMaker está la aplicación FileMaker en sí. Dentro de la aplicación, se tienen documentos, ventanas, y menús. Las ventanas contienen bases de datos, que contienen presentaciones, que, a su vez, contienen registros y campos.



Jerarquía de contenido en FileMaker

Todos los objetos tienen propiedades o características que pertenecen a su posición en su "ley del más fuerte" de la aplicación. La mayoría de objetos, como presentaciones, ventanas, bases de datos, guiones, y menús tienen una denominación de propiedad. Los objetos “campos” pueden tener propiedades adicionales que describen, por ejemplo, el número de repeticiones definidas para ese campo ("*campo repetitivo*"), o el cálculo introducido en un campo de cálculo ("*Cálculo*"). La mayoría de las propiedades son de tipo "*sólo lectura*" (*r/o*), esto significa que se pueden obtener las propiedades que están definidas para este objeto pero no se pueden cambiar. Las propiedades de un objeto lo hacen apropiado únicamente para ciertas tareas. Esto se puede ver en la llamada a ventanas, documentos, y bases de datos del siguiente ejemplo.

TRUCO: Cuando se hace una referencia a objetos por sus nombres, hay que estar seguro de cerrarlo entre comillas ("") al principio y al final del nombre del objeto.

```
tell application “FileMaker Pro”
    tell window “Pruebas”
        tell layout “Lista”
            tell field “Campo A”
```

A continuación se enumeran los objetos FileMaker que se pueden controlar a través de AppleScript. Para obtener más información acerca de objetos y sus propiedades, vea el diccionario de FileMaker Pro.

Application (Aplicación)
El objeto *application* permite especificar qué programa se desea controlar, que en este caso es FileMaker Pro. Una vez que se especifique la aplicación, se tiene acceso a todos los objetos y comandos únicos que la pertenecen. Casi todos los Scripts que se escriban comenzarán con la llamada a la aplicación que se desea controlar.

```
tell application "FileMaker Pro"
        activate
end tell
```

Window, *document*, y *database* (Ventana, documento y base de datos)
Estos objetos se refieren a los archivos FileMaker. Sus propiedades, de cualquier forma, los hacen útiles en diferentes circunstancias. Por ejemplo, mientras que el objeto "*database*" se refiere a un conjunto completo de registros de una base de datos, el objeto "*document*" pertenece sólo al conjunto de registros encontrado. El objeto "*window*" tiene propiedades únicas, como el tamaño, que permite cambiar las propiedades de una ventana en la pantalla.

Layout (Presentación)
El objeto *layout* se puede usar para moverse entre presentaciones, generar informes, ordenar registros, o referirse a campos incluidos en una presentación en concreto. Este objeto es de sólo lectura "*r/o*" y se puede acceder por nombre o número. Las presentaciones se enumeran de

acuerdo al orden definido en la opción del menú "Opciones de Presentación".

TRUCO: A los más introducidos el FileMaker les sonará haber escuchado algo acerca de la “Presentación 0”, Pues bien, aunque esta existe, sólo la podremos acceder a través de AppleScript. Esta presentación contiene todos los campos definidos en nuestro archivo, así será fácil poder recurrir a una presentación donde sabemos, de seguro, que está el campo que necesitamos sin temer a un error de AppleScript que nos indique que no se encuentra el objeto.

Record (Registro)
Los registros son uno de los pocos objetos a los cuales ha de referirse sólo por número. Use este objeto cuando esté interesado en un registro en particular o un grupo de registros. Este podría ser el primer registro de una búsqueda, el quinto de la base de datos, o el registro que se muestre en pantalla.

Field / Cell (Campo/Celda)
Use el objeto *field* cuando esté interesado en valores introducidos en más de un registro. El objeto field se usa también para buscar las opciones definidas para un campo, como el número de repeticiones, opciones de verificación, y cálculos. El objeto *cell*, por el otro lado, es más como una hoja de cálculo. Se utiliza para referirse al valor específico contenido en un campo de un registro en particular.

Menu / Menu Item
Los menús en FileMaker son accesibles por nombre (Apple, Archivo, Edición, Seleccionar, Presentación, Organizar, Formato, Guión, y Ventana) o por número (1-9). Los elementos, o items, de menú como Nuevo Registro, Refind, y Alinear son también accesibles por nombre o número. Algunas de las opciones de menú, incluyendo los Privilegios de Acceso, Borrar Registro, Borrar Todos, y los menús de ortografía no son accesibles a través de los eventos Apple. En los casos donde se produce un error, por ejemplo, donde existe un submenu con objetos no “scriptables”, se producirá el error: FileMaker Pro got an error: Object not found.

Guiones FileMaker
Este objeto se refiere a los guiones individuales que se definen ScriptMaker® y son accesibles por nombre o por número.

Comprensión de los eventos

Los eventos son los comandos de la aplicación que se pueden programar con Scripts. Si los objetos son los nombres de la frase creada en un script, y los eventos son los verbos, tiene sentido que necesitemos dar al evento un objeto contra el cual actuar. En términos de Apple event, el objeto de un evento es llamado parámetro. Los eventos pueden tener también parámetros opcionales que son como las opciones o preferencias. Cuando se construyen pasos de script, se coloca primero el evento y luego el objeto o parámetro ya que decimos qué queremos hacer (evento) con un objeto determinado.

A continuación se enumeran algunos de los eventos más útiles así como algunos ejemplos y trucos. El conjunto de todos los eventos se puede consultar más adelante, en el diccionario de FileMaker incluido en este libro.

Tell
El evento *Tell* se usa por AppleScript para especificar el objeto que se quiere controlar. Todos los

pasos incluidos tras *Tell* se dirigirán hacia ese objeto hasta que se incluya un paso con *End Tell*. *Tell* se utiliza más comúnmente con el objeto application. La primera vez que se use un paso con *Tell Application*, AppleScript puede preguntar dónde está la aplicación indicada utilizando un diálogo standard de apertura de ficheros. Una vez que se ha guardado la ruta de esta aplicación en memoria, el diálogo no volverá a aparecer. A continuación se incluye un guión en AppleScript utilizando Tell:

```
tell application "FileMaker Pro"
        show layout 1 -- Vamos a la primera presentación de nuestra base de datos
        create new record -- Generamos un nuevo registro
end tell
```

Este sencillo guión dice a FileMaker Pro que se vaya a la primera presentación de la base de datos actual y añada un nuevo registro. Note que si no se especifica ningún objeto tipo base de datos, documento, o presentación, FileMaker usa el objeto actual como objeto por defecto. Sin embargo, si no se especifica el registro, el primer registro del documento, no el actual, es el que se toma por defecto. Para enviar ordenes al registro actual se deberá utilizar la propiedad de bases de datos "*current record*". El siguiente guión de AppleScript muestra cómo hacer esto copiando el resultado de la propiedad "*current record*" a una variable definida por el usuario, que llamamos "*MiRegistro*" y luego pega la fecha en un campo del registro actual.

```
tell application "FileMaker Pro"
        set MiRegistro to current record of database 1 -- Obtenemos el ID del registro actual
        set cell "Fecha Revisión" of MiRegistro to "30/12/94" -- Ponemos la fecha 30/12/1994 en el registro anterior
end tell
```

A lo largo de este documento se verán dos tipos de ejemplos: Pasos de guión de AppleScript y guiones completos. Se podrán identificar fácilmente los guiones completos porque tienen al menos tres líneas de longitud, comienzan con "Tell Application" y finalizan con "End Tell." Los pasos de guión de AppleScript suelen ser de una sola línea y no se podrán compilar por sí solos. Para ejecutar un paso de guión se necesitará insertar este dentro del paso de guión de FileMaker "*Ejecutar AppleScript*" o colocarlo entre un "*Tell*" y "*End Tell*" como se muestra a continuación.

```
Tell Application "FileMaker Pro"
   -- (insertar los guiones aquí)
End Tell
```

Asimismo se deberá tener una base de datos abierta cuando se utilicen los ejemplos.

Open

Este evento se utiliza para abrir objetos a nivel del Finder, como bases de datos de FileMaker. Para utilizar el evento *Open*, se deberá especificar una ruta hacia el archivo en cuestión. Comience con el nombre del volumen (el disco duro o partición del Mac). Añada el nombre de cada carpeta a abrir hasta llegar al documento. Finalmente, añadir el nombre del archivo. Todo esto tiene que estar separado por dos puntos (:), el archivo, carpeta, y volumen, y tiene que estar encerrado entre paréntesis. A continuación se incluye un guión para abrir la base de datos llamada "*Compact Discs*" :

```
tell application "FileMaker Pro"
        open file "Macintosh HD:Carpeta FileMaker Pro:Lista CD.fp5" -- abrir una base de datos
        open file "Macintosh HD:Carpeta FileMaker Pro:Batters" with password "admin" -- abrir una base de datos con
la clave "admin"
        open {"Macintosh HD:Carpeta FileMaker Pro:Lista CD.fp5", "Macintosh HD:Carpeta FileMaker
Pro:Músicos.fp5"} ¬
                with password {"admin"} -- abrir bases de datos con la misma clave
        open {"Macintosh HD:Carpeta FileMaker Pro:Batters", "Macintosh HD:Carpeta FileMaker Pro:Músicos.fp5"} ¬
                with password {"admin", "usuario"} -- arir bases de datos con clave distinta
        open file "Macintosh HD:Carpeta FileMaker Pro:Lista CD.fp5" with password "" -- abrir base de datos con la
clave en blanco
end tell
```

TRUCO: Use el símbolo de continuación "¬" (tecleando alt-Return) para poder continuar escribiendo el guión en la línea siguiente.

Count

El evento *Count* se puede utilizar para buscar el número de elementos incluidos en un objeto. Se puede usar *Count* para contar el número de registros de una base de datos, el número de registros encontrados, o el número de campos que hay en una presentación. Se aplica a los siguientes objetos: ventanas, documentos, bases de datos, guiones, registros, campos, celdas, y presentaciones.

TRUCO: para poder ver los resultados, podemos abrir el Registro de Eventos, en el editor de scripts, y activar la casilla “resultados de eventos”

```
tell application "FileMaker"
        count of record -- registros de un archivo (incluyendo la búsqueda)
        count record of database 1 -- registros de un archivo (se ignoran la búsqueda)
        count request -- las peticiones de búsqueda en un archivo
        count field -- campos en un archivo
        count FileMaker script of window 1 -- scripts de FileMaker en el archivo
        count of cell of document 1 -- celdas en el archivo (incluyendo la búsqueda)
        count cell of record 2 -- celdas en el registro
end tell
```

Este evento es más útil cuando se definen bucles o repeticiones (loops) que operan con un conjunto de objetos, como los registros de una presentación en concreto. Por ejemplo, para repetir una operación en cada registro de un conjunto, se definirá un bucle del tipo, "*repeat with i from 1 to x*", donde i es la variable que se incrementa en 1 y x es el número de registros encontrados. El siguiente guión utiliza el evento *Show* para buscar todos los registros con "*Khachaturian*" en el campo "*Compositores*". Luego un loop comienza con el primer registro encontrado y copia "*Favorito*" en el campo Preferencias, un registro tras otro, hasta que se haya alcanzado el último registro. En tanto que el objeto "*document*" se refiere sólo al conjunto de registros encontrados, se puede saber cuántos registros se han encontrado diciendo "*Count Records of Document 1*".

```
tell application "FileMaker" -- nos referimos a FileMaker
```

```
        show (every record whose cell "Compositores" = "Tchaikovsky") -- Buscamos los registros con Tchaikovsky
        repeat with i from 1 to (count record of document 1) -- comenzamos el bucle desde 1 hasta todos los encontra-
dos
                set cell "Preferencias" of record i to "Favorito" -- cambiamos el valor del campo
        end repeat -- terminamos el bucle
end tell
```

El siguiente conjunto de guiones muestra cómo contar el número total de registros de la base de datos, el número de presentaciones, número de campos, y número de guiones:

```
Count Record of database "Compact Discs"
Count Layout
Count Field
Count FileMaker Script
```

Create

Create se utiliza para añadir nuevos registros como si se seleccionara "*Nuevo Registro*" en el menú Edición. Si se usa el parámetro opcional "*with data*", se creará un registro con datos predefinidos. Aquí hay algunos ejemplos:

```
(*Para usar este guión, se tiene que activar la base de datos Músicos.fp5
*)
tell application "FileMaker"

        create new record -- registro (en los registros de la búsqueda actual y sin datos)
        create new record with data "J.S. Bach" -- registro (con datos en un campo)
        create new record with data {"Katchaturian", "Mediocre"} -- registro (con varios campos
en)
        create new record at database 1 -- registro (en el conjunto completo) en la base de datos
1

        create new request -- petición de búsqueda (sin datos). Aunque daría error, nos permite
hacer esto para poder pasarle datos, posteriormente
        set cell "Compositores" of request 1 to "Tchaikovsky" -- Le pasamos el dato que queremos
buscar
        create new request with data "Katchaturian" -- petición de búsqueda (con un campo)
        create new request with data {"Katchaturian", "Mediocre"} -- petición de búsqueda (con
varios campos en el orden de creación de los campos en esa presentación)
        create new request with data {"", "Favorito"} with properties {omitted:true} --petición de bús-
queda (con la opción de omitir)
        (* En este ejemplo se generarán las peticiones de búsqueda correctamente, pero no se
mostrarán, ni se realizará la búsqueda. Para poder comprobarlas, lo haremos a través de la
opción "Modificar Ultima Búsqueda" - Comando+R *)

        create new menu with properties "MiMenu" -- menu
        create new menu at menu "External" with properties "Menu 2" -- menu dentro de un menu
```

(o submenu)

create new menu item with properties "Elemento de Menu" -- elemento de menu

create new menu item at menu "Menu 2" of menu "External" with properties "Elemento de Menu" -- elemento de menu (de un menu)

end tell

Como hemos visto, FileMaker nos ofrece la posibilidad de generar menús, y elementos de menú que nos permitirá ejecutar opciones externas a FileMaker.

Usando la capacidad de repetición de AppleScript, se pueden crear fácilmente cientos o miles de registros con sólo unos pasos de guión. Pruebe esto:

```
Tell application "FileMaker Pro"
   Repeat 200 times
     Create New Record
   End Repeat
End Tell
```

Delete

Use el evento *Delete, por ejemplo,* para borrar registros de una base de datos de forma selectiva. Por ejemplo, el primer paso del siguiente guión borrará los diez primeros registros del conjunto actual. El segundo paso utiliza la cláusula "*whose*" para buscar y borrar en el mismo momento.

```
Delete Record 1 through 10 -- Borramos los registros del 1 al 10
Delete every Record whose Cell "Canciones" = "Love ¬ --Borramos todos los registros que cumplen
   is like Oxygen"        --una condición
Delete requests -- Borramos todas las peticiones de búsqueda
```

Do Script

Use el evento *Do Script* para lanzar un guión ScriptMaker® del menú de guiones de FileMaker Pro. Los guiones de FileMaker siempre son buenos para guardar búsquedas complejas, o recordar ajustes de página y opciones de impresión, o imprimir sin diálogos. Estas características no están soportadas con los eventos de Apple. Para ejecutar un guión de FileMaker, pruebe alguno de los siguientes ejemplos:

```
Do Script FileMaker Script 1
Do Script FileMaker Script "Imprimir Lista CD Favoritos"
Do Script last FileMaker Script
```

El evento *Do Script* es la clave de la programación condicional. Es un ejemplo de cómo el soporte de eventos Apple por FileMaker Pro nos permite realizar tareas que no se pensaran que fueran posibles. A continuación hay un ejemplo de un guión condicional que realiza diferentes guiones de FileMaker dependiendo del valor introducido en el primer registro del conjunto de registros hallados:

```
Tell application "FileMaker Pro"
  If Cell "Preferencias" = "Mediocre" then
    Do Script FileMaker Script "Archiva"
  else
    Do Script FileMaker Script "Mantén e Imprime"
  end if
end Tell
```

Show

Use el evento Show para mostrar una presentación o grupo de registros. *Show* literalmente dice a FileMaker que traiga al frente una pantalla, documento, base de datos, registro, o presentación que se especifique. Cuando se trabaje con registros en conjunción con la cláusula "*whose*", Show es más parecido a una petición de búsqueda de FileMaker. El objeto especificado en un evento *Show* debe estar abierto o disponible; por lo que sería necesario abrir una ventana, documento, o base de datos antes de mostrar un objeto de este.

Nota: realizar *Show* puede tardar más que si se realiza una búsqueda con FileMaker debido al hecho que *Show* no hace uso de los indices de FileMaker, pero puede llegar a ser más potente (sobre todo en versiones anteriores de FileMaker), ya que podremos combinar varios Show para poder ampliar o reducir nuestra búsqueda. A continuación hay unos ejemplos que utilizan *Show*:

```
Show Window 2
```

Esto traerá la segunda base de datos listada en el menú Ventana al frente

```
Show Layout "Lista Compositores"
```

Este guión muestra cómo cambiar a otra presentación especificada por nombre.

```
Show every Record whose Cell "Compositores" = "Mussorgsky"
```

Una de las aplicaciones más interesantes del evento *Show* es la realización de lo que equivale a una búsqueda en FileMaker.

TRUCO: Para definir búsquedas “personalizadas” en bases de datos cuya clave no permita editar guiones, por ejemplo, podremos crearlas dejando los criterios “a disposición del usuario” y que este utilice unos campos creados para que este usuario pueda indicar los criterios y parámetros de búsqueda.

Do Menu

Este evento ofrece el control sobre casi todos las opciones de menú de FileMaker Pro, incluyendo alguna que no son accesibles a través de los guiones de FileMaker. Por ejemplo, usando *Do Menu* se puede cambiar el estilo del texto de un campo de Times 10 puntos a Helvética 12 puntos. Para usar un evento *Do Menu* se tiene que especificar el menú y la opción del menú que nos interesa.

Al construir un guión con *Do Menu*, se comenzará diciendo "*Do Menu*" y luego se especifica la opción del menú y el menú al cual pertenece. Aunque pueda parecer extraño usar la palabra Menu dos veces consecutivas en nuestro guión, esto es necesario, porque el evento *Do Menu*

y el objeto *menú ítem* utilizan el término "*menú*". Aquí hay un ejemplo que selecciona "*Definir Partes...*" del menú de FileMaker “Presentación” (hay que estar en modo Presentación antes de ejecutar este guión):

```
Do Menu Menu Item "Definir Partes..." of Menu "Presentación"
```

NOTA: Note los puntos suspensivos en este guión. Esté seguro de crearlos como tal (pulsando alt-punto) en vez de tres puntos consecutivos cuando se utilicen nombres de selecciones de menú.

Otra gran ventaja ofrecida gracias al soporte de eventos Apple por FileMaker es la posibilidad de acceder a submenus. Las opciones de menú de FileMaker como son Fuente y estilo son ejemplos de submenus. Cuando se selecciona Estilo del menú Formato, aparece una nueva lista de opciones: Texto Normal, Negrita, Cursiva, etc. Para poder seleccionar Negrita como estilo se selecciona una opción del menú Estilo del menú Formato, como se ve en el siguiente ejemplo:

```
Do Menu Menu Item "Negrita" of Menu "Estilo" of Menu "Formato"
```

TRUCO: Cuando nos referimos a opciones de menú, recuérde que los menús son parte de la aplicación. Por tanto, hay que incluir un "*tell application*" antes de incluir un *Do Menu* para indicar que ese evento se aplica sólo al programa FileMaker.

Get Data/Set Data

Considérese estos eventos como los "caballos de batalla", porque casi todos los guiones podrán contenerlos. En muchos casos, estos eventos son como copiar y pegar. Se utilizan cuando se necesita acceder, manipular o editar datos. La flexibilidad de estos eventos viene dada por lo generales que son. Por ejemplo, para obtener la propiedad de un objeto, se puede usar tanto *Set Data* como *Get Data*. Si se necesita saber cuántas repeticiones se han definido para un campo repetitivo llamado "Títulos", se podrá usar Get Data para obtener esta información. Este caso se tendrá que construir como:

```
Get Repeat Size of Field "Títulos"
```

TRUCO: Al usar los eventos *Get Data* y *Set Data* se puede omitir la palabra "*Data*" en los guiones.

Muchas veces, es más útil guardar la información obtenida por *Get Data* en una variable y así la podemos utilizar a esta más tarde. Use el evento *Set Data* para guardar valores en una variable que se defina. A continuación se muestra un guión que muestra cómo se haría:

```
Set numerodeRepeticiones to Repeat Size of Field "Títulos"
```

En el caso anterior, el evento *Set Data* se utiliza para guardar el número de repeticiones en una variable que llamamos "numerodeRepeticiones". Ahora que se ha guardado este valor en una variable, no hay necesidad de escribir otro guión conteniendo *Get Data* la próxima vez que se necesite el número de repeticiones en ese guión. Por esta razón, muchos programadores en AppleScript utilizan *Set Data* antes que *Get Data*.

Set Data se utiliza frecuentemente también para introducir los valores de los campos de FileMaker. Para hacer esto, se selecciona un campo y se construye un “set” igual a una variable. Aquí hay algunos ejemplos:

```
Set Cell "Duración"to "52 minutos"
Set Cell "Número de Repeticiones" to numerodeRepeticiones
```

El siguiente ejemplo copia un valor en un campo llamado "Preferencias", al principio del conjunto de registros encontrados:

```
Set Field "Preferencias" to "Buena"
```

Sort

El evento *Sort* ofrece un gran control sobre cómo se ordenan los registros. Para usarlo, se puede especificar una presentación, el campo o campos por los que se quiere que se ordene, y la dirección (ascendente, descendente, numérica, o personalizada). Si no se especifica ninguna presentación, se utilizará la actual. Si no se especifica ningún campo, FileMaker desordenará la base de datos (volviendo al orden de creación de los registros). A continuación se incluyen algunos ejemplos:

TRUCO: Para poder ordenar de forma “personalizada” en bases de datos cuya clave no permita editar guiones, por ejemplo, podremos crearlas dejando los criterios “a disposición del usuario” y que este utilice unos campos creados para que este usuario pueda indicar la forma de ordenar.

```
Sort layout 1 by field "Preferencias" in order ascending
Sort layout "Lista CD" by {field "Duración", field ¬
   "Preferencias"} in order {descending, ascending}
```

NOTA: No usar comillas al indicar la dirección.

Save

Finalmente, está el evento *Save*, que se puede utilizar para indicar a FileMaker que guarde en disco todos los cambios recientes que se hayan hecho. Es una buena práctica al programar indicar a FileMaker que guarde los cambios al principio de todos los guiones que usen *Get Data* o *Set Data*. Si, durante la ejecución de un guión, el cursor se queda activo dentro de un campo, los últimos cambios introducidos no se obtendrán por *Get Data* ni *Set Data*. Sólo se podrán obtener los datos guardados. *Save* se puede utilizar en bases de datos, ventanas, o documentos con el mismo efecto. La sintaxis se muestra en los siguientes ejemplos:

```
Save Window 1
Save Window "Lista CD"
Save Database 1
Save Database "Músicos"
```

Plugin, y mis “plugout” con AppleScript

La tendencia actual de FileMaker está encaminada hacia los plugin, que son pequeños progra-

mas creados en C, con cualquiera de las herramientas existentes, para poder desarrollar distintas acciones (para ver una lista detallada de plugin disponibles, vea la web de FileMaker (www.filemaker.com). En cambio, AppleScript está un poco dejada de lado (¿con la esperanza que, quien trabaje con FileMaker, se introduzca en el mundo del desarrollo profesional?) en contra de la interfaz y herramientas que convierten a FileMaker en una base de datos muy intuitiva a la vez de fácil de usar por gente nada experta en bases de datos ni en programación.

En muchos casos, el uso de un plugin para Mac estaría justificado cuando se implantan acciones que, bien por su rapidez, bien por la falta de medios, se obtengan unos beneficios amortizables con la inversión a realizar en la adquisición de este. Recordemos que, al igual que AppleScript en un Mac, se pueden implantar acciones semejantes con VisualBasic en un PC, aunque la versión para Mac, en cuanto a este punto se refiere, es más potente. (Este sería el caso donde se justificara el uso de un plugin ya que nos evitaría tener que desarrollar dos códigos distintos si se usara mac o PC, ya que el uso de un único código implementaría las acciones que deseamos utilizar.) La mala noticia es que la mayoría de los plugin, a dia de hoy, que se hacen para FileMaker está optimizado para Mac, por no hablar de los costes que tienen algunos, sobre todo si se adaptan en un sistema multipuesto.

Muchos de los plugin que existen ofrecen acciones que se pueden crear fácilmente, y de forma más legible con AppleScript. Por esta razón, yo he pasado a denominarlos “plugout”. Por ejemplo:

```
External ("file-Append", "0|archivo.txt|Datos - " & CampoNombre)
```

sería lo mismo que

```
write (“Datos - " & CampoNombre) to file archivo.txt starting at eof
```

En ambos casos, tanto utilizando el plugin gratuito (se ha escogido un plugin gratuito en beneficio del lector) File Toolbox como utilizando AppleScript, y teniendo en cuenta que el campo CampoNombre contiene el nombre “Pepe”, se añade el texto “Datos - Pepe” al final del archivo “archivo.txt”.

En algunos casos será necesario el uso de algunas adiciones de script. Aunque algunas de ellas ya se incorporan durante la instalación de el sistema operativo, otras será necesario obtenerlas, por ejemplo, a través de internet, en la dirección http://www.osaxen.com se podrá obtener más información acerca de las adiciones de script más comunes (en tanto que las adiciones de script se pueden crear por cualquier programador, no sólo por Apple, van apareciendo cada vez más adiciones de script, por lo que, obtener una relación completa de las adiciones de script existentes será complicado. Asimismo, la mayoría de las adiciones de script se podrán encontrar gratuitas y otras con algún coste adicional. No obstante, algunas adiciones de script van introduciendo nuevas características, parámetros, etc. por lo que es recomendable comprobar el diccionario de las adiciones de script que se utilizan en este texto para comprobar que las tiene disponibles en su sistema.

Antes de empezar

Hay que tener en cuenta los siguientes aspectos para poder crear nuestros guiones manejando AppleScript:

Variables o "flags": En muchos casos, tanto manejando AppleScript, como si sólo manejamos FileMaker, necesitamos apoyarnos en unos valores que pueden ser iguales durante el uso de nuestra base de datos. Estos pueden ser únicos para cada usuario, o el mismo para todos los usuarios, así como para los registros de nuestra base de datos. Por esta razón, definiremos un campo de texto para cada variable distinta que podamos manejar si se trata de la misma para todos los usuarios, y un campo global, accesible por cada archivo que constituye nuestra aplicación, si es una variable distinta para cada usuario. En nuestro caso, este campo lo denominaremos gAS con un número, para identificarlos, así tendremos gAS1, gAS2, gAS3, … (ello nos indicará que se trata de un campo global utilizado para nuestros AppleScript) para que cada usuario pueda obtener su valor después de ejecutar el AppleScript sin interferir en los valores del resto de los usuarios.

NOTA: Es importante vaciar las variables siempre que no la necesitemos más incluyendo el paso de guión "Establecer Campo ["gAS1" , """"]"

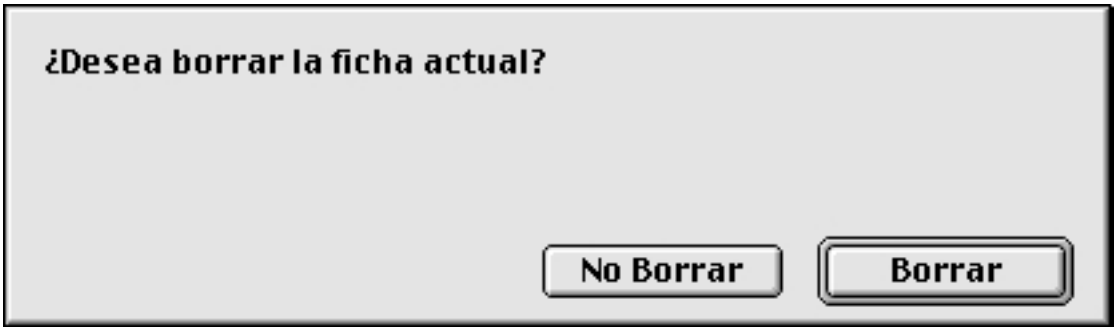
Comillas. Las comillas se utilizan mucho en los guiones creados con AppleScript, sobre todo cuando se pasan valores a una variable en forma de texto, por ejemplo, **set MiValor to "Esto es un valor de texto"**. Por ello hay que tener en cuenta los siguientes aspectos a la hora de, por ejemplo, crear un campo donde introduzcamos nuestro guión de AppleScript:

Una comilla sencilla (") la obtendremos mediante cuatro comillas (""""),
Una comilla, al iniciar o al terminar un texto, la obtendremos mediante tres comillas y el texto correspondiente ("""texto"),
Una comilla, dentro de un texto, la obtendremos mediante dos comillas ("Este es ""mi texto entre comillas""")

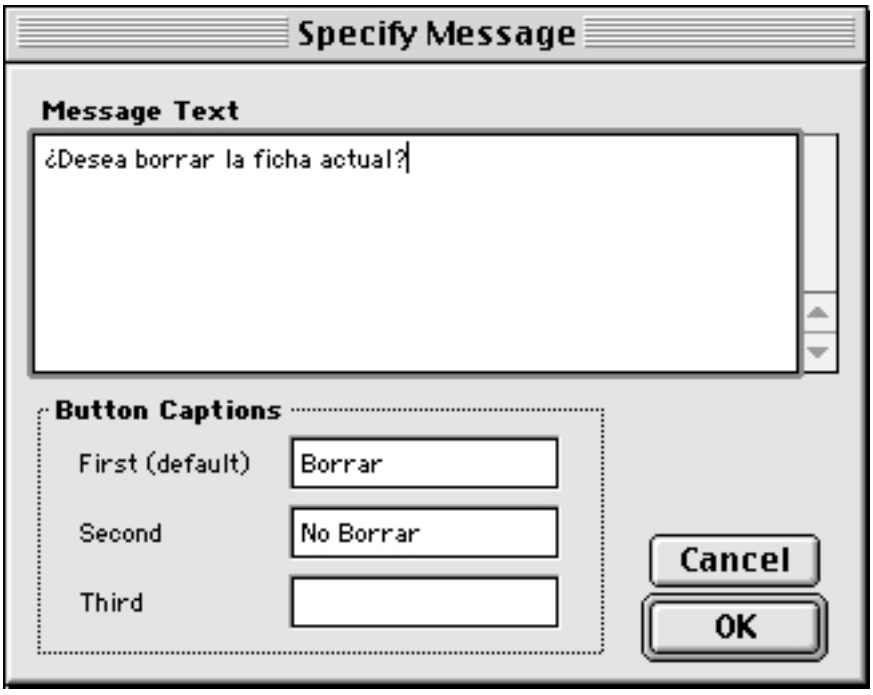
Por último, recordar que el salto de línea entre el código de AppleScript se debe obtener a través del carácter especial ("¶") que nos ofrece FileMaker.

Diálogos (Adiciones Estándard)

Un segundo ejemplo, más ilustrativo (e impactante) sería el de los diálogos. Hasta la versión 6, donde los diálogos pueden personalizarse, FileMaker nos permite hacer dos tipos de diálogos esperando la interacción del usuario, y contra los cuales podremos actuar para afrontar nuestras acciones en base a uno u otro resultado. Por ejemplo, Muchas veces he llegado a ver gente que borra archivos, datos, etc… sin caer en la cuenta de los datos que existen en el mismo. Así podríamos avisar al usuario con cualquiera de los diálogos siguientes:



La diferencia entre ambos estriba en la forma en que se preparan los diálogos. El primer diálogo de borrar está creado por FileMaker, y el segundo está creado por AppleScript. En ambos, se otiene un parámetro que se podrá manejar posteriormente, de forma automática, en FileMaker, o definiéndolos en AppleScript. Así, nuestro código sería:



Este es el cuadro donde se introduce el mensaje a mostrar. Los parámetros de retorno los maneja FileMaker, por lo que se deben incluir las instrucciones correspondientes inmediatamente después a la ejecución de este paso de guión extrayendo éste con un cáculo, ya sea lógico o de cualquier otro tipo, donde se incluya "Status (CurrentMessageChoice)".

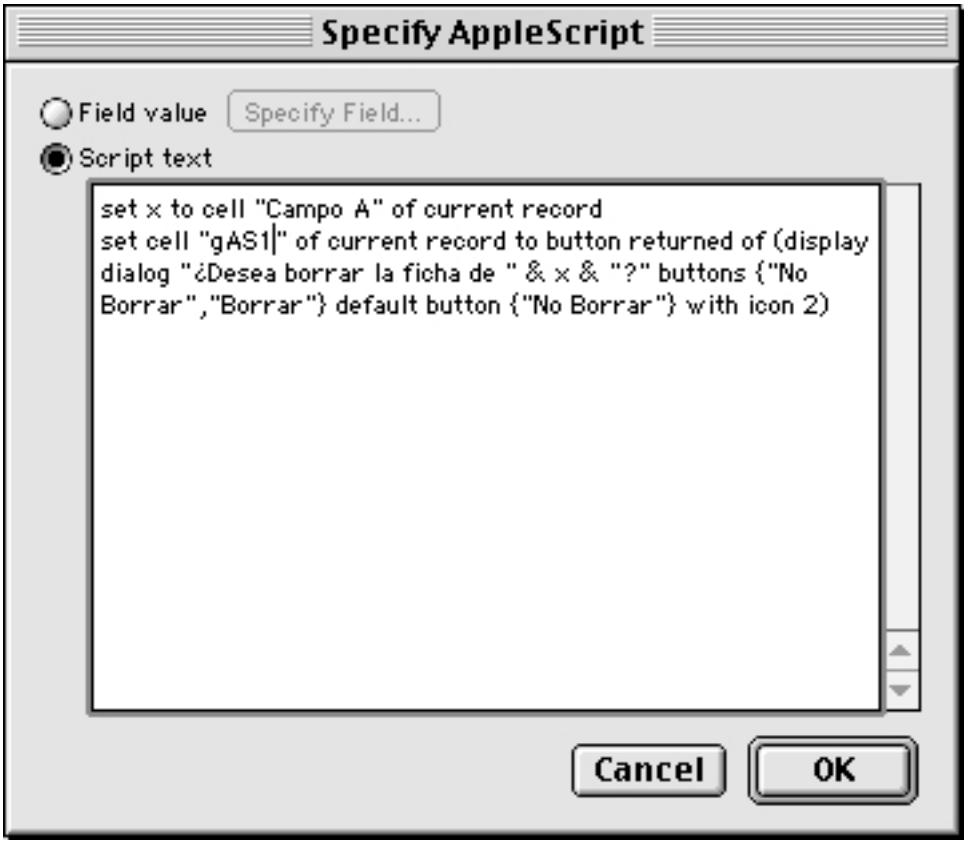
El cuadro siguiente muestra nuestro código de AppleScript dentro de un cuadro creado con el paso de guión "Ejecutar AppleScript".

En este código observamos los siguientes pasos:
se toma el valor que queremos mostrar en base al contenido del campo del registro actual
" **set x to cell "Campo A" of current record** ",
asignamos a un campo del parámetro de retorno en base al valor contenido por el botón que se seleccione, veremos más adelante que los valores de los botones también se los definimos nosotros" **set cell "gAS1" of current record to button returned of** ",
definimos el mensaje que queremos mostrar
" **(display dialog "¿Desea borrar la ficha de " & x & "?" ...)** ",
le indicamos el texto de los botones que se mostrarán y el botón que marcamos por defecto "

…buttons {"No Borrar","Borrar"} default button {"No Borrar"}… “
asímismo, podremos añadir algo de adorno a nuestro diálogo
“ …with icon 2) “.

En este caso, sólo se pasa a un campo, que puede ser de cualquier tipo, para mostrar la ventaja obtenida frente al correspodiente paso/s que deberíamos crear con ScriptMaker, aunque con AppleScript también se pueden desarrollar guiones más complejos, estos podrán necesitar igualmente la ayuda de algún paso de ScriptMaker para concluir la tarea que deseamos realizar con el guión que estamos preparando.

Esta forma de diálogo nos evita tener un manejo a posteriori y de forma inmediata del parámetro de retorno pudiendo dar preferencias a otros casos, por ejemplo, si implantamos un sistema de gestión de errores en nuestra base de datos.

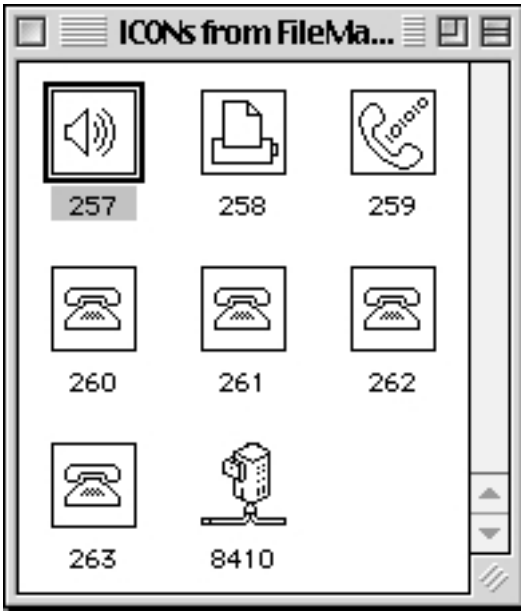


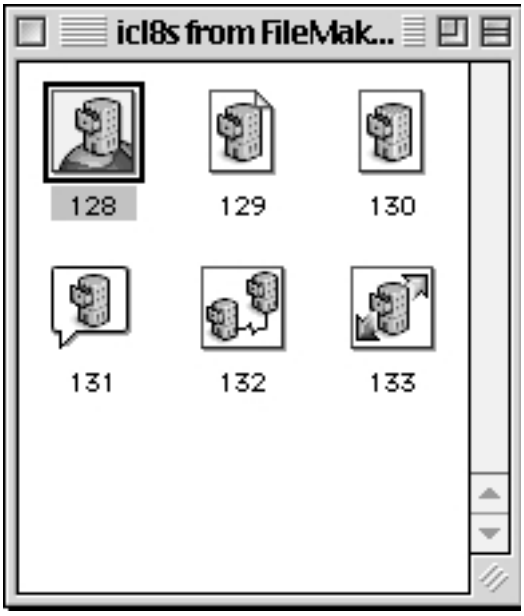
En este último cuadro podemos observar, además, que FileMaker nos da la la opción de pasar el código del AppleScript que queremos ejecutar a partir del valor de un campo, lo cual, aumenta aún más la potencia de este paso de guión ya que, por ejemplo, podríamos crear un diálogo distinto cada vez, según la necesidad de nuestra tarea. Por ejemplo, al desarrollar una base de datos donde llevemos un control de nuestras cuentas, podremos crear un diálogo para indicar el saldo actual en el momento conveniente cambiando el icono a mostrar según la importancia que queramos dar al mismo, y ofreciendo por defecto distintos botones; todo ello sin tener que crear un guión de AppleScript muy complejo ya que lo podemos incorporar en un campo de cálculo de FileMaker.

Otros parámetros opcionales son:

[default answer "Respuesta por defecto"] -- El texto por defecto, en una caja editable
[buttons {"Botón 1","Botón 2","Botón 3"}] -- una lista de hasta tres botones
[default button 1 o "Botón 1"] -- el número o texto del botón por defecto
[with icon stop/note/caution 0/1/2...] -- ...el icono que se muestra en el diálogo, indicado por nombre o número.

El icono que podemos mostrar puede ser cualquiera de los incuidos en el programa creado en AppleScript, el programa al cual o desde el cual se ejecuta el AppleScript, y, por último, uno de los del Finder. Para localizarlos, AppleScript los busca en este mismo orden. Para poder ver los iconos que tiene un programa se puede utilizar un programa de edición de recursos, como es Resedit. A continuación podemos observar los iconos propios de FileMaker 4, junto con su número, que es el que podemos indicar en este parámetro.



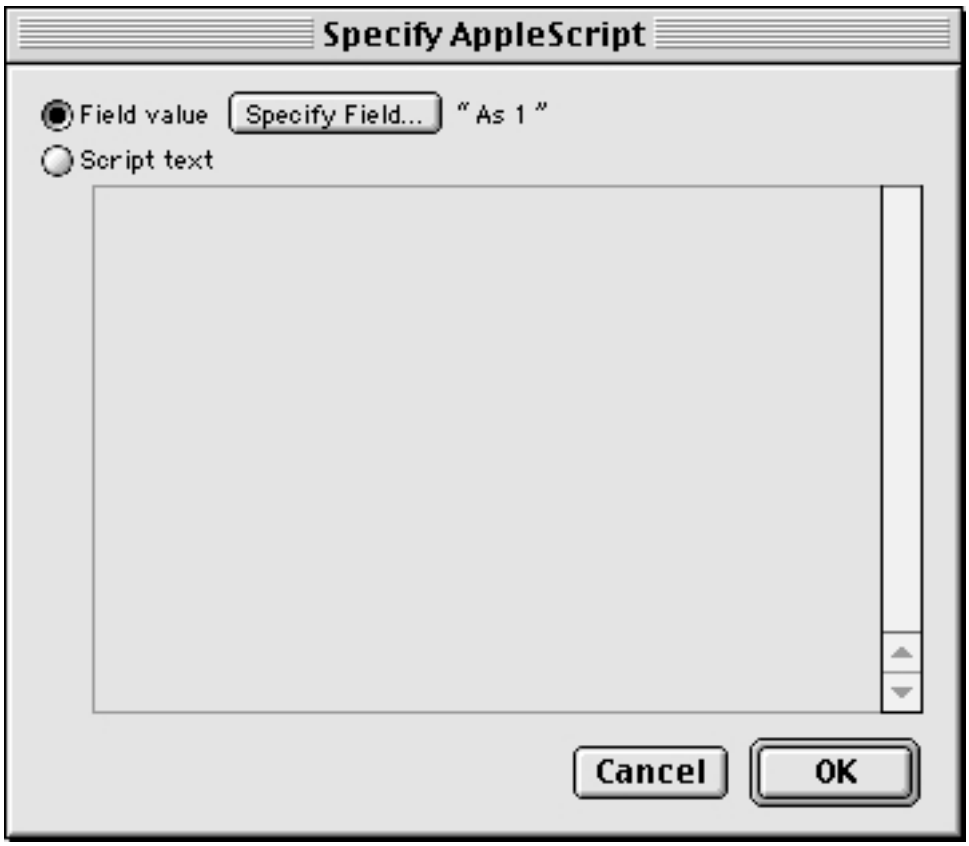


[giving up after 10] -- número de segundos tras el cual, el diálogo desaparecerá.

Listas de Valores (Adiciones Estándard)

Una de las herramientas que nos ofrece FileMaker es la generación de listas de valores. Estas pueden generarse de forma estática, por ejemplo, cuando sabemos de antemano qué valores pueden darse en un campo determinado, y también se pueden generar de forma dinámica a través de aquellos valores contenidos en el mismo campo teniendo en cuenta todos los registros de la base de datos. Por ello, nos podremos beneficiar de la potencia que nos ofrece AppleScript al poder manejar también las listas de valores. Así, podremos solicitar al usuario de la base de datos que escoja un valor, y, a partir de su decisión, ejecutar las acciones correspondientes para poder concluir la tarea que deseamos. Por ejemplo, y retomando nuestra base de datos donde llevemos un control de nuestras cuentas, podremos cambiar entre distintas presentaciones que hayamos generado anteriormente en nuestra base de datos para poder anotar un apunte que nuestro banco nos haya comunicado utilizando un formato similar al que recibimos por parte de este.

Para ver el desarrollo de este guión utilizaremos la propiedad que nos ofrece FileMaker de ejecutar un AppleScript a partir del contenido de un campo. Así crearemos nuestro código de AppleScript en un campo global, que denominamos gAS1, utilizado a modo de variable, que luego vaciaremos.



En este campo, introduciremos, a través de un cálculo, nuestro código AppleScript de la siguiente forma:

```
"set x to {}" & "¶" &
```

Iniciamos nuestra variable, en AppleScript, a la que le otorgamos formato de lista. Esto nos evitará un error al tener una lista no válida si no tuviéramos ningún valor en nuestra lista de valores, por lo que no se generaría.

```
"set x to x & {""" &
```

Retomamos nuestra variable e iniciamos la lista de valores de AppleScript comenzando a incluir la primera comilla para identificar nuestro primer valor de la lista en AppleScript.

```
Substitute(
ValueListItems(Status(CurrentFileName); "Lista");
"¶";
""","""
)
```

A continuación obtenemos nuestra lista de valores existente en nuestra base de datos. Esta nos la dará FileMaker al utilizar "`ValueListItems (dbname; valuelist)`". En este ejemplo, utilizaremos "`Status(CurrentFileName)`" para no tener problemas a la hora de identificar la base de datos con la que estamos trabajando. Asimismo, tendremos que cambiar el retorno de carro que nos

da FileMaker como resultado ante nuestra petición de " ValueListItems " por una coma para que sea inteligible por AppleScript como separación entre valores. Aprovechamos este cambio, asímismo, para introducir las comillas que encierran a los valores.

```
& """} as list" & "¶" &
```

Tras ello, cerramos nuestra lista de valores e indicamos a AppleScript que estamos manejando listas de valores

```
"set cell ""Campo A"" of current record to (choose from list x)"
```

Por último, generamos en pantalla nuestra lista de valores, y, el resultado, se lo pasamos al campo " "Campo A" " para poder manejarlo a posteriori. Una vez obtenida la decisión, es decir, que le indicamos a nuestra base de datos de qué banco hemos recibido el justificante, realizaremos las acciones correspondientes para que nuestra base de datos nos ofrezca una interfaz más amigable para introducir los datos correspondientes al nuevo pago de la hipoteca de nuestra casa.

Otras opciones que podemos manejar en nuestra lista de valores son las siguientes:

[with prompt "Mensaje"] -- El mensaje que daremos para indicar que se escoja un valor
[default items {"Lista"}] -- lista de elementos seleccionados por defecto.
[OK button name "Botón OK"] -- el nombre del botón OK
[cancel button name "Botón Cancelar"] -- el nombre del botón Cancelar
[multiple selections allowed true] -- ¿Se permite una selección múltiple?. Por defecto es false
[empty selection allowed true] -- ¿Se puede hace click sobre el botón OK sin seleccionar ningún valor?. Por defecto es false

Código interno o código externo

En algunas ocasiones, será necesario que se preparen pequeñas aplicaciones de AppleScript de forma que podamos recurrir a ellas cuando nos sea necesario. Dado que algunas acciones que admite FileMaker se tienen que hacer "hablándole" a la aplicación misma, es por lo que necesitamos estar "fuera de ella", así se crearía la comunicación entre nuestro AppleScript y FileMaker. Un ejemplo de ello son los menús, por ejemplo, si queremos crear una base de datos donde también tengamos que manejar las presentaciones, (¡sí, me refiero al "Modo Presentación"!) sin necesidad de estar nosotros presentes, no nos queda más remedio que hacerlo a través de AppleScript, ya que ScriptMaker no nos lo permite. Mientras que, como hemos visto hasta ahora, cuando queremos establecer una comunicación entre AppleScript y los objetos como son archivos, registros, campos… de FileMaker lo podremos hacer desde dentro de la aplicación en tanto que estos son elementos manejados por la misma, hay otros que son propios de la aplicación, es decir, estamos "hablando" con el objeto que está en el "nivel 1" de la jerarquía de contenido. Igualmente, podremos desarrollar algún guión que maneje datos de FileMaker mientras el usuario sigue trabajando con Filemaker.

Guiones comunes

¿Quién no ha desarrollado una aplicación con dos o más archivos de base de datos en los que hay un guión que se desarrolla de la misma forma en todos los archivos? (Por ejemplo, "Ir al

Menu Principal"). Pues bien, con AppleScript ya no tendremos que perder ese tiempo escribiendo lo mismo, y podremos aprovecharlo para tomar un refresco (entiendo que a estas alturas el lector ya ha cogido tal habilidad con AppleScript que no se tenga que preocupar de concentrarse mucho en este texto…). Para ello se dispone de los denominados "*Menus Externos*". Al usar los objetos *Menu* y *Menu Item*, otros programas podrán crear menús en FileMaker, así como enviar el evento *Do Menu* a los menus existentes en FileMaker. FileMaker soporta la adición de menus personalizados bajo el submenú "*External*" que aparecerá bajo el menú de *Guiones*. Cuando una aplicación crea un nuevo menú o elemento de menú, envía un evento *Create* con un parámeto de datos iniciales especificando las propiedades del nuevo item de menú.

Las propiedades del item de menu que se deben fijar con este evento son:
1) la propiedad *Notify Address* (0 para los menus normales de FileMaker). Esto debería ser el ID de proceso de la aplicación que será notificada cuando se seleccione el nuevo item del menu. En este caso, será válido con indicar su nombre.
2) la propiedad *Unique ID*.

```
create menu item with properties {name:"Guión Común", ID:111}
```

Cuando se seleccione la nueva opción de menu, FileMaker envía un evento de Menu seleccionado (class ID: kMEN, event ID: kMIS) a la aplicación que la contiene, con el ID único así como el Notify Address del menú seleccionado.

Ejemplos de FileMaker

Hasta la versión 6 de FileMaker, se instalaban, también unos ejemplos de uso de FileMaker y AppleScript bajo la carpeta "FileMaker and Apple Events". Estos ejemplos dan algunas ideas que muestran la potencia que nos puede dar nuestro desarrollo usando ambas aplicaciones. Estos ejemplos son:

Applet Example. Este es un ejemplo muy potente. Muestra cómo se puede enviar a una aplicación hecha en AppleScript "petciones" para realizar una función dados unos parámetros. El funcionamiento es muy básico y fácil de implementar ya que, desde FileMaker sólo ejecutamos el programa AppleScript, y éste se encarga de evaluar un valor que tenemos en un campo y, en función de este, realizar distintas funciones.

Copying Records: Este ejemplo muestra cómo copiar registros entre dos bases de datos. Para ello, primero busca los registros cuyo primer nombre sea "Fred" en la base de datos de la que se va a copiar, los ordena, y los toma en forma de lista para poder crear un registro nuevo con cada uno de los valores en el archivo donde se va a hacr la copia de estos

Creating Menus: Este ejemplo muestra cómo se pueden generar menús externos a FileMaker. Estos menús se podrán utilizar para realizar funciones ajenas a FileMaker o funciones no incluídas en la versión de FileMaker que tengamos.

Deleting Duplicates: Este guión busca y borra registros duplicados. Para ello utiliza un campo de cálculo, pero se puede hacer con cualquier campo que sea válido.

Finding Records: Este guión muestra diferentes formas de buscar registros a través de AppleScript. Es, igualmente, un ejemplo muy ilustrativo de cómo se puede usar a un Applet (programa generado en AppleScript) para que ejecute una acción pasándole parámetros.

Graphing Example: Este ejemplo muestra cómo se puede compartir información con Microsoft Excel a través de AppleScript. En este caso se usa Excel para crear un gráfico y devolver el resultado de nuevo a la base de datos.

HyperCard Example: Para los sistemas más antiguos, también se incluye este ejemplo para poder usarlo con HyperCard

Importing Pictures: Este es un ejemplo que muestra cómo importar un gráfico a una base de datos. En este caso lo que hacemos es utilizar la ruta al archivo en vez de importar el gráfico en la base de datos misma. Es como si utilizáramos la opción “Insertar->Gráfico” y marcáramos la opción de “Guardar una referencia al achivo”

Matching Data: Este ejemplo es, también muy potente ya que muestra cómo mostrar unos registros en base a un campo que coincida con otro de una segunda base de datos. Es como si generáramos una relación “al vuelo”. Se puede utilizar, por ejemplo, para poder entregar soluciones hechas con FileMaker donde se puedan generar relaciones sin necesidad de editar el código.

Relational Data: Este ejemplo, el último que se instala, nos muestra una forma de poder trabajar con registros y datos relacionados. En este caso, al contrario que el anterior, las relaciones tienen que existir en la base de datos.

Conclusión

El uso de Filemaker Pro junto con AppleScript puede dar soluciones a programas que incorporan características avanzadas como la ejecución de guiones de FileMaker de forma condicional, control de menús, y comunicación entre programas. Tras las ideas mostradas en este artículo, se puede probar alguna o todas las ideas siguientes de guiones:

. Controlar FileMaker. Lógicamente, si podemos enviar eventos fuera de FileMaker, también podemos enviárselos a FileMaker, por lo tanto, podremos controlar funciones que pueden ser críticas, por ejemplo, comprobar que está funcionando un servidor web de FileMaker, hacer que, tras un tiempo definido se cierren las bases de datos y se vuelvan a abrir con una clave distinta a la que usábamos, realizar funciones cada cierto tiempo, como generar alarmas a una hora y un día en concreto, etc...

. Controlar otras aplicaciones con el clic de ratón desde FileMaker. Por ejemplo, crear un correo fusionando datos de FileMaker con AppleWorks. Crear una hoja de cálculo de Excel basada en los datos obtenidos de una base de datos de FileMaker.

. Manipular datos entre bases de datos de FileMaker. Las herramientas de programación como AppleScript tienen funciones de búsqueda y sustitución de textos muy potentes que permiten extraer, editar, y reemplazar datos de formas muy creativas. AppleScript permite ahorrar tiempo de forma real en casos donde se necesita realizar cálculos de texto o cuando se necesita exportar y manipular datos a otras aplicaciones.

. Creación de diálogos. Usando AppleScript se pueden diseñar diálogos sofisticados, paletas, alertas, y pantallas para bases de datos de FileMaker o como intermediario entre diferentes apli-

caciones. Por ejemplo, con FrontMost (una herramienta de desarrollo incluida con el Kit de Desarrollo de AppleScript 1.1), se pueden crear paletas flotantes con una lista de guiones que están disponibles para una presentación en particular, o crear un diálogo de búsqueda y reemplazo, o una paleta de publicación con menús de fuente, estilo, cuerpo, y color.

Aprender cómo se realizan guiones implica familiarizarse con un lenguaje de programación como AppleScript y conocer los eventos y objetos que soportan los programas. Espero que este texto le ayude a comenzar a experimentar con AppleScript y se replantee cómo usar FileMaker.

Diccionario de AppleScript en FileMaker 5.5

A continuación se incluye el diccionario AppleScript incluido en la versión 5.5 de FileMaker para su comprensión. Este se ha extraído de la base de datos de AppleEvents que se instala cuando seleccionamos una instalación standard.

Eventos

Begin Transaction kAEMiscStandards kAEBeginTransaction
¿Qué se puede hacer?
El evento Begin Transaction se utiliza para provocar que FileMaker admita Apple Event desde la aplicación que lo provocó, o desde aquella que devuelva el ID (identificador único) que pasó FileMaker.

Este evento es más comúnmente utilizado en un entorno de red donde muchos usuarios realizan transacciones de forma simultánea, y cada uno de estos necesita disponer de la base de datos en un estado determinado.

Ejemplos

```
tell application "FileMaker Pro"
 set i to 1
 with transaction
  repeat with i from 1 to the number of records in database 1
   set cell 1 of record i of database 1 to i
  end repeat
 end --transaction
end tell
```

Nota: Tras compilar el script anterior, al comando "end" tras "end repeat" se traducirá como "end transaction" por defecto. El script será funcional de esta forma, pero después que se haya recompilado tras cualquier edición del mismo, la palabra "transaction" se deberá borrar.

Class Info kAECoreSuite kAEGetClassInfo
¿Qué se puede hacer?
Al obtener la información "class" de un objeto podremos ver cómo se trata mostrándonos FileMaker qué objetos son elementos de ese elemento (pertenecen directamente a la jerarquía del objeto), y qué propiedades tiene el objeto.

Su sintaxis es muy sencilla. Todo lo que se necesita es indicar el nombre del "class". Los nombres de classs de FileMaker son: Application, Window, Document, Database, FileMaker Script, Layout, Field, Record, Cell, Menu, y Menu Item.

La información devuelta por Class Info está enfocada a audiencia técnica. La mayoría de los usuarios finales, particularmente usuarios de AppleScript, no necesitarán acceder a esta información. Consulte la base de datos de eventos que se instala junto con FileMaker.

Nota: Serán comprensibles todos los comandos Get. Por tanto se puede decir get class info of

record, o sencillamente class info record.

Ejemplos

```
tell application "FileMaker Pro"
 class info application
 get document class info
 window class info
 class info menu
 class info database
 FileMaker script class info
 class info menu item
 record class info
 get layout class info
 class info field
 cell class info
end tell
```

Close kAECoreSuite kAEClose
¿Qué se puede hacer?
Con el evento Close podremos cerrar ventanas, documentos y bases de datos.

Para utilizar este evento se necesita especificar el objeto al cual nos referimos ya sea window, document, o database, por ejemplo, window 1, o document "Test File".

Ejemplos

```
tell application "FileMaker Pro"
 close window "Addresses"
 close database "Músicos"
 close document 1
end tell
```

Count kAECoreSuite kAECountElements
¿Qué se puede hacer?
Podremos contar los items de un objeto en particular. Esto se realiza normalmente cuando se cuentan los elementos de ciertas características en otro objeto. Este evento es una de las razones por las que la comprensión de la jerarquía de objetos en FileMaker, también conocido como contenedor, es importante.

A continuación se muestran los objetos contra los cuales el evento Count puede operar:
Windows
Documents
Databases

FileMaker Scripts
Layouts (la presentación 0 nunca se cuenta)
Records
Requests
Fields
Cells

Ejemplos

Se puede decir: Para contar:

```
tell application "FileMaker Pro"
        count of record -- registros de un archivo (o peticiones de búsqueda)
        count record of database 1 -- registros de un archivo (se ignoran la búsqueda)
        count request -- peticiones en un archivo
        count field -- campos en un archivo
        count FileMaker script of window 1 -- scripts de FileMaker en el archivo
        count of cell of document 1 -- celdas en el archivo (incluyendo la búsqueda)
        count cell of record 2 -- celdas en el registro
end tell
```

Create New kAECoreSuite kAECreateElement
¿Qué se puede hacer?
Con el evento Create (New) se pueden crear nuevos registros, peticiones de búsqueda, menus y elementos de menu.

Al crear un nuevo registro o petición, FileMaker devuelve una referencia del objeto nuevo en forma de un indicativo de objeto tal como
record ID 5 of database "Músicos" of application "FileMaker Pro".

Se pueden añadir registros dentro de un conjunto de estos, o tras realizar una búsqueda indicando, en ambos casos, los objetos document o database. Por defecto, FileMaker añadirá registros nuevos al conjunto de registros encontrados. No obstante, al especificar el objeto database con el evento create se añadirá el nuevo registro al conjunto de registros completo, incluyéndolo así "en su entorno" por ejemplo cuando se está mostrando el resultado de una búsqueda. Esto nos permite una mayor eficacia a la hora de crear registros.

Cuando se crean objetos, hay que tener en cuenta que se tiene cierto control sobre dónde se coloca el nuevo objeto, y si este tenía datos inicalmente o sobre otra propiedad. Se tiene que usar el parámetro de localización "at" para especificar un layout, database, document, o window en los que se quiere crear el objeto, y la opción "with properties" para fijar una propiedad de este como puede ser la propiedad "omitted" de una petición. Para ir a un registro o petición, se puede usar el evento Go To como en "go to (create record)".

Todos los menus definidos por el usuario se crean en el "External menu", que aparece la primera vez que se crea un nuevo menú externo. Estos menus pueden estar anidados. Ver los puntos Menu y Menu Item para más información.

Nota: FileMaker puede notificar a cualquier aplicación que cree un elemento de menu en su

menu External que se ha seleccionado un elemento de menu. Salvo la versión actual del Editor de Scripts ya que este no responde a Apple events, y por tanto no tiene forma de responder a los elementos de menú creados por este en FileMaker. Las aplicaciones como HyperCard y Excel pueden responder a elementos de menú que haya en FileMaker.

Ejemplos

Se puede decir: Para Crear:

```
tell application "FileMaker Pro"
        create new record -- registro (en los registros de la búsqueda actual y sin datos)
        create new record with data "J.S. Bach" -- registro (con datos en un campo)
        create new record with data {"Katchaturian", "Mediocre"} -- registro (con varios campos)
        create new record at database 1 -- registro (en el conjunto completo)
        create new request -- petición de búsqueda (sin datos)
        create new request with data "Katchaturian" -- petición de búsqueda (con un campo)
        create new request with data {"Katchaturian", "Mediocre"} -- petición de búsqueda (con varios campos)
        create new request with properties {omitted:true} --petición de búsqueda (con la opción de omitir)
        create new menu with properties "MiMenu" -- menu
        create new menu at menu "External" with properties "Menu 2" -- menu dentro de un menu (o submenu)
        create new menu item with properties "Elemento de Menu" -- elemento de menu
        create new menu item at menu "Menu 2" of menu "External" with properties "Elemento de Menu" -- elemento de
menu (de un menu)
end tell
```

Size kAECoreSuite kAEGetDataSize

¿Qué se puede hacer?

Use el evento Data Size para obtener el tamaño de los datos (en bytes) contenidos por un objeto. Puede ser útil para obtener el número de caracteres en un campo, mientras que haya un byte por carácter en un campo.

Nota: Data Size devuelve el tamaño de la información manejada por los Apple events y n siempre se corresponde con cómo guarda la información FileMaker. Por ejemplo, la frase "data size of current record" devolverá un valor como 136 que corresponde al número de bytes del objeto referenciado como registro 1 de la base de datos "Compact Discs" de la aplicación "FileMaker Pro" no a la información contenida en el registro.

Ejemplos

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        data size of every field of layout 0 -- contar los caracteres de una base de datos
        data size of cell "Compositores" of last record -- contar los caracteres de un campo en particular
        data size of cell "Compositores" of record 1 as "TEXT" -- obtener el tamaño de un campo como valor de texto
        data size of records 1 through 4 -- obtener el tamaño de un grupo de registros
end tell
```

Delete kAECoreSuite kAEDelete
¿Qué se puede hacer?
El evento Delete nos permite borrar registros, peticiones de búsqueda, menus, y elementos de menu.

Nota:
- Los menus y elementos de menu que se pueden borrar únicamente son aquellos creados como menu externo.
- Al borrar todas las peticiones, se obtendrá una petición vacía (request 0) hasta que se cree otra o hasta que la base de datos se cambie a otro modo en cuyo momento se creará una petición inicial en blanco (request 1).

Ejemplos

Se puede decir: Para borrar :

```
tell application "FileMaker Pro"
        delete record 1 -- primer regisatro (incluye el conjunto actual)
        delete last record -- último registro (incluye el conjunto actual
        delete every record -- borrar todos los registros del conjunto actual
        delete every record of database 1 -- borrar todos los registros de la base de datos
        delete every request -- borrar todas las peticiones
        delete menu item 1 of menu "Externo" -- primer elemento de menu del menu "Externo"
        delete menu item "elemento de menu" of menu "Externo" -- el elemento de menu "elemento de menu"
        delete menu 2 of menu "Externo" -- segundo elemento de menu del menu "Externo"
        delete (every record whose cell 1 = "x") -- todos los registros que cumplan la condición
end tell
```

(Otros operadores se pueden incluir con la cláusula "whose" como: <, ≤, >, ≥, ≠, AND, OR, NOT, begins with, ends with, contains).

Do Menu kAEMiscStandards kAEDoMenu
¿Qué se puede hacer?
Con el evento Do Menu se pueden ejecutar comandos de menu de FileMaker.

Hay que recordar lo siguiente cuando se trabaje con menus:

• Es posible acceder a menus anidados especificando el anidamiento (ver abajo).
• Los items de Menu a los que se refiere por nombre deben coincidir exactamente con el que aparece en FileMaker. Donde aparezcan los puntos suspensivos, estos se tienen que crear teclando Alt+Intro.
• Se debe incluir el menu cuando nos refiramos a un item de menu (por ejemplo, do menu menu item "New Record" de menu "Edit").
• Algunos items de menu están disponibles sólo en algunos modos (por ejemplo, "Nueva Presentación…" sólo está disponile en modo presentación).
• Los menús están en el “nivel 1” dentro de la jerarquía de contenido, por lo tanto, no podremos utilizar este evento más que “desde fuera” de FileMaker.

Nota: En tanto que todos los items de menu están deshabilitados cuando se está ejecutando un guión de ScriptMaker, no se podrá usar el evento Do Menu para seleccionar items de menu mientras se esté ejecutando el guión. Además no se podrá utilizar eficazmente el estado de un item de menu para determinar el modo actual de la aplicación.

Ejemplos

Se puede decir: Si se quiere acceder a:

```
tell application "FileMaker Pro"
        do menu menu item "MiltemdeMenu" of menu "External" -- menu definido por el usuario
        do menu menu item "Visualizar" of menu "Modo" -- un item de menu de FileMaker
        do menu menu item "Negrita" of menu "Estilo" of menu "Formato" -- item de menu de un submenu
        do menu menu item 6 of menu 5 -- item de menu (omitido) por índice
end tell
```

Do Script kAEMiscStandards kAEDoScript
¿Qué se puede hacer?
El evento Do Script sirve para ejecutar un guión del menu de guiones de FileMaker.

Para utilizar el evento Do Script se necesita indicar un identificador del objeto, como FileMaker script 1, o last FileMaker script, o FileMaker script "Imprimir Informe de Ventas". (Siempre hay que referirse a un objeto guión como FileMaker script <x>; la palabra "FileMaker" es parte del nombre del objeto.)

Do Script puede ser útil al escribir guiones que recuperen búsquedas complejas, ordenen, sigan un orden de importación, y ajustes de páginas, y luego llamen al guión para realizar el trabajo correspondiente, particularmente si se realiza una impresión. Los guiones de FileMaker ofrecen la posibilidad de recuperar ajustes de página, imprimen sin diálogos, y demás, que son características que no están dispoibles para Apple Events.

Nota: el evento Do Script se realizará mejor cuando empiezan los guiones de Filemaker. Dado que Fiemaker no es capaz de procesar Apple Events cuando está ejecutando un guión, es posible recibir un error de tiempo (timeout) cuando se envían eventos a FileMaker al enviar un evento Do Script. A continuación se ofrecen algunas sugerencias para evitar los errores de tiempo:

1) alargar el periodo de espera utilizando una frase como "with timeout of 600 seconds"
2) implemente un mecanismo de control de flujo usando FileMaker para enviar un evento al applet para seguir con el próximo evento a FileMaker.
3) use una ruutina para enviar contínuamente eventos a FileMaker y continuar sólo cuando FileMaker esté preparado. Realice llamadas a la rutina WaitUntilReady tras enviar el evento Do Script y tras enviar otro evento a FileMaker. A continuación hay un ejemplo de tal rutina:

```
on EsperaFilemaker()
        delay 5 -- ejecutamos el guión cada 5 segundos
        try
                tell application "FileMaker Pro" to get version -- pedimos a FileMaker la versión
```

```
                return
        on error number errnum
        end try
end EsperaFilemaker
```

Nota: En tanto que sólo se puede ejecutar un guión de FileMaker al mismo tiempo, cualquier guión llamado utilizando el evento Do Script al ejecutar un AppleScript utilizando el paso "Do Script FileMaker Script" se ejecutará tras completar el guión.

Ejemplos

Se puede decir: Si se quiere hacer:

```
tell application "FileMaker"
        do script FileMaker script 1 -- el primer guión
        do script last FileMaker script -- el último guión
        do script FileMaker script "Imprimir informe de ventas" -- un guión por su nombre
        do script FileMaker script after FileMaker script 1 -- el segundo guión (que es el posterior al guión 1)
        do script FileMaker script before FileMaker script 3 -- el segundo guión (que es el anterior al guión 3)
        do script (every FileMaker script whose name begins with "Nombre") -- todos los guiones que cumplan la condición de que el nombre empiece pos "Nombre"
end tell
```

Duplicate kAECoreSuite kAEClone

¿Qué se puede hacer?

Use el evento Duplicate para duplicar registros, peticiones de menus, y elementos de menu en un archivo abierto.

Al duplicar registros o peticiones, FileMaker devuelve una referencia al nuevo objeto de la misma forma que un identificador de objeto como record ID 5 of database "Músicos" of application "FileMaker Pro".

Los registros duplicados se pueden añadir al conjunto de registros actual o a todos los registros si se especifica el ojeto document o database. Por defecto, FileMaker añade nuevos registros al conjunto encontrado. Especificando el ojeto database con el evento duplicate, no obstante, añadirá el nuevo registro a un conjunto de registros completo. Esto nos da resultados más rápidos cuando se duplican registros.

Cuando se duplican objetos, tenga en cuenta que se tiene algo de control sobre dónde se coloca el nuevo objeto. Use el parámetro de localización "to" para especificar una presentación, base de datos, documento, o ventana en donde se duplicará el objeto. Para navegar a un registro duplicado o petición, use el evento Go To como en un go to (duplicate record 1).

Ejemplos

```
tell application "FileMaker"
        duplicate record 2 -- duplicamos el registro 2 en el conjunto actual
        duplicate record 2 of window "basura" -- duplicamos el registro 2 en el conjunto actual
        duplicate record 1 to database 1 -- duplicamos el primer registro en la base de datos (no se muestra en el conjunto actual)
```

```
        duplicate record 2 to document 1
        duplicate request 1 -- duplicamos una petición de búsqueda
end tell
```

End Transaction kAEMiscStandards kAEEndTransaction
¿Qué se puede hacer?
Use el evento End Transaction par concluir una transacción que se haya comenzado con la sentencia Begin (o With) Transaction.

La mayoría de los usuarios no necesitrarán precisamente esta utilidad. Pero en un conjunto en grupo, donde muchos usuarios pueden tener múltiples transacciones ejecutándose simultáneamente, y cada uno pretende que la base de datos esté en un estado en concreto, pueded ser un evento útil.

Para finalizar una transacción, se tiene que pasar el ID de la transacción de esa transacción como un atributo del evento End Transaction, no como parámetro. Cuando se termina una transacción, FileMaker permite que otra aplicación manipule la base de datos, procesando los Apple Events en el mismo orden en el que se recibió.

Ejemplos

```
tell application "FileMaker Pro"
        set i to 1 -- iniciamos el valor del contador
        with transaction
                repeat with i from 1 to the number of records in database 1 --generamos un bucle
                        set cell 1 of record i of database 1 to i -- establecemos el campo 1 al valor de i
                end repeat
        end --transaction
end tell
```

Nota: Tras compilar este guión, la frase "end" siguiente a "end repeat" se convertirá en "end transaction". El guión será funciona de esta forma, pero después que se haya recompilado el guión correctamente tras cualquier modificación posterior, se deberá borrar la palabra "transaction".

Event Info kAECoreSuite kAEGetEventInfo

¿Qué se puede hacer?
Use el evento Info para obtener una lista de eventos pertenecientes a la clase especificada, con información detallada acerca de los eventos, incluyendo parámetros, parámetros de respuesta, identificadores, y demás.

La sintaxis para el evento Info es muy sencilla de usar. Todo lo que se necesita dar es un nombre de clase. FileMaker tiene seis clases de eventos: Core, Miscellaneous, Required, Database, FileMaker Pro, y URL. Están abreviados como core, misc, aevt, DATA, FMPR, y GURL respectivamente, para su uso con FileMaker.

La lista de eventos que devuelve Info está destinada a una audiencia técnica. La mayoría de usuarios finales, particularmente usuarios de AppleScript, no necesitarán acceder a la información contenida en el evento Info.

Ejemplos

Se puede decir: Para tener información acerca de:

```
tell application "FileMaker Pro"
        event info "core" -- eventos de la suite core
        event info "misc" -- suite de eventos varios
        event info "aevt" -- suite de eventos requeridos
        event info "DATA" -- suite de eventos de la base de datos
        event info "FMPR" -- suite de eventos de FileMaker
        event info "GURL" -- suite de eventos de URL
end tell
```

Exists kAECoreSuite kAEDoObjectsExist

¿Qué se puede hacer?

Use el evento Exists para estar seguros que los objetos que se quieren utilizar están disponibles. El evento Exists funciona con todos los objetos de FileMaker.

Exists no localizará valores vacíos. Por ejemplo, "name of record 1 exists" devuelve verdadero, en tanto que existe la propiedad, incluso aunque el nombre esté en blanco (porque los registros de FileMaker no tienen nombres).

Ejemplos

```
tell application "FileMaker Pro"
        FileMaker script "Ir a Ayuda" exists
        record 100 exists
        exists window 1
end tell

tell application "FileMaker Pro"
        if (exists window "Músicos") = false then -- Comprobamos que no tengamos abierto músicos
                open file "Macintosh HD:Músicos" -- si es así, lo abrimos
        else -- si la tenemos abierta
                show --la activamos
        end if
end tell
```

Find kAEFileMaker kAEFind

¿Qué se puede hacer?

Use el evento Find para completar la petición de búsqueda del conjunto actual en una base de datos. Para utilizar el evento Find sólo se tiene que especificar la ventana donde se va a realizar las peticiones de búsqueda. Si no se especifica ninguna ventana, el evento Find trabajará

con la base de datos que esté primero. Las peticiones de Find se pueden crear con el evento Create y se pueden editar con el evento Set Data.

Nota: Si se usa el evento Find cuando se ejecute un guión utilizando el paso de guión Ejecutar AppleScript, se provocará una pausa de ScriptMaker. En este caso habrá que utilizar el paso de guión Realizar Búsqueda.

Ejemplos

```
tell application "FileMaker Pro"
        delete every request
        create request
        set cell "Compositores" of request 1 to "Katchaturian"
        find
end tell
```

Get Data kAECoreSuite kAEGetData

¿Qué se puede hacer?

Con el evento Get Data se puede tener información de un objeto, como una celda. Se puede hacer también que la información que devuelva sea de un tipo determinado, como texto, numérico, integral o lista. El tipo por defecto que devuelve un objeto con elementos múltiples es una lista (ej.; al obtener información de un campo cuando hay muchos registros). El resultado del tipo de un objeto que devuelve un sólo valor es texto (ej.: un registro con un campo).

El evento Get Data se puede utilizar para obtener tipos de datos como imágenes y sonidos de un campo contenedor. Cuando se ha importado una imágen a un campo contenedor utilizando la opción "Guardar una referencia al archivo", se obtendrá una referencia al archivo donde está el dibujo. Asimismo, FileMaker permite establecer un archivo de referencia a un campo contenedor via Apple Events.

Nota: Se debe tener acceso de lectura para poder obtener información de las presentaciones, registros, campos, y celdas. Asimismo, la palabra "get" en AppleScript siempre se comprende. Por tanto, se puede decir "get data record 1" o "record 1".

Ejemplos

Se puede decir: Si se quiere obtener información de:

```
tell application "FileMaker Pro"
        every field of layout 0 -- todos los campos del archivo local
        every field of layout 0 as text -- todos los campos del archivo local, como texto
        cell "Canciones" of last record -- un campo del último registro
        cell "Preferencias" -- celda "Preferencias" del primer registro
        records 1 through 4 as text -- un rango de registros
        record after record 1 -- registro basado en una posición relativa
        record before record 3 -- registro basado en una posición relativa
        first record -- el primer registro
        last record -- el último registro
        middle record -- el registro del medio
```

```
        some record -- un registro aleatorio
        every record whose cell 1 = "Love" -- registros que cumplan cierto criterio
end tell
```

(Otras operaciones que están disponibles para su uso con la cláusula "whose" son , ≤, ≥, _, <, >, AND , OR, NOT, begins with, ends with, contains.)

GetURL kAEURL kAEGetURL
¿Qué se puede hacer?
Use el evento GetURL para abrir una base de datos de FileMaker por internet utilizando un navegador web, como Netscape™. El evento GetURL de la suite URL permite a las otras aplicaciones definidas en el Internet Helper abrir archivos especificados con la sintaxis del Uniform Resource Locator (URL).

Para usar el evento GetURL, primero ha de estar seguro de registrar FileMaker como protocolo con la aplicación helper utilizando la siguiente sintaxis. Nótese que sólo se necesita registrar la aplicación una vez.

```
tell application "Netscape Communicator™"
        register protocol "FMP3" for protocol "FMP3"
end tell
```

A continuación envíe el evento GetURL de la aplicación helper a FileMaker. El evento GetURL acepta una cadena representando una URL para una base de datos de FileMaker que se haya albergado via TCP/IP. Para habilitar la red TCP/IP, ajuste la preferencia general del protocolo TCP/IP tanto en el servidor como en el invitado.

Las bases de datos se pueden especificar utilizando cualquiera de las formas a continuación, donde servidor es la dirección IP o nombre de domínio de la máquina que alberga la base de datos y archivo es el nombre de la base de datos:

"FMP3://[servidor]/archivo"
"<FMP3://[servidor]/archivo>"
"URL:FMP3://[servidor]/archivo"
"<URL:FMP3://[servidor]/archivo>"

Si el servidor está vacío o se ha especificado con un asterisco, FileMaker buscará en la zona local el archivo.

Nota: Cuando se ejecute un AppleScript a través del paso de guión Ejecutar AppleScript, usando el evento GetURL provocará que el guión de ScriptMaker termine tras le ejecución del paso "Ejecutar AppleScript". Los pasos de guión incluidos tras el paso "Ejecutar AppleScript" que incuya el evento GetURL no se ejecutarán.

Ejemplos

```
tell application "FileMaker Pro"
        getURL "FMP3://claris.com/Product Catalog.FP3"
        getURL "FMP3://199.99.99.99/Product Catalog.FP3"
end tell
```

Para usuarios de Netscape:

Cree un documento HTML con la sintaxis siguiente:

<A HREF="FMP3://claris.com/Musicos.FP3"> Hacer Click aquí para abrir la base de datos Musicos</A>

Note que el archivo "Musicos.FP3", la dirección 199.99.99.99, y el nombre de dominio claris.com son meros ejemplos. Reemplace el nombre del archivo con el nombre de la base de datos a la que se quiere acceder y asegúrese que la IP o el nombre de dominio hacen referencia al servidor donde están las bases de datos.

Go To kAEFileMaker kAEGoto
¿Qué se puede hacer?
El evento Go To nos permite navegar entre objetos de FileMaker como bases de datos, presentaciones, registros, y celdas. Por ejemplo, el evento Go To se puede utilizar como tabulador para ir a un campo.

El evento Go To funcionará con los siguientes ojetos: ventana, documento, base de datos, registro, petición, y celda. El objeto debe estar abierto o disponible; así podría ser necesario abrir una ventana, documento, o base de datos antes de navegara algo contenido en estos.

El evento Go To se puede usar para copiar y pegar el contenido de un campo. Cuando se copie o se corte entre campos será necesario seleccionar primero algo. Esto se puede hacer con el evento Do Menu (por ejemplo do menu menu item "Seleccionar Todo" of menu "Ed"). Note que no es necesario ejecutar este paso para un campo contenedor ya que, con el simple hecho de ir a este campo se selecciona su contenido.

Hay que tener en cuenta también que cuando se transfiere información entre aplicaciones a través del portapapeles, el sistema opoerativo del Macintosh requerirá que se active la aplicación fuente y la de destino. Esto se puede hacer en AppleScript utilizando el comando "activate" dirigido a la aplicación antes de ejecutar un corte, copia o pegado.

Nota:
- Ejecutar un evento Go To cuando una base de datos está en modo vista previa o en modo presentación cambiará el modo a visualizar. Si se usa el evento Go To con una petición cambiará la base de datos a modo Buscar.

- FileMaker no soporta la navegación entre filas dentro de portales o repeticiones dentro de campos repetitivos más que la primera fila o repetición.

Ejemplos

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        go to record 1 -- ir al primer registro
        go to (create record) -- crear y seleccionar un registro
        go to record after current record -- ir al registro siguiente
        go to record before current record -- go to the previous record
        go to window 2 -- cambiar a otra ventana
        go to layout "Lista CD" -- mostrar una presentación distinta
        go to layout after current layout -- ir a la siguiente presentación
        go to layout before current layout -- ir a la presentación anterior
        go to cell 1 of record 1 -- entrar en un campo
        go to request 5 -- mostrar una petición de búsqueda
end tell
```

El ejemplo a continuación copia el contenido de un campo de un registro en particular y lo pega en un campo de otro registro.

```
tell application "FileMaker Pro"
        activate
        go to cell "Compositores" of record 1
        do menu menu item "Seleccionar Todo" of menu "Ed"
        copy
        go to cell "Compositores" of record 2
        paste
end tell
```

Open kAECoreSuite kAEOpen
¿Qué se puede hacer?
Use el evento Open para abrir bases de datos o documentos.

La sintaxis para el evento Open es muy sencilla de utilizar. Todo lo que se tiene que hacer es dar un alias, como "Macintosh HD:Musicos.fp3".

Se pueden abrir archivos que usen claves incluyéndolas como parámetro con el evento Open. Se puede incluso abrir más de un archivo protegido con clave cada vez si se usa una lista de claves en la lista de archivos. Si todos los archivos tienen la misma clave, se puede incluir esa clave, y FileMaker la utilizará cuando abra cada uno de los archivos que se hayan especificado. Asimismo, se pueden abrir archivos con una clave en blanco especificando una cadena vacía. Si una base de datos tiene una clave en blanco, el evento Open usará automáticamente esa clave al abrir archivos salvo que se especifique otra clave.

Hay algunas diferencias técnicas entre el evento Open Document y el evento Open. Véase la presentación de objetos FileMaker para obtener más información.

Nota: Use la sintaxis Open File desde AppleScript cuando se pretenda abrir un sólo archivo. Cuando se quieran abrir varios archivos, se ha de utilizar sólo el evento Open. Véase los ejemplos a continuación.

Nota: No se puede utilizar el evento Open para abrir una base de datos de FileMaker cuando se ejecute desde un paso de guión Ejecutar AppleScript. Utilice el paso de guión Abrir para abrir

todas las bases de datos necesarias antes de ejecutar el AppleScript.

Ejemplos

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        open file "Macintosh HD:Carpeta FileMaker Pro:Lista CD.fp5" -- abrir una base de datos
        open file "Macintosh HD:Carpeta FileMaker Pro:Batters" with password "admin" -- abrir una base de datos con la clave "admin"
        open {"Macintosh HD:Carpeta FileMaker Pro:Lista CD.fp5", "Macintosh HD:Carpeta FileMaker Pro:Músicos.fp5"} ¬
                with password {"admin"} -- abrir bases de datos con la misma clave
        open {"Macintosh HD:Carpeta FileMaker Pro:Batters", "Macintosh HD:Carpeta FileMaker Pro:Músicos.fp5"} ¬
                with password {"admin", "usuario"} -- arir bases de datos con clave distinta
        open file "Macintosh HD:Carpeta FileMaker Pro:Lista CD.fp5" with password "" -- abrir base de datos con la clave en blanco
end tell
```

Open Application kAERequiredSuite kAEOpenApplication
¿Qué se puede hacer?
Use el evento Open Application para lanzar FileMaker desde el escritorio. También le podremos utilizar para abrir cualquier otro programa que soporte el evento Open Application.

En tanto que se supone que el escritorio es el único que puede abrir otras aplicaciones, FileMaker responderá a este evento sólo cuando se envíe desde el Finder. Si se quiere abrir otras aplicaciones desde FileMaker se deberá utilizar la sintaxis de la aplicación en vez de open application. Véase los ejemplos a continuación.

La mayoría de los usuarios de AppleSript no necesitarán utilizar este evento.

Para usar el evento Open Application se necesita especificar un objeto, como "HD:Applications Folder:MacWrite Pro".

Ejemplos

```
tell application "Finder"
        launch application "FileMaker Pro"
end tell

tell application "FileMaker Pro"
        launch application "ClarisWorks"
end tell
```

Open Documents kAERequiredSuite kAEOpenDocuments
¿Qué se puede hacer?
Use el comando Open Documents para abrir archivos necesarios para obtener datos.

Se debe especificar la ruta del archivo.

Hay algunas diferencias técnicas entre el evento Open Document y el evento Open. Véase la presentación de objetos de Filemaker para más información.

Normalmente, la mejor forma de abrir archivos es utilizando el evento Open Document.

Nota: Se debe utilizar la sintaxis Open File desde desde Applescript cuando se quiera abrir un archivo. Cuando se abran muchos archivos, se tiene que utilizar sólamente el evento Open. Ver ejemplos a continuación.

Nota: No se puede usar el evento Open para abrir una base de datos de Filemaker al ejecutarse desde el paso de guión Ejecutar AppleScript de ScriptMaker. Use el paso de guión Abrir Archivo para abrir todas las bases de datos necesarias para ejecutar el AppleScript antes de su ejecución.

Ejemplos

```
tell application "FileMaker Pro"
 open file "HD:Músicos"
 open file "HD:Compact Discs" with password "laClave"
end tell
```

Print (archivos) kAERequiredSuite kAEPrintDocuments
¿Qué se puede hacer?
Use el evento Print para imprimir archivos, de la misma forma que seleccionando la opción Imprimir desde el menu Archivo.

Se puede utilizar el evento Print para imprimir archivos que no están abiertos. No obstante, los archivos se cerrarán una vez que se haya completado el evento. Los archivos que ya estaban abiertos antes de enviar el evento Print permanecerán abiertos.

El evento Print es parte del conjunto de comandos requeridos, y es una extensión del evento Print que se encuentra en el conjunto del núcleo (Core Suite).

El evento Print funcionará con los siguientes objetos:
Windows
Documents
Databases

Para utilizar el evento Print se debe dar un alias al objeto a imprimir. Un alias es como una especificación del objeto, por ejemplo "HD:Documents:Compact Discs".

Nota: Es mejor no usar el evento Print cuando se pueda generar y ejecutar un guión de ScriptMaker, en tanto que el diálogo de imprimir no se puede cancelar cuando se usa el evento Print. Además, el uso de un guión de ScriptMaker hará que el archivo se mantenga abierto si se necesita.

Nota: El evento Print devolverá un error si se ejecuta desde el paso de guión Ejecutar AppleScript. Para generar un informe por impresora se utilizará el paso de guión Imprimir

Ejemplos

```
tell application "FileMaker Pro"
        print alias "Classic:Músicos.fp5" -- Imprimimos un archivo dada su ubicación en el Finder
end tell
```

Print (objects) kAECoreSuite kAEPrint
¿Qué se puede hacer?
Use el evento Print para imprimir un archivo de FileMaker.

La sintaxis del evento Print es muy sencilla de utilizar. Todo lo que se necesita es dar un alias. Print es equivalente a hacer un "Imprimir" desde el escritorio, por lo que el archivo se cerrará después que se haya terminado de imprimir.

Algunas alternativas al uso del evento Print son:
• Crear un guión de ScriptMaker en la base de datos correspondiente y usar los eventos de Apple para enviar un Do Script para ejecutar este guión.
• Enviar el evento Do Menu a FileMaker; por ejemplo, do menu menu item "print…" of menu "file".
• Usar el evento Print desde el conjunto requerido, utilizando los Apple events alias. Véase los datos acerca del evento Print del conjunto requerido.

Nota: El evento Print nos dará un error si se ejecuta desde el paso de guión Ejecutar AppleScript de ScriptMaker. Use el paso de guión Print de ScriptMaker para imprimir el informe.

Ejemplos

```
tell application "FileMaker Pro"
        print alias "Macintosh HD:Músicos"
end tell
```

Quit kAERequiredSuite kAEQuitApplication
¿Qué se puede hacer?
El evento Quit hace que se cierre la aplicación FileMaker, de igual forma que si se selecciona la opción Salir del menu archivo. Se puede utilizar el evento Quit para volver al Finder, por ejemplo, si una serie de scripts se ha terminado de ejecutar, y se desea presentar un escritorio limpio.

El hecho de enviar el evento Quit a FileMaker hará que se cierre el programa. No se necesita pasar más información.

Ejemplos

```
tell application "FileMaker Pro"
        quit
end tell
```

Save kAECoreSuite kAESave
¿Qué se puede hacer?
Use el evento Save para guardar un archivo de FileMaker. Esto no es normalmente necesario en tanto que FileMaker guarda los archivos automáticamente. El evento guardar se puede utilizar para salir de un registro o petición, y por tanto sin dejar campos activos y actualizando la información en el registro.

En tanto que FileMaker guarda la información periodicamente cada pocos segundos, no tendría que ser necesario el uso de este evento. No obstante, se puede configurar FileMaker para guardar los archivos con menos frecuencia (para evitar el gasto en exceso de la batería, por ejemplo, en un ordenador portátil). No obstante, tomará sentido hacer una petición a FileMaker para que escriba los cambios al disco periódicamente.

Use el evento Save cuando se acceda a datos que puede que no estén guardado en el disco todavía. Por ejemplo, use Save antes de obtener o pasar datos a un campo cuando se haya introducido información en un campo y el cursor sigue activo. Otro caso donde Save puede ser útil es poniendo datos en un campo u obteniendo las opciones del campo cuando tenemos una lista de valores basada en los conternidos del campo.

Ejemplos

```
tell application "FileMaker Pro"
        set cell "Preferencias" of record 5 to "Bueno"
        save database 1
        get choices of cell "Preferencias"
end tell
```

Set Data kAECoreSuite kAESetData
¿Qué se puede hacer?
Use el evento Set Data para incluir datos en un objeto, como un campo, celda, registro o petición.

Cuando se utilice el evento Set Data se debe incluir una especificación del objeto, como record 1, y los datos que se quieran incluir en el ojeto.

```
tell application "FileMaker Pro"
        set record 1 to {"Peter I. Tchaikovski", "Bueno"}
end tell
```

El evento Set Data se puede usar para fijar tipos de datos como dibujos, sonidos, y películas QuickTime en campos de contenedor de FileMaker. También es posible utilizar Apple Events para fijar una referencia a un archivo de dibujo en un campo contenedor -por ejemplo set cell 1 to file "Macintosh HD: Mi Dibujo". FileMaker trata la imagen como si se hubiera importado utilizando la opción "Guardar una referencia al archivo".

NOTA: la referencia al dibujo tiene que ser del tipo "file".

Nota: Se debe tener acceso de escritura a fin de establecer datos. Si se intentara establecer datos en un objeto de sólo lectura, FileMaker fallará el evento, y devolverá el error errAEWriteDenied (-10006).

Nota: No se puede establecer el objeto actual a otro objeto cuando se ejecute el AppleScript utilizando el paso de guión Ejecutar AppleScript. El ejemplo siguiente dará un error:

set current record to record 1

En vez de esto, use el evento Go To.

go to record 1

Ejemplos

Se puede decir: Si se quiere fijar datos en:

```
tell application "FileMaker Pro"
        set record 1 to {"Manuel de Falla", "Muy Bueno"} -- un registro con al menos dos campos
        set cell "Canciones" of ¬
                last record to "El Sombrero de Tres Picos" -- un campo en el último registro
        set cell "Canciones" of ¬
                record after record 1 to "El Sombrero de Tres Picos" -- un campo en el segundo registro
        set checked of menu item ¬
                "menu 2" of menu "External" to true -- un elemento de menu definido por el usuario
        set data last record to ¬
                "a\tb\tc" -- introducimos datos en los tres primeros campos del último registro(según el orden de tabulación)
        set data cell 1 of ¬
                middle record to "Malo" -- el registro central
        set data field 1 of window 1 to "Manuel de Falla" -- un campo de un registro de una ventana
        set data (every record of window ¬
                "Músicos" whose cell 1 contains ¬
                "a" and cell 2 = "b") to "c" -- registros que cumplan una cierta condición
        set data name of menu item 1 of ¬
                menu "External" to "NuevoNombre" --nombre de un elemento de menú definido por el usuario
        set data enabled of menu item 1 of ¬
                menu "External" without to -- la propiedad habilitado de un elemento de menu
        set data visible of window 1 with to -- la propiedad visible de una ventana
end tell
```

(Otros operadores que se pueden utilizar con las cláusulas "whose" son:<, ≤, >, ≥, ≠, AND, OR, NOT, begins with, ends with, contains.)

Show kAEMiscStandards kAEMakeObjectVisible
¿Qué se puede hacer?

Use el evento Show para mostrar una presentación, o grupo de registros. Cuando se esté operando con varios registros y se utilice la cláusula "**whose**", Show es muy parecido a una petición de búsqueda de FileMaker.

El evento **Show** puede funcionar con los siguientes objetos: **Window**, **Document**, **Database**, **Record**, **Request**, y **Layout**. El objeto deberá de estar abierto necesariamente o disponible; así, puede que sea necesario abrir el objeto **window**, **document** o **database** antes de mostrar algo en estos.

Para usar el evento Show, hay que especificar el objeto a mostrar, como **records whose cell "Preferencias" ≤ "Bueno"**, o **every record**, o **layout 2.**

El uso de **Show** con el objeto **Database** selecciona registros de la base de datos entera. De todas formas, un evento **Show** dirigido a **Document**, devuelve registros del conjunto hallado actualmente. Al utilizar el objeto **Document**, se puede empezar con un rango de búsqueda y así afinar el conjunto encontrado con los eventos **Show** subsecuentes.

Nota: al usar el evento **Show** en una ventana, base de datos o documento hace que el objeto sea visible si este estaba oculto pero no lo activa (ie. no lo trae al frente). En este caso, use el evento **Go To** para activarlo y mostrar una base de datos.

En tanto que el evento **Show** utiliza la terminología AppleScript para buscar datos, no admite la simbología de FileMaker para realizar búsquedas como los relativos a la fecha actual o a los registros duplicados (ej.: //, !). Para utilizar este tipo de símbolos habrá que crear una búsqueda e incluir este símbolo en una celda.
A continuación se ofrece una tabla mostrando la terminología AppleScript y los símbolos correspondientes de FileMaker para realizar búsquedas.

Terminología equivalente entre FileMaker y AppleScript
AppleScript FileMaker
contains texto
begins with texto*
ends with *texto
equals = (coincidencia exacta)
greater than >
less than <
greater than or equal to ≥
lesser than or equal to ≤
not equal to Petición omitida
or (Multiples peticiones)
and Datos en múltiples campos dentro de una petición de búsqueda
greater than and less than ...
No soportado via Show son: Duplicados (Duplicate), Fecha Actual (Current Date), Un sólo carácter (Single character), Fecha actual (Today's date) y fecha u hora inválidas (Invalid date or time)

Ejemplos

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        show record 1 -- obtener el primer registro de un conjunto de registros
        show window "Lista CD" -- hacer visible una ventana oculta
        show layout "Lista Compositores" -- mostrar una presentación distinta
```

```
        show every record of database 1 -- buscar todos los registros
        show (every record of database 1 ¬
                whose cell "Canciones" contains ¬
                "Picos") -- buscar un grupo de registros
        show (every record of document 1 ¬
                whose cell "Compositores" contains ¬
                "de") -- buscar registros dentro del grupo de registros actual
        show (every record of database 1 ¬
                whose cell "Preferencias" = "") -- buscar registros que están vacíos
        show (every record of database 1 ¬
                whose cell "Canciones" ≠ "") -- buscar registros que no estén vacíos
end tell
```

(Otros operadores que se pueden utilizar son: <, ≤, >, ≥, ≠, AND, OR, NOT, Begins with, Ends with, contains.)

Sort kAEDatabaseSuite kAESort
¿Qué se puede hacer?
Use el evento Sort para reordenar el conjunto actual de registros encontrados en un archivo de FileMaker.

El evento Sort tiene tres componentes:

• La presentación a ordenar, que es la presentación donde se muestranlos resultados de la ordenación. Si no se especifica ninguna presentación, se utilizará la actual.
• El campo por el que se ordena la presentación. Si no se indica ningún campo, FileMaker desordenará la base de datos.
• La dirección de la ordenación - ascendente (ascending), descendente (descending), o personalizada (custom). Si no se indica este parámetro, FileMaker lo ordenará de forma ascendente. Para ordenar un campo de forma personalizada hay que usar la lista de valores asociada al campo en la presentación especificada (o la presentación actual si no se especifica ninguna).

Se pueden especificar más de un campo para hacer una ordenación multi-nivel.

Ejemplos

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
 sort by field "Compositores" -- ordenar por un campo
 sort layout "Lista CD" ¬
  by field "Canciones" in order descending -- ordenar por un campo en orden descendiente
 sort layout 1 by ¬
  {field "Preferencias", field "Duración"} ¬
   in order {descending, ascending} -- ordenar por dos campos, con diferente sentido de ordenación
 sort window 1 -- desordenar una base de datos
end tell
```

OBJETOS

Application cApplication
¿Qué se puede hacer?
El objeto Application no es uno de los que normalmente se utilice por los usuarios, excepto cuando se inicialice una serie de comandos. Por ejemplo, se puede decir:

```
tell application "FileMaker Pro"
 create new record with data {"Beethoven", "Bueno"}
end tell
```

La sintaxis anterior dirige las peticiones a la aplicación FileMaker.

A continuación hay algunos ejemplos de Apple Events que se aplican directamente a Application, con algunos ejemplos de sintaxis:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
 exists application "FileMaker Pro" -- comprueba que FileMaker se esté ejecutando
 class info of application -- obtener la información class de la aplicación
 quit -- sale de FileMaker
end tell
```

¿Qué propiedades tiene?

Best Type El Best Type del objeto Application es 'obj ', significando objecto.
Class. El Class del objeto Application es 'capp'.
Default Type. El Default Type del ojeto Application es 'obj ', el mismo que Best Type.
Frontmost Frontmost es un valor lógico que nos indica si la aplicación es el proceso que está más por delante o no.
Name Devuelve el nombre de la aplicación tal como aparece en el escritorio. Este es el nombre que se debe utilizar en una sentencia Tell.
Version Devuelve la version de la aplicación que está en el recurso 'vers' - ej.: "3.0v1".

Ejemplos

El principal uso que los usuarios le darán al objeto Application es la sentencia Tell Por ejemplo:

```
tell application "FileMaker Pro"
 -- <incluir las sentencias>
end tell
```

El siguiente script muestra cómo se puede acceder o usar cada una de las propiedades de **Application.**

```
tell application "FileMaker Pro"
 copy name of application "FileMaker Pro" to appName
 copy version of application "FileMaker Pro" to appVersion
```

```
copy best type of application "FileMaker Pro" to appBestType
if frontmost of application "FileMaker Pro" is true then beep 5
copy default type of application "FileMaker Pro" to appDefaultType
copy class of application "FileMaker Pro" to appClass
end tell
```

Cell cCell
¿Qué se puede hacer?
Un objeto Cell es la intersección de un registro y un campo; es el valor de un campo en particular para un registro en concreto. Use el objeto Cell para obtener o manejar información.

Las celdas relacionadas son accesibles via Apple events. El nombre de una celda relacionada incluye el nombre de la relación separado por dos veces el signo de dos puntos como en "Canciones::Duración". De forma similar, el ID de una celda relacionada incluye un identificador para la relación. Para acceder a una celda relacionada esta debe estar en una presentación del archivo maestro. La presentación 0 sólo incluye todas las celdas especificadas en la base de datos local (las celdas relacionadas no pueden ser accesibles desde esta presentación).

A continuación hay unos ejemplos que son aplicables comúnmente a las celdas, con sintaxis de ejemplo:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
copy cell 1 to aVariable -- Copiar el valor de una celda a una variable
count cells of database 1 -- Cuenta el número de celdas deuna base de datos
exists cell 5 -- Comprueba que exista una base de datos
class info cell -- Obtener la información Class Info de una celda
get data cell 1 -- Otener información de una celda
data size cell 1 -- Obtener el tamaño de la información contenida en una celda
show (records whose cell "Compositores" is ¬
"Beethoven") -- Buscar registros basados en el valor de una celda
set cell 1 to "Bueno" -- Establecer información en una celda
end tell
```

¿Qué propiedades tiene?

Best Type El Best Type de una celda es 'ccel'.
Class El Class de un objeto celda es 'ccel'.
Default Type El Default Type es 'TEXT' para campos de texto, 'doub' para campos numéricos, 'PICT' para campos contenedores, y 'ldt ' para campos de fecha y hora.
Name. Devuelve el nombre de campo de una celda tal como aparece en el diálogo Definir Campos. Si la celda es relacionada, el nombre de la celda incluye el nombre del campo relacionado separaro por el signo de dos puntos (::) como en "Canciones::Duración".
ID La propiedad de ID de una celda es un identificador único y persistente representado por un registro y el ID del campo (ej.: {1,2}). Cuando una celda es relacionada, el ID toma la forma de {registro ID, {relación ID, campo ID}} donde el ID del registro es el del archivo maestro, y el ID

del campo es del archivo relacionado. Por ejemplo, el ID de la celda {1, {2, 3}} es el identificador único de una celda relacionada del registro en una base de datos local cuyo ID es 1, utilizado el campo de la base de datos relacionada cuyo ID es 3 como se ha definido por la relación cuyo ID es 2.
CellValue La propiedad CellValue devuelve el contenido de una celda.
Access. La propiedad Access devuelve los privilegios de acceso de la celda, que puede ser uno o más de los siguientes: Sin acceso (No Access), Lectura (Read), Escritura (Write), Actualizar (Update), Crear (Create), No Borrar (No Delete), Borrar (Delete), Total (Full).
Protection La propiedad protection devuelve el estado de protección de la celda, que es una o más de las siguientes: Sólo Lectura (Read Only), Protegido por fórmulas (Formulas Protected), Lectura (Read), Escritura (Write).
Formula La propiedad Formula devuelve la fórmula utilizada para calcular su valor. Esta propiedad puede estar vacía ("") para los campos que no están definidos como campos de cálculo.
Lock La propiedad lock devuelve el estado de protección de una celda, que puede ser: Desbloqueada (Unlocked), Bloqueo compartido (Shared Lock), Bloqueo exclusivo (Exclusive Lock).
Repeat Size La propiedad Repeat Size devuelve un número que indica el número de repeticiones que se han definido para mostrarse. Los campos no repetitivos tienen el valor de 1; los campos repetitivos devuelven el tamaño definido por el campo en una presentación determinada. Al especificar el tamaño de las repeticiones de una celda contenida en la presentación 0 devolverá el número máximo de repeticiones definidas en el diálogo de Definir Campos.
Choices La propiedad Choices contiene la lista de valores asociada con la celda en una presentación determinada.
GlobalValue. La propiedad GlobalValue es una propiedad lógica que devuelve verdadero (true) si la celda está definida en el diálogo Definir Campos para que sea un campo global.

Ejemplos:

```
tell application "FileMaker Pro"
 cell 1
end tell
```

Este comando toma toda la información de la primera celda de la base de datos que está más al frente. Para obtener las propiedades de una celda usando AppleScript:

```
tell application "FileMaker Pro"
 best type of cell 1
 default type of cell 1
 class of cell 1
 cellValue of cell 1
 name of cell 1
 formula of cell 1
 ID of cell 1
 lock of cell 1
 protection of cell 1
 repeat size of cell 1
 choices of cell 1
 globalValue of cell 1
```

end tell

Database cDatabase
¿Qué se puede hacer?
Use el objeto Database para distinguir entre bases de datos distintas, en caso que se tenga abierta más de una.

En tanto que el objeto Database es un archivo de FileMaker considerado sólo como una colección de campos y registros, seleccione Database en vez de Document cuando se quiera que FileMaker ignore temporalmente el estado del documento, como el conjunto de registros encontrado, cuando se evalúe nuestra petición.

Por ejemplo, si se pide el registro 1 de la base de datos 1, se obtendrá el primer registro de la base de datos indistintamente al conjunto de registros encontrados. Si se pide el registro 1 del documento 1, se obtendrá el primer registro del conjunto de registros encontrado actualmente, si lo hubiera.

Cuando se usan las propiedades del registro y la presentación actual no es necesario especificar el objeto Database. Por ejemplo, un guión como **tell application "FileMaker Pro" to get current record** nos devolverá una referencia al registro actual de la base de datos que esté más al frente.

A continuación hay algunos Apple events que se aplican cumúnmente a bases de datos, con ejemplos de sintaxis:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        close database 1 -- cerrar una base de datos
        count records of database 1 -- contar los registros de una base de datos
        exists database "Músicos" -- ver si existe una base de datos (si está abierta)
        class info database -- obtener el Class de una base de datos
        get data database 1 -- obtener datos de una base de datos
        data size database 1 -- obtener el tamaños de los Datos de una base de datos
        open file "Macintosh HD:Músicos" -- abrir una base de datos
        set current record to record 5 -- cambiar el registro actual
        set current layout to layout 2 -- cambiar la presentación actual
        set multiuser of database 1 to true -- activar el uso de la base de datos a Multiusuario
end tell
```

¿Qué propiedades tiene?

Best Type El Best Type del objeto Database es sencillamente 'obj ', significanto objecto.
Class La Class del objeto Database es 'cDB ', significando Base de datos.
Default Type La Default Type también es'obj ', al igual que la Best Type.
Name Devuelve el nombre de la base de datos tal como aparece en el escritorio. Este es el nombre que se debe usar en una sentencia Tell desde AppleScript.
Lock La propiedad Lock property devuelve el estado actual de la base de datos: Desbloqueado (Unlocked), Bloqueado compartido (Shared Lock), Bloqueo Exclusivo (Exclusive Lock).
Current Layout La propiedad Current Layout devuelve una referencia a la presentación actual.
Current Record La propiedad Current Record devuelve una referencia al registro actual.

Access La propiedad Access devuelve los privilegios de acceso de la base de datos, que puede ser uno o más de la lista siguiente: Sin Acceso (No Access), Lectura (Read), Escritura (Write), Actualización (Update), Creacción (Create), No Borrar (No Delete), Borrar (Delete), Todos (Full). MultiUser El estado de compartición actual de la base de datos. Se puede utilizar para establecer el estado de compartición de la base de datos en multi-usuario o único usuario.

Ejemplos:

El siguiente guión devolverá la información de la primera celda de la base de datos 2.

```
tell application "FileMaker Pro"
 cell 1 of database 2
end tell
```

Para solicitar las propiedades de la base de datos se puede usar lo siguiente:

```
tell application "FileMaker Pro"
 best type of database 1
 class of database 1
 default type of database 1
 name of database 1
 lock of database 1
 current layout of database 1
 current record of database 1
 access of database 1
 multiuser of database 1
end tell
```

Para obtener información del registro actual se puede usar las siguientes instrucciones:

```
tell application "FileMaker Pro"
 every cell of current record
end tell
```

Document cDocument
¿Qué se puede hacer?
Use el objeto **Document** para distinguir entre bases de datos diferentes, en caso que se tenga abierta más de una. Seleccione **Document** en vez de **Database** cuando quiera que FileMaker use el conjunto de registros encontrados para evaluar una petición. **Document** y **Window** son sinónimos.

Al trabajar con objetos **Document**, FileMaker se centrará en cosas que no sean relativas a los datos actuales del archivo, como el conjunto de registros encontrado, al contrario que ocurre cuando se trabaja con objetos **Database**.

Por ejemplo, si se pide el registro 1 de la base de datos 1, se obtendrá el primer registro de esta indistintamente a los registros encontrados. Si se pide el registtro 1 del documento 1, se obtendrá el primer registro de los registros encontrados, si se hubiera realizado una búsqueda.

A continuación hay unos Apple events que se aplican comúnmente a los documentos, con ejemplos de sintaxis:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
 close document 1 -- cerrar un documento
 count document -- contar documentos
 exists document "Músicos" -- comprueba si está abierto un documento
 class info document -- obtener el parámetro Class del documento
 get data document 2 -- obtener datos de un documento
 data size document 1 -- obtener el tamaño de la información de un documento
 open file "HD:Músicos" -- abrir un documento
 print document "Addresses" -- imprimir un documento
end tell
```

¿Qué propiedades tiene?

Best Type El Best Type del objeto Document es 'obj '.
Class. El Class de un objeto Documento es 'docu'.
Default Type El Default Type es también 'obj ', el mismo que Best Type.
Name Devuelve el nombre del documento tal como aparece en el escritorio. Este es el nombre que se debe usar en una instrucción Tell desde AppleScript.
Modified La propiedad modified devuelve verdad (true) si se ha editado un documento en la sesión actual.

Ejemplos:

```
tell application "FileMaker Pro"
 tell document 1
  --[instrucciones]
 end tell
end tell

tell document "Músicos" of application "FileMaker Pro"
 if modified is true then
  copy cell 1 to MiVariable
 end if
end tell
```

Para solicitar las propiedades del documento se puede hacer de la siguiente forma:

```
tell application "FileMaker Pro"
 best type of document 1
 default type of document 1
 class of document 1
 name of document 1
```

modified of document 1
end tell

Field cColumn
¿Qué se puede hacer?
Use el objeto **Field** para manipular información en todos los registros o para especificar una parte de información de un registro.

Los objetos **Field** son denominados a veces **columns**, y son análogos a las columnas de una hoja de cálculo. Un campo de un registro en particular es una celda (**cell**). Normalmente, se querrá establecer y obtener datos de una celda, y no de una columna.

Los campos relacionados son accesibles también via Apple Events. El nombre de un campo relacionado incluye el nombre de la relación separada por el signo de dos puntos dos veces (::). De forma similar, el ID de un campo relacionado incluye un identificador de la relación. Para acceder a un campo relacionado este debe estar en una presentación en el archivo maestro. Dado que la presentación 0 incluye todos los campos especificados en la base de datos local, los campos relacionados no son accesibles desde esta presentación.

Nota: Es importante observar que los campos no están contenidos por los registros. Así si se dijera a FileMaker set field 1 to MiVariable, se cambiará esa celda en todos los registros del conjunto actual se modificarán.

A continuación hay algunos Apple events que se aplican comúnmente a los registros, con sintaxis de ejemplo:

Se puede decir: Si se quiere:
tell application "FileMaker"
count field 1 class cell -- contar celdas mostradas de un campo (registros mostrados)
exists field "Compositores" -- comprobar si existe un campo
class info field -- obtener la propiedad class de un campo
field 1 -- obtener datos del campo 1 en los registros mostrados
field 1 as text -- obtener datos del campo 1 de los registros mostrados a modo texto (el resultado será delimitado por tabulador)
data size field 1 -- obtener el tamaño del campo 1 de los registros mostrados
set field 1 to "Medio" -- estalecer el valor del campo 1 (se hace en todos los registros del conjunto actual)
sort by field "Compositores" -- ordenar por un campo
end tell

¿Qué propiedades tiene?

Best Type El Best Type de un objeto Field es 'ccol', significando columna.
Class El Class de un objeto Field es también 'ccol'.
Default Type El Default Type es 'TEXT' para campos de texto, 'doub' para campos numéricos, 'PICT' para campos contenedores, y 'ldt ' para campos de fecha y hora.
Name Devuelve el nombre de un campo tal como aparece en el diálogo de Definir Campos. Si se trata de un campo relacionado, el nombre del campo incluirá el nombre del archivo relacio-

nado separado por e signo de dos puntos dos veces (::).
ID La propiedad ID de un campo es un identificador único y persistente de un campo otorgado en base al orden de creación. Cuando ae trata de un campo relacionado, el ID es una lista con dos valores representando el ID de la relación y el ID del campo en la base de datos relacionada.
Access La propiedad Access devuelva los privilegios de acceso del campo, que puede ser una o má s de las siguientes: Sin Acceso (No Access), Lectura (Read), Escritura (Write), Modificación (Update), Creación (Create), No Borrar (No Delete), Borrar (Delete), Todos (Full).
Protection La propiedad protection devuelve el estado de protección del campo, que puede ser uno o más de los siguientes: Sólo Lectura (Read Only), Protegido por fórmulas (Formulas Protected), Lectura (Read), Escritura (Write).
Lock La propiedad lock devuelve el estado de bloqueo del campo, que puede ser uno de los siguientes: Desbloqueado (Unlocked), Bloqueo compartido (Shared Lock), Bloqueo exclusivo (Exclusive Lock).
Nulls OK Nulls OK es una propiedad lógica que devuelve verdad (true) si el campo está definido, en el diálogo de Definir Campos, para permitir valores vacíos, y falso (false) si el campo debe contener un valor.
Repeats Repeats es una constante que devuelve 'rFxd' si el campo ha sido definido como un campo repetitivo en el diálogo de Definir Campos, y 'rSgl' en caso contrario.
Repeat Size Repeat Size devuelve un valor que indica el número de repeticiones de un campo en una presentación. Los campos no repetitivos tienen un valor de 1; los campos repetitivos nos darán el tamaño máximo definido para un campo en una presentación en concreto. Para determinar el número máximo de repeticiones para un campo como se especifica en el diálogo para Definir Campos, se tiene que especificar la presentación 0.
Unique Value Unique Value es una propiedad lógica que devuelve verdad (true) si se ha definido el campo, en el diálogo para Definir Campos,para requerir un único valor para cada registro, y falso (false) si no es así.
Formula La propiedad Formula devuelve la fórmula de cálculo del campo. Esta propiedad estará vacía ("") en campos que no est definidos como campos de cálculo.
Choices La propiedad Choices contiene la lista de valores asociada a un campo en una presentación en concreto.
GlobalValue La propiedad GlobalValue es una propiedad lógica que devuelve verdad (true) si el campo se ha definido, en el diálogo para Definir Campos, como un campo global.

Ejemplos:

```
tell application "FileMaker"
        best type of field 1
        class of field 1
        default type of field 1
        name of field 1
        ID of field 1
        access of field 1
        protection of field 1
        lock of field 1
        nulls OK of field 1
        repeats of field 1
        repeat size of field 1
```

```
        unique value of field 1
        formula of field 1
        choices of field 1
        globalValue of field 1
end tell
```

FileMaker Script cFileMakerScript
¿Qué se puede hacer?
Use el objeto FileMaker script para decir a FileMaker que ejecute un guión de ScriptMaker creado en un archivo de FileMaker. Por ejemplo, se puede decir:

```
tell application "FileMaker Pro"
 do script FileMaker script "Archiva"
end tell
```

A continuación hay algunos Apple Events que se aplican comúnmente a FileMaker Scripts, consintaxis de ejemplo.

Se puede decir: Si se quiere:

```
tell application "FileMaker"
        count database 1 ¬
                class FileMaker script -- contar quiones de ScriptMaker en una base de datos
        exists FileMaker script "Mantén e Imprime" -- comprobar si existe un guión
        class info FileMaker script -- obtener la propiedad class de un guión
        data size FileMaker script 1 -- obtener el tamaño de un guión
end tell
```

¿Qué propiedades tiene?

Best Type El Best Type de un guión de ScriptMaker es 'obj '.
Class La propiedad Class de un objeto guión es 'cSCP'.
Default Type La Default Type es también 'obj ', el mismo que Best Type.
Name devuelve el nombre de un guión tal como aparece en la pantalla.
ID El ID es un número único que identifica el guión.

Ejemplos

A continuación hay un ejemplo de uso del objeto FileMaker Script

```
tell window 1 of application "FileMaker Pro"
        do script "Imprimir Lista CD Favoritos"
end tell
```

Para solicitar las propiedades de FileMaker Script se puede hacer de la siguiente forma:

```
tell window 1 of application "FileMaker Pro"
        best type of FileMaker script 1
```

```
        class of FileMaker script 1
        default type of FileMaker script 1
        name of FileMaker script 1
        ID of FileMaker script 1
end tell
```

Layout cTable
¿Qué se puede hacer?
Use el objeto **Layout** para moverse entre presentaciones, o restringir los campos disponibles.

Los objetos **layout** se llaman también tablas (**tables**) Cuando se trabajan con presentaciones, FileMaker considera sólo los campos que están en esa presentación.

Siempre se dispone de una presentación interna que contiene todos los campos definidos en la base de datos. Esta presentación se llama **layout 0**, y se puede usar cuando no se esté seguro dónde están los campos con los que se va a trabajar. Los datos se pueden obtener de la presentación 0 en el mismo orden que se crearon a través del diálogo Definir Campos. Nótese que la presentación 0 no contiene información específica de presentaciones como listas de valores asociadas a un campo ni campos relacionados.

A continuación hay algunos Apple events que se aplican comúnmmente a presentaciones, con ejemplos de sintaxis:

Se puede decir: Si se quiere:

```
tell window 1 of application "FileMaker Pro"
        count database 1 class layout -- contar presentaciones en una base de datos
        exists layout "Lista CD" -- comprobar si existe una presentación
        class info layout -- obtener la propiedad class de una presentación
        get data layout 1 -- obtener información de una presentación
        data size layout 1 -- obtener el tamaño de la información de una presentación
        show layout 1 -- cambiar a una presentación
        set field 1 of layout 2 to "Bueno" -- establecer datos en una presentación
        sort layout 2 by field ¬
                "b" in order descending -- ordenar en una presentación
end tell
```

¿Qué propiedades tiene?

Best Type La Best Type de un objeto presentación es 'ctbl', significando tabla, dado que una presentación en FileMaker es realmente una tabla.
Class La Class de una presentación es 'ctbl'.
Default Type La Default Type es 'TEXT'.
Name Devuelve el nombre de una presentación tal como aparece en FileMaker.
ID La ID es un número único y persistente para una presentación.
Access La propiedad Access devuelve los privilegios de acceso de la presentación, que puede ser una o más de las siguientes: Sin Acceso (No Access), Lectura (Read), Escritura (Write),

Modificación (Update), Creacción (Create), No Borrar (No Delete), Borrar (Delete), Todos (Full).
Protection La propiedad protection devuelve el estado de protección de la presentación, que es una o más de las siguientes: Sólo Lectura (Read Only), Protegido por fórmulas (Formulas Protected), Lectura (Read), Escritura (Write).
Lock La propiedad lock devuelve el estado de bloqueo de la presentación, que es una de las siguientes: Desbloqueada (Unlocked), Bloqueo compartido (Shared Lock), Bloqueo Exclusivo (Exclusive Lock).
Kind La propiedad kind devuelve el tipo de presentación, que puede ser o Tabla (**Table**) o Presentación (**View**). Layout 0 es una tabla (**Table**), en tanto que es la presentación real de una base de datos (una base de datos de FileMaker tiene sólo una tabla), y el resto de las presentaciones tiene la propiedad kind de presentación (**Layout**).
Visible Visible es un valor lógico que indica si la presentación es visible o no en el menu de presentaciones.

Ejemplos:

Ejemplo usando el objeto Layout:

```
tell application "FileMaker Pro"
        show layout "Lista CD"
end tell
```

Para obtener las propiedades de una presentación se pueden usar las siguientes instrucciones:

```
tell application "FileMaker Pro"
        best type of layout 1
        class of layout 1
        default type of layout 1
        name of layout 1
        ID of layout 1
        access of layout 1
        protection of layout 1
        lock of layout 1
        kind of layout 1
end tell
```

Menu cMenu
¿Qué se puede hacer?
Use el objeto Menu para acceder a elemetos de menu que ejecuten comandos desde menus de FileMaker. Por ejemplo, se puede decir:

```
tell application "FileMaker Pro"
        do menu menu item 5 of menu "Guiones"
end tell
```

Este guión dice a FileMaker que ejecute el quinto elemento bajo el menu Guiones, que es un guión de FileMaker creado por el desarrollador de la base de datos. Recuerde contar los sepa-

radores de los menus como elementos de menu cuando nos refiramos a elementos de menu por su número índice.

También, los usuarios tienen la posibilidad de crear menus en FileMaker. Un menu creado por el usuario se coloca automáticamente en un submenú llamado "External", bajo el menu Guiones. Los meus y los elementos de menu pueden ser anidados en el menu External. Véase un ejemplo en la sección de ejemplos AppleScript.

Nota: FileMaker tiene dos modos (Mode) de menus. El modo menu utilizado en Visualizar, Buscar, y Vista Previa tiene un ID de 4 mientras que el modo menu utilizado en Presentación tiene un ID de 22.

Nota: En tanto que todos los elementos de menu están deshabilitados mientras que se ejecuta un guión, no se puede usar el evento **Do Menu** para seleccionar elementos de menu mientras se ejecute un guión usando el paso de guión Ejecutar AppleScript. Además, no se puede usar el estado de control de un elemento de menu para determinar el modo actual de la aplicación.

A continuación hay algunos Apple events que se usan comúnmente con los menus:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        create new menu ¬
                with properties "MiPrimerMenu" -- crear un nuevo menu
        delete menu "MiPrimerMenu" -- borrar un menu
        exists menu "MiPrimerMenu" -- comprobar que exista un menu
        class info menu -- obtener la propiedad Class de un menu
        set enabled of menu 1 ¬
                of menu "External" to false -- hacer que un menu esté inactivo
end tell
```

Nota: El menu Apple es el menu 1. Menus jerárquicos, como el menu de estilo, son ambos menus y elementos de menu.

¿Qué propiedades tiene?

Best Type. La Best Type del objeto menu es 'obj'.
Class La Class del objeto Menu object es 'cmnu'.
Default Type La Default Type es 'obj '.
Name Devuelve el nombre del menu tal como aparece en la pantalla. Este es el nombre que se debe utilizar en una instrucción Tell desde AppleScript.
ID La ID del menu es un número (integral pequeño) que devuelve el ID de recursos del menu.
Enabled Enabled es una propiedad lógica que devuelve verdad si el menu está actualmente habilitado (accesible), y falso si el menú está deshabilitado.

Ejemplos

Para acceder a un menu, se puede decir:

```
tell application "FileMaker Pro"
```

```
        do menu menu item "Buscar" of menu "Modo"
 end tell
```

Para crear un menu definido por el usuario, se puede decir:

```
tell application "FileMaker Pro"
        create new menu with properties "Mi Menu"
        create new menu with properties "Mi Menu2"
        create new menu with properties {name:"Mi Menu3", ID:123, enabled:false}
end tell
```

Para obtener las propiedades de un menu se puede utilizar lo siguiente:

```
tell application "FileMaker Pro"
        best type of menu 1
        class of menu 1
        default type of menu 1
        name of menu 1
        ID of menu 1
        enabled of menu 1
end tell
```

Menu Item cMenuItem
¿Qué se puede hacer?
Use el objeto Menu Item para acceder a elementos de menu de FileMaker. Por ejemplo, se puede decir:

```
tell application "FileMaker Pro"
        do menu menu item "Copiar" of menu "Edición"
end tell
```

Este guión dice a FileMaker que ejecute el quinto elemento de menú “Copiar” que aparece bajo el menu de edición.

Los usuarios tienen la posibilidad de crear elementos de menu en FileMaker. Un menu creado por el usuario se coloca automáticamente en un submenu llamado "External", bajo el menu de guiones. Los menus y los elementos de menu pueden ser anidados en el menu "External". Véase un ejemplo en la sección de ejemplos.

Nota: En tanto que todos los elementos de menu están deshabilitados cuando se ejecuta un guión de ScriptMaker, no se puede usar el evento Do Menu para seleccionar elementos de menu mientras se esté ejecutando un guión de FileMaker usando el pase de guión Ejecutar AppleScript. Además no se puede comprobar eficazmente el estado de un elemento de menu para comprobar el modo actual de la aplicación.

A continuación hay algunos Apple events que se aplican comúnmente a los elementos de menu, con ejemplos de sintaxis:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        create new menu item with properties ¬
                "Elemento" at menu "External" -- crear un nuevo elemento de menu
        delete menu item ¬
                "Elemento" of menu "External" -- borrar un elemento de menu
        exists menu item ¬
                "Elemento" of menu "External" -- comprobar la existencia de un elemento de menu
item exists
        class info menu item -- obtener la propiedad class de un elemento de menu
        set enabled of menu item 2 ¬
                of menu "External" to false -- deshabilitar un elemento de meniu
        set checked of menu item 2 of ¬
                menu "External" to true -- colocar una marca delante de un elemento de menu
end tell
```

¿Qué propiedades tiene?

Best Type La Best Type del objeto Menu Item es sencillamente 'obj ', significando objeto.
Class La Class del objeto Menu Item es 'cmen'.
Default Type La Default Type también es 'obj ', la misma que Best Type.
Name Devuelve el nombre del elemento de menu tal como aparece en la pantalla.
Los puntos suspensivos que aparecen tras algunos elementos de menú deben incluirse generándose al teclear Alt y punto.
ID La ID es un número únioco y persistente que se debe crear via Apple events.
Enabled Enabled es una propiedad lógica que devuelve verdad si el elemento de menu está habilitado actualmente (accesible), y falso si el elemento de menu está deshailitado.
Item Number La Item Number es el número de elemnento de menu, significando su posición dentro del menu.
Checked Checked en una propiedad lógicaque devuelve verdad si el elemento de menu está marcado actualmente, y falso en caso contrario.
Notify Address Notify Address es el ID de la aplicación que creó el elemento de menu. Se utiliza por los elementos de menu creados por el usuario en los menus externos. Por ejemplo, se puede crear un elemento de menu en FileMaker que llame a un guión o programa de AppleScript cuando el elemento de menu se seleccione. La aplicación que crea un menu debe tener alguna forma de responder cuando el elemento de menu se seleccione.

Ejemplos

Para acceder a un menu, se podría decir:

```
tell application "FileMaker Pro"
        do menu menu item "Buscar" of menu "Modo"
end tell
```

Para crear un elemento de menu definido por el usuario, se podría decir:

```
tell application "FileMaker Pro"
        create new menu item with properties "Mi Elemento de Menu"
        create new menu item with properties "nuevo elemento de menu2"
        create new menu item with properties ¬
                {name:"nuevo elemento de menu3", ID:123, enabled:false}
end tell
```

Para solicitar las propiedades de menu se puede utilizar lo siguiente:

```
tell application "FileMaker Pro"
        best type of menu item 1 of menu 2
        class of menu item 1 of menu 2
        default type of menu item 1 of menu 2
        name of menu item 1 of menu 2
        ID of menu item 1 of menu 2 -- sólo para elementos de menu creados por el usuario
        enabled of menu item 1 of menu 2
        item number of menu item 1 of menu 2
        checked of menu item 7 of menu 2
        notify address of menu item 1 of menu "external" -- sólo para elementos de menu creados
por el usuario
end tell
```

Record cRow

¿Qué se puede hacer?

Use el objeto **Record** para acceder a información a través de la base de datos. Los objetos **Record** se denominan también **files**.

A continuación hay algunos Apple events que se aplican comúnmente a registros, con sintaxis de ejemplo:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        copy record 1 to UnaVariable -- copiar un registro
        count record -- contar registros
        create new record with data "Bueno" -- crear un registro nuevo
        delete record 1 -- borrar un registro
        duplicate record 1 -- duplicar un registro
        record 5 exists -- comprueba si existe un registro
        class info record -- obtener la propiedad class de un registro
        get data record 1 -- obtener información de un registro
        data size record 1 -- contar los caracteres de un registro
        show (every record whose cell ¬
                "Preferencias" contains "Bueno") -- buscar un grupo de registros
        show every record of database 1 -- buscar todos los registros
        set record 1 to {"P.I. Tchaikovski", "Bueno"} -- establecer información en un registro
        every cell of current record -- obtener el contenido del registro actual
```

```
end tell
```

¿Qué propiedades tiene?

Best Type La Best Type del objeto Record es 'crow', significando fila, para registro.
Class La Class del objeto Record es 'crow'.
Default Type La Default Type es 'TEXT'.
Name Siempre devuelve "", en tanto que los registros de FileMaker no tienen nombres.
ID El ID es un número único y persistente designado a un registro.
Access La propiedad Access devuelve los privilegios de acceso de un registro, que puede ser uno o más de los siguientes: Sin Acceso (No Access), Lectura (Read), Escritura (Write), Modificable (Update), Crear (Create), No Borrar (No Delete), Borrar (Delete), Todos (Full).
Protection La propiedad protection devuelve el estado de protección del registro, que puede ser una o más de los siguientes: Sólo Lectura (Read Only),Protegido por fórmulas (Formulas Protected), Lectura (Read), Escritura (Write).
Lock La propiedad lock devuelve el estado de bloqueo del registro,que es una de las siguientes: Desbloqueado (Unlocked), Bloqueo Compartido (Shared Lock), Bloqueo exclusivo (Exclusive Lock).

Ejemplos:

Este guión obtiene toda la información del primer registro de la base de datos que está más al frente. Los datos se obtienen en modo de lista, si se deesea, se puede obtener en forma de texto tabulado indicando que sea "**as text**"

```
tell application "FileMaker Pro"
        record 1
end tell
```

Para obtener las propiedades del registro, se puede usar lo siguiente :

```
tell application "FileMaker Pro"
        best type of record 1
        class of record 1
        default type of record 1
        name of record 1
        ID of record 1
        access of record 1
        protection of record 1
        lock of record 1
end tell
```

Request cRequest
¿Qué se puede hacer?
Use el objeto **Request** para crear, modificar o acceder a una petición de búsqueda de FileMaker que puede devolver registros de una base de datos.

Para crear una nueva serie de peticiones, primero se ha de borrar cualquier petición previa usando la instrucción "**delete every request**", y luego use la instrucción "**create request**" para crear la primera petición. Tras crear una petición, use el evento **Find** para realizar la búsqueda.

El objeto **Request** tiene un comportamiento similar al objeto **Record** excepto que sólo se almacena un conjunto de peticiones en la base de datos. Por ejemplo, cuando se use el evento **Show** en una petición en particular, FileMaker mostrará todas las peticiones actuales y hará visible la especificada. Al usar un evento **Go To** o **Show** con una petición se cambiará la base de datos al modo Buscar.

Nota: al borrar todas las peticiones, habrá una petición nula (request 0) hasta que se cree una nueva petición o hasta que la base de datos se cambie a otro modo, en donde se creará al mismo tiempo una petición inical vacía (petición 1).

A continuación hay algunos Apple events que se aplican comúnmente a las peticiones, con sintaxis de ejemplo:

Se puede decir: Si se quiere:

```
tell application "FileMaker Pro"
        copy request 1 to MiVariable -- copiar una petición
        count request -- contar las peticiones
        create request with data "Bueno" -- crear una nueva petición
        create request with properties ¬
                {omitted:true} -- crear una petición omitida
        duplicate request 1 -- duplicar una petición
        request 5 exists -- ver si existe una petición
        class info request -- obtener la propiedad class de una petición
        get data request 1 -- obtener los datos de una petición
        go to request 4 -- ir a la petición 4
        show request 2 -- mostrar la petición 2
        show every request -- mostrar todas las peticiones
        set request 1 to {"Peter I. Tchaikovsky", "Bueno"} -- establecer datos de una petición
        set cell 1 of request 1 to "Bueno" -- establecer datos de una petición
        set omitted of request 2 to true -- omitir una petición
        find -- ejecutar las peticiones de búsqueda
        delete request 1 -- borrar una petición
        delete every request -- borrar todas las peticones
end tell
```

¿Qué propiedades tiene?

Best Type La Best Type de un objeto Request es 'list'.
Class La Class del objeto Request es 'cRQT'.
Default Type La Default Type es 'TEXT'.
Name Siempre devuelve "", en tanto que las peticiones de FileMaker no tienen nombres.
ID La es un número único designado para una petición.
Omitted Omitted es una propiedad lógica que devuelve verdad si se ha omitido una petición de búsqueda, y falso en caso contrario. La propiedad omitted se puede usar para excluir peticiones.

Ejemplos:

El siguiente guión busca cualquier registro con preferencias de Bueno en la base de datos más activa.

```
tell application "FileMaker Pro"
        delete every request
        create request
        set cell "Preferencias" of request 1 to "Bueno"
        find
end tell
```

Para obtener las propiedades de una petición se puede usar los siguiente:

```
tell application "FileMaker Pro"
        best type of request 1
        class of request 1
        default type of request 1
        name of request 1
        ID of request 1
        omitted of request 1
end tell
```

Window cWindow ¿Qué sepuede hacer?
Use el objeto Window para distinguir entre archivos de FileMaker que esté abiertos. Por ejemplo, se puede decir:

```
tell window 2 of application "FileMaker Pro"
 create new record with data {"Beethoven", "Mediocre"}
end tell
```

Esta sintaxis dirige las instrucciones a la segunda ventana de la aplicación FileMaker. La ventana actual es siempre la ventana 1.

A continuación hay algunos Apple events que se aplican comúnmente a **window**, con sintaxis de ejemplo:

Se puede decir: Si se quiere:

```
tell window 2 of application "FileMaker Pro"
        copy window 1 -- copia una especificación de objeto de la ventana
        close window 1 -- cerrar una ventana
        count windows -- contar el número de ventanas abiertas
        window "Lista CD" exists -- comprueba que esté abierta una ventana en particular
        class info window -- obtener la propioedad Class del objeto window
        get data window 1 -- obtener datos de una ventana (especificación de objeto)
        print alias "ventana1:Carpeta1:HardDisk" -- print a window
end tell
```

Se puede elegir usar un objeto **Window** sobre el objeto **Document**, en tanto que el objeto **Window** tiene ciertas propiedades el objeto **Document** no las tiene, tal como la propiedad de ampliación.

¿Qué propiedades tiene?

Best Type La Best Type del objeto Window es 'obj', significando objeto.
Class. La Class del objeto Window es 'cwin', significando ventana.
Default Type La Default Type es también 'obj', la misma que Best Type.
Bounds La propiedad Bounds devuelve las medidas el rectángulo que contiene la ventana.
Name Devuelve el nombre de la ventana tal como aparece en el escritorio. Este es el nombre que se tiene que usar entre instrucciones Tell desde AppleScript.
Visible Visible es una propiedad lógica que devuelve verdad si la ventana es visible, y falso en caso contrario.
Index La index de un objeto window es el número de la ventana, que indica su posición relativa entre las ventanas abiertas. La ventana activa siempre es 1, y las otras ventanas que están abiertas, si hay algunas, se numerarán a partir del 2, en orden consecutivo.
Floating Floating es una propiedad lógica que devuelve verdad si la ventana es flotante, y falso en caso contrario.
Zoomable Zoomable es una propiedad lógica que devuelve verdad si la ventana es apliable, y falso en caso contrario.
Zoomed Zoomed es una propiedad lógica que devuelve verdad si la ventana actual está ampliada, y falso en caso contrario.
Modal Modal es una propiedad lógica que devuelve verdad si la ventana es modal, y falso en caso contrario.
Resizable Resizable es una propiedad lógica que devuelve verdad si la ventana es re-dimensionable, y falso en caso contrario.
Has Close Box Has Close Box es una propiedad lógica que devuelve verdad si la ventana tiene cuadro de cierre, y falso en caso contrario.
Has Title Bar Has Title Bar es una propiedad lógica que devuelve verdad si la ventana tiene barra de título,y falso en caso contrario.

Ejemplos

El siguiente guión de AppleScript muestra cómo cerrar todas las ventanas abiertas:

```
tell application "FileMaker Pro"
        if window 1 exists then
                repeat count of window times
                        close window 1
                end repeat
        end if
end tell
```

Para obtener las propiedades de window, se puede usar lo siguiente:

```
tell window 1 of application "FileMaker Pro"
        best type of window 1
        class of window 1
```

```
        default type of window 1
        bounds of window 1
        name of window 1
        index of window "Lista CD"
        floating of window 1
        zoomable of window 1
        zoomed of window 1
        resizable of window 1
        has close box of window 1
        has title bar of window 1
end tell
```

Integrar FileMaker con otras Aplicaciones bajo Windows

En comparación con lo que acabamos de ver, FileMaker lleva peor suerte bajo sistemas operativos Windows®. No obstante, se puede llegar a recrear cierta inter-comunicación con otras aplicaciones.

FileMaker opta por ser, básicamente, “cliente”. Esto quiere decir que su comportamiento está enfocado a poder utilizar la información que tenemos en nuestra base de datos para manejarla con programas como, por ejemplo, Excel.

A pesar de ello, y a partir de la introducción del soporte de objetos ActiveX desde la versión 5, podremos llegar a manejar nuestra base de datos, e incluso, pasarle el resultado de la información que hayamos manejado.

Hay que tener en cuenta que, FileMaker, hasta la versión 7 no muestra lo que hayamos preparado para una u otra plataforma. De tal forma que, si usamos el paso de guión “Ejecutar AppleScript” el código introducido no se visualizará en Windows, e igualmente nos ocurre lo mismo si usamos el paso de guión “Enviar DDE Execute”.

De forma contraria, si utilizamos el paso de guión “Enviar Mensaje” se convierte en “Enviar Apple Event” en sistemas Mac OS. Gracias a que, con este paso de guión podemos utilizar un campo para incluir nuestro “mensaje”, podríamos utilizar un campo de cálculo donde tuviéramos el contenido para uno u otro sistema, y poder enviar el mensaje correcto en dependencia del sistema operativo que estemos utilizando en ese momento.

Introducción a DDE Execute

Dynamic data exchange (DDE) es el protocolo de inter-comunicación entre aplicaciones que permiten a los programas de Windows enviar y/o recibir información. Una aplicación puede ser cliente/servidora o ambos. Las aplicaciones cliente envían información o envían comandos a otras aplicaciones. Una aplicación que recibe información o comando es un servidor DDE. DDE puede referirse a una transferencia a la vez o una actualización de datos automática entre aplicaciones enlazadas.

FileMaker para Windows es una aplicación DDE cliente que permite manipular un archivo abierto en otro programa a través del comando de ScriptMaker Ejecutar comando DDE. FileMaker no puede recibir comandos DDE, y tampoco soporta el enlace de datos.

El paso de ScriptMaker Ejecutar DDE permite automatizar tareas entre aplicaciones, como pegar infromación en el documento de otra aplicación o crear un gráfico en un programa tipo hoja de cálculo. Teóricamente, cualquier comando que se pueda ejecutar con el lenguaje macro del servidor se podrá enviar por FileMaker al servidor a través de un comando DDE Execute. Para utilizar DDE correctamante, se necesita conocer tanto el lenguaje de script de la aplicción servidor (programa que recibe el comando) como la sintaxis DDE que soporta. Por ejemplo, algunas aplicaciones sólo comprenden los comandos DDE si estan están encerradas entre corchetes.

Para poder realizar una comunicacion entre FileMaker y la aplicación servidor a través de comandos DDE, ambas tienen que estar ejecutándose. Se puede intentar enviar antes el

comando “Enviar Mensaje…” de ScriptMaker para abrir automáticamente la aplicación servidor, pero puede ser más conveniente abrirlo manualmente. Cuando se ejecuta un comando con “Enviar mensaje…” a otra aplicación, la aplicaión que la recibe le devuelve el control a FileMaker cuando ha recibido el comando, no cuando ha terminado de ejecutarle. Por ello, FileMaker seguirá ejecutando el guión. Si el guión envía un comando DDE a una aplicación que no está completamente abierta, parecerá que este no ha hecho nada y se obtendrá un error.

El diálogo DDE Execute que aparece al preparar un guión tiene tres componentes: Nombre de Servicio, Tópico, y Comandos. El nombre de servicio es habitualmente el nombre de la aplicación tal como aparece en el archivo ejecutable. Por ejemplo, el nombre de servicio de Microsoft Word es Winword. FileMaker no tiene un nombre de servicio porque no recibe comandos DDE Execute. Para hacerlo más fácil de recordar, piense en Servicio=Servidor.

Si se piensa en DDE Execute como una conversación en un sentido, el tópico se refiere al “tópico de la conversación”. Si se quiere enviar comandos al mismo archivo todo el tiempo, el tópico de la conversación es ese archivo. El tópico debe incluir el nombre completo del documento, incluyendo su ruta. Opcionalmente, si se quiere enviar información al documento que está abierto por la aplicación actualmente o a un nuevo documento, se puede utilizar un nombre de tópico general. La mayoría de las aplicaciones permiten el nombre de tópico System, piense que WordPerfect para Windows utiliza el nombre de tópico Commands. No obstante, hay que tener cuidado cuando se utilice el tópico System. Si el comando DDE Execute pega en la aplicación servidor utilizando el tópico System, el comando DDE pegaría el contenido del portapapeles en el punto de insercción actual del documento abierto por el servidor, sin tener en cuenta de qué documento se trata.

Se puede especificar lo que quiere que haga la aplicación servidor en la caja de Comandos. Como se ha mencionado anteriormente, se necesita utilizar el lenguaje de macros de la aplicación servidor para que comprenda su tarea. Además, se tiene que saber la sintaxis utilizada por el servidor, así como si se necesita encerrar los comandos DDE entre corchetes.
Al contrario que con otras aplicaciones, no es necesario incluir DDEInitiate ni DDEExecute en el diálogo de comandos. Asimismo, FileMaker no tiene ningún requerimiento de sintaxis para involucrarse en una conversación; sólo se tiene que conocer los comandos que entenderán la aplicación servidor.

Por ejemplo, para enviar un comando DDE a Lotus 1-2-3 para Windows, se debe utilizar la sintaxis siguiente:

[RUN(comando)]
donde comando es una orden de macro o comando de menu clásico. Los comandos de Lotus 1-2-3 se deben encerrar entre llaves {}, y los comandos de menú se deben encerrar entre comillas “”. El comando
[RUN(“/DFA:A1..A:C10~~~~”)]
es un comando DDE válido que Lotus 1-2-3 para Windows interpreta como “Seleccionar el menu Datos, y rellenar desde la celda A1 hasta la C10”. Se puede teclear este comando directamente dentro del diálogo de comandos de DDE Execute de ScriptMaker, y funcionará.
El ejemplo DDE Execute de la figura 1 es equivalente a la macro de AmiPro siguiente:

```
FUNCTION 123_DDE()
id = DDEInitiate("123W","SYSTEM")
DDEExecute(id,"[[RUN(""/DFA:A1..A:C10~~~~"")]]")
```

DDETerminate(id)
END FUNCTION

Se puede comprobar que la orden DDE Execute del macro de AmiPro tiene paréntesis extras, corchetes y comillas que no se requieren por la aplicación servidor. Estos delimitadores extras son parte de la sintaxis cliente de AmiPro cuando este envía comandos via DDE. FileMaker no tiene ninguna sintaxis extra que debamos aprender.

FileMaker ofrece flexibilidad adicional al permitirnos especificar el nombre de servicio, el tópico, y los comandos en modo texto, de forma constante, o contenidos en un campo, que pueden variar. Aunque esta utilidad es un poco difícil de comprender en un primer momento, es muy útil cuando se quiere enviar una serie de comandos DDE Execute basados en el valor de un campo en registros distintos. Esto es todo lo que se tiene que saber acerca de cómo utilizar el paso de guión Enviar DDE Execute en FileMaker. A partir de aquí, se necesita información acerca de cómo recibe los comandos DDE Execute en nuestra aplicación servidor.
No obstante, al contrario que los enlaces, los comandos DDE Execute y su sintxis no están bien documentadas tanto de forma impresa como de forma de ayuda en línea por la mayoría de las aplicaciones. Algunas aplicaciones tienen kits de desarrollo de macros disponibles así como notas técnicas acerca de DDE. Además, los siguientes artículos pueden ayudar a iniciarnos:
Miranda, Lou. “Get Direction With DDE.” Windows Tech Journal April 1993: 22-27.
Strauss, Karen. “DDE? D-I-Y!” Windows User August 1993: 59-63.
La base de datos de conocimiento de Microsoft tiene muchos artículos relacionados con DDE. No hay que hechar de menos aquellos que traten de Windows 3.00 únicamente, ya que estos son válidos igualmente para versiones posteriores de Windows.

Usar DDE Execute para crear elementos de programa

Los siguientes guiones utilizan el paso de guión DDE Execute para crear items de programa, o iconos, para las bases de datos de FileMaker (Ver figura 1). Cuando se completa el guión, seremos capaces de abrir un documento haciendo doble click sobre su icono en el Administrador de Programas (Para más información acerca de la herramienta DDE Execute, se puede encontrar en el artículo número 102099, "DDE Execute Overview" de la base de datos TechInfo.)
Este es un buen ejercicio para cualquiera que quiera aprender acerca del uso del comando DDE Execute. También es una herramienta práctica para cualquiera que quiera automatizar la adición o el borrado de iconos de un grupo de programas. El Administrador de programas tiene un límite de 50 iconos de programa por grupo, no obstante, se puede necesitar modificar los guiones si se tienen más de 50 archivos. En tanto que los arhivos no se añaden ni se borran automáticamente en la ventana, los guiones están diseñados para reconstruir el contenido del grupo cada vez que se ejecuta.
Se va a crear una base de datos en FileMaker que no hace nada más que mantener una base de datos de los nombres de archivo y sacar información al gestor de aplicaciones para crear elementos de programa. Cuando se ejecute el script, se copian los nombres de todas las bases de datos creadas por FileMaker en el disco C: a un archivo de texto, llamado filelist.txt. Después se importa el texto a la base de datos, creando un registro diferente por cada archivo de la lista.
El campo al que se importa contendrá laruta completa a cada archivo, por lo tanto, cuando se hace un doble click sobre su icono, el administrador de programas sabrá en qué directorio encontrará el archivo. Un cálculo en la base de datos podrá extraer el nombre del archivo sin la ruta, que se utilizará por la descripción del icono de la aplicación. El tercer y último campo será un campo de cálculo que contendrá los comandos DDE Execute para crear elementos de pro-

grama para cada archivo.

Hay tres guiones en la solución. El primer guión no hace nada más que enviar un mensaje a un archivo por lotes (.bat) que crea archlist.txt. El segundo guión borra los elementos de programa existentes en el grupo de programa e importa la lista de los archivos de base de datos desde archlist.txt. El tercer guión llama al primero y envía una instrucción DDE Execute al administrador de programas en base al valor del campo de cálculo. Este campo de cálculo contendrá un instrucciones para decir al administrador de programas que añada un elemento de programa al grupo de archivos en base al nombre del archivo que contiene el campo de este registro.Cuando FileMaker completa el comando DDE Execute del primer registro, se despalza al siguiente registro, repitiendo este proceso hasta que alcanza el último registro de la bas de datos.
Instrucciones paso a paso
Antes de empezar: A fin de entablar en una conversación DDE, ambas aplicaciones, FileMaker y la aplicación servidora deben estar ejecutándose. Salvo que se esté utilizando un escritorio que no sea de Windows, el administrador de programas ya se estará ejecutando, por lo que no se tendrán que presentar problemas para el ejemplo siguiente
1. Cambiar al Administrador de Programas y seleccionar Nuevo en el menu Archivo. Seleccionar Grupo de programa y teclear Archivos tanto para la descripción como para el nombre del grupo. Hacer Click sobre OK.
2. Abrir un programa edior de textos, como Notepad, y teclear lo siguiente:

```
c:
cd\
dir *.fm /s /b /o > c:\fmpro\listarch.txt
```

Guardar el resultado como archivo .bat y salir del editor. Ejecutar el archivo por lotes haciendo un doble click sobre listarch.bat dentro del administrador de programas. Esto creará un archivo de texto con todos los archivos de FileMaker que estén en el disco C: (Par más información acerca de lo que lo que hacen los parámetros /s /b y /o, teclee "dir /?" en el DOS.)
3. Abrir FileMaker y crear un nuevo archivo. Se puede poner el nombre que se quiera. Crear los dos primeros campos:
Ruta (texto)
Archivo (cálculo, resultado texto) = Right (Ruta, (11 - Position (Right (Ruta, 11), "\", 1)))
El último campo que se crea, DDE Calc, es actualmente un comando DDE Execute que tiene la siguiente sintaxis:
AddItem(CommandLine, Name, IconPath, IconIndex, xPos, yPos)
donde:
• Command Line = C:\FMPRO\FMPRO.EXE (Ruta al archivo)
• Name (esta es la descripción del elemento de programa) = (Sólo Archivo)
• IconPath es blanco, indicando que la ruta del icono es la misma que la de la línea de comando
• IconIndex es 1, que es el ID de un documento de FileMaker
• xPos e yPos se omiten, por lo que los nuevos iconos se añadirán en el primer espacio que haya disponible en vez de en un lugar fijo
Crear DDE Calc como sigue, incluyendo las comillas al principio y al final de la línea. Para mayor claridad, el carácter ^ indica un espacio.
DDE Calc (calculation, text result)

```
= "[AddItem(C:\FMPRO\FMPRO.EXE^"^&^Ruta^&^","^&^Archivo^&^",,1)]"
```

4. Importar el archivo listarch.txt en el campo Ruta. (ListArch.txt se crea cuando se ejecuta archivo.bat creado en el paso 2.)
5.Crear los guiones en orden.
Script: Importar Nombres Archivo

Send Message:
Send the open document/application message
File: files.bat
Traer aplicación al frente
Script: Crear Elementos de Programa (ver Figura 2)
Ir al Registro/Petición [Salir después del último]
DDE Execute [“PROGMAN”]
Service Name: PROGMAN
Topic: PROGMAN
Commands (Field value):
DDE Calc
Perform Script [“Crear Elementos de Programa”]
Script: Actualizar Grupo de Programas
Buscar Todo
Borrar Todos [Sin diálogo]
Importar Registros desde ListArch.txt [Restaurar orden Importación, Sin diálogo]
Send message:
Send the open document/application message
File: PROGMAN.EXE
Traer aplicación al frente
DDE Execute [“PROGMAN”]
Service Name: PROGMAN
Topic: PROGMAN
Commands (Text):
[DeleteGroup(Files)]
[CreateGroup(Files,Files)]
[ShowGroup(Files,1)]
Ejecutar Guión [“Crear Elementos de Programa”]
Close []

No es necesario incluir "Crear Elementos de Programa en el menu de guiones. Para añadir iconos para cada base de datos, primero se ha de ejecutar "Importar Archivos" desde en menu de guiones para obtener una lista actualizada de los archivos de FileMaker del disco duro. Cuando este guión termine, ejecutar "Actualizar Grupo de Programas"

Nota: Como se ha dicho, el Administrador de programas tiene un límite de cincuenta elementos por grupo. Si tiene más de cincuenta archivos, puede parecer que estuviera fallando el guión; pero no es así. Tenga en cuenta que el administrador de programas no permitirá que se añadan más iconos, ScriptMaker seguirá procesando cada archivo. Cuando se complete el guión, nos volverá a FileMaker y cerrará la base de datos.

Los comandos DDE utilizados en este artículo están disponibles en el Microsoft KnowledgeBase, ID #Q72907.

Visual Basic Script

Visual Basic Script es una versión reducida, en cuanto a funcionalidades, del conocido Visual Basic. Se presenta esta opción como una más a tener en cuenta a la hora de trabajar con FileMaker en un PC.

Utilizaremos el mismo lenguaje de programación que se incluye en algunas páginas web, y, debido a que el "motor" lo tiene el sistema operativo Windows por defecto, también lo podremos utilizar, por ejemplo, con archivos que se mantengan en el mismo directorio donde está el desarrollo que hagamos con FileMaker. Veremos ejemplos que van desde cómo obtener la ruta a la carpeta Documentos, sea cual sea la versión del Windows que estemos utilizando, hasta cómo poder hacer que FileMaker ejecute un guión.

Hay que tener en cuenta que, frente a Visual Basic, Visual Basic Script es, como FileMaker, un lenguaje interpretado. Tiene la desventaja de, al ser abierto, poder ser modificado el código por cualquier usuario (hay una herramienta que permite cifrar el código para impedir que se pueda leer), pero la gran ventaja de ocupar menos espacio que la misma aplicación desarrollada con Visual Basic, ya que este último necesita incluir las librerías necesarias para su ejecución. Por lo demás, y dado que los guiones que desarrollaremos en Vbs no serán, a priori, muy complejos, la velocidad de ejecución entre uno y otro no difiere mucho.

Antes de poder ejecutar desde FileMaker los guiones desarrollados en VBS, se tiene que haber podido averiguar, como mínimo, la ruta que tenemos en el ordenador donde estamos trabajando hacia el motor CSscript, sobre todo si ejecutamos el guión pasando parámetros. Igualemente importante es poder comprobar la versión que se está utilizando (recomendable la versión 5 o superior, válida para windows 98 en adelante).

Los ejemplos incluidos se pueden generar en un archivo de texto normal, el cual lo guardaremos con el nombre que queramos, pero con la extensión .vbs. Una vez creado el archivo lo podremos ejecutar haciendo doble click sobre este directamente en el escritorio de Windows.

Buscar el WScript

CScript es el motor que utilizamos para poder ejecutar vbs desde la línea de comandos. Tendremos que especificar la ruta completa para poder ejecutar cualquier guión de vbs. El resultado lo meteremos en un archivo de texto que se llamará Resultado.txt

```
Dim fso, MiArchivo, elDirectorio' Declaración de variables
elDirectorio =  (WScript.Path) ' Obtenemos la ruta al Cscript
Set fso = CreateObject("Scripting.FileSystemObject")  ' Generamos el objeto "archivo"
Set MiArchivo = fso.CreateTextFile("c:\Resultado.txt", True)  ' Generamos el archivo en c:
MiArchivo.WriteLine(elDirectorio) ' le pasamos a línea
MiArchivo.Close ' cerramos el archivo
```

Tras haber generado este archivo, lo podremos incluir en nuestro desarrollo para utilizarlo en caso necesario, por ejemplo, haciendo una importación.

Crear archivos de texto.

Lo podemos utilizar para poder crear guiones en vbs utilizando, a su vez, este vbs, y poder ejecutarlo. Cada línea de comandos que queramos incluir se definirá como línea mediante el

comando **WriteLine**. Para poder incluir unas comillas se pondrán comillas dobles

```
Dim fso, MiArchivo
Set fso = CreateObject("Scripting.FileSystemObject")
Set MiArchivo = fso.CreateTextFile("c:\archivoPrueba.txt", True)
MiArchivo.WriteLine("Esto es una prueba.")
MiArchivo.Close
```

Carpetas especiales.

Son las "propias del usuario", es decir:
AllUsersDesktop
AllUsersStartMenu
AllUsersPrograms
AllUsersStartup
Desktop
Favorites
Fonts
MyDocuments
NetHood - Enorno de Red
PrintHood - Impresoras
Programs - Menu de Programas (dentro del menú Inicio)
Recent
SendTo
StartMenu - Menú Inicio
Startup
Templates

Estas tienen un lugar concreto para cada usuario del ordenador, así como para distintas versiones del sistema operativo. Para poder referirnos a ellas, tendremos que poder obtener su ruta a través de la función:

```
WshShell.SpecialFolders("LaCarpeta")
```

El siguiente ejemplo nos devolverá un diálogo con la ruta a la carpeta Escritorio. Podemos modificar el valor "**Desktop**" por cualquiera del listado anterior para obtener el resultado que queremos:

```
set WshShell = WScript.CreateObject("WScript.Shell")
Dim fso, laRuta
laRuta =  WshShell.SpecialFolders("Desktop")
fso = MsgBox(laRuta,65,"Ruta a mi Carpeta")
```

TRUCO: Con la incorporación de una nueva estructura de directorios en Windows®, es normal que se produzcan errores al intentar abrir archivos externos a FileMaker, por ejemplo, un archivo de texto. Usando un guión como el anterior nos podremos asegurar de poder encontrar siempre este archivo independientemente de la versión de Windows que se esté usando.

Controlar FileMaker

FileMaker se puede controlar desde un VBS mandándole "entradas de teclado", es decir, podemos hacer que FileMaker introduzca los caracteres que hayamos definido en nuestro guión. Esto

incluye también caracteres especiales, por lo que podremos ejecutar guiones de FileMaker que se muestren bajo el menú "Guiones", o cualquier otro ya que podremos tener acceso a las opciones del menú a las que tenga acceso el usuario que esté usando FileMaker.

Para ello utilizaremos el método **SendKeys**. Este envía una o varias pulsaciones de teclas a la ventana activa (como si se escribieran en el teclado).
objeto.SendKeys(cadena)
Argumentos
objeto
Objeto WshShell.
cadena
Cadena que indica qué pulsaciones de teclas desea enviar.

La mayoría de los caracteres del teclado se representan mediante la pulsación de una única tecla. Sin embargo, algunos caracteres se forman con la combinación de varias teclas (CTRL+MAY/S+INICIO, por ejemplo). Para enviar un único carácter del teclado, envíelo como el argumento cadena. Por ejemplo, para enviar la letra x, envíe el argumento cadena con el valor "x".

NOTA: Si desea enviar un espacio, envíe la cadena " ".
Puede utilizar SendKeys para enviar más de una pulsación simultáneamente. Para ello, cree un argumento de cadena compuesta que represente una secuencia de pulsaciones, para lo que tiene que escribir una detrás de otra. Por ejemplo, para enviar las pulsaciones de las teclas a, b y c, debe enviar el argumento de cadena "abc". El método SendKeys utiliza algunos caracteres como modificadores de los caracteres normales (en lugar de utilizar su valor impreso). Este conjunto de caracteres especiales incluye los paréntesis, llaves, corchetes y:
el signo más "+",
el acento circunflejo "^",
el signo de porcentaje "%",
y la tilde "~"
Para enviar estos caracteres, inclúyalos entre llaves "{}". Por ejemplo, para enviar el signo más, envíe el argumento cadena con el valor "{+}". Los corchetes ("[ ]") no tienen un significado especial para SendKeys, pero debe incluirlos entre llaves para que funcionen correctamente con aquellas aplicaciones que sí les dan un significado especial, por ejemplo, para el intercambio dinámico de datos (DDE, Dynamic Data Exchange).
Para enviar los corchetes como caracteres, envíe el argumento de cadena "{[}" para el corchete izquierdo y "{]}" para el corchete derecho.
Para enviar las llaves como caracteres, envíe el argumento de cadena "{{}" para el corchete izquierdo y "{}}" para el corchete derecho.
Algunas pulsaciones de teclas no generan caracteres (como ENTRAR y TAB). Otras pulsaciones representan acciones (como RETROCESO e INTERRUMPIR). Si desea enviar estos tipos de pulsaciones, envíe los argumentos que se muestran en la siguiente tabla:

Tecla	Argumento
RETROCESO	{BACKSPACE}, {BS} o {BKSP}
INTERRUMPIR	{BREAK}
BLOQ MAY/S	{CAPSLOCK}
SUPR o SUPRIMIR	{DELETE} o {DEL}
FLECHA ABAJO	{DOWN}
FIN	{END}

ENTRAR {ENTER} o ~
ESC {ESC}
AYUDA {HELP}
INICIO {HOME}
INS o INSERTAR {INSERT} o {INS}
FLECHA IZQUIERDA {LEFT}
BLOQ NUM {NUMLOCK}
AV PAG {PGDN}
RE PAG {PGUP}
IMPRIMIR PANTALLA {PRTSC}
FLECHA DERECHA {RIGHT}
BLOQ DESPL {SCROLLLOCK}
TAB {TAB}
FLECHA ARRIBA {UP}
F1 {F1}
F2 {F2}
F3 {F3}
F4 {F4}
F5 {F5}
F6 {F6}
F7 {F7}
F8 {F8}
F9 {F9}
F10 {F10}
F11 {F11}
F12 {F12}
F13 {F13}
F14 {F14}
F15 {F15}
F16 {F16}

Para enviar aquellos caracteres del teclado formados por la pulsación de una tecla normal combinada con MAY/S, CTRL o ALT, cree un argumento de cadena compuesta que represente dicha combinación. Para ello, incluya antes de la pulsación de la tecla normal uno o varios de los siguientes caracteres especiales:

Tecla Carácter especial
MAY/SCULAS +
CTRL ^
ALT %

NOTA: Estos caracteres especiales no se incluyen entre llaves cuando se utilizan de este modo. Si desea indicar que se debe mantener pulsada una combinación especial de las teclas MAY/SCULAS, CTRL y ALT mientras se presionan otras teclas, cree un argumento de cadena compuesta que incluya entre paréntesis las pulsaciones de teclas modificadas. Por ejemplo, para enviar una combinación de pulsaciones de teclas que indique que la tecla MAY/SCULAS se mantiene presionada mientras:
se presionan las teclas e y c, envíe el argumento de cadena "+(ec)".
se presiona la tecla e y, a continuación, una c (sin presionar MAY/SCULAS), envíe el argumento de cadena "+ec".

Puede utilizar el método SendKeys para enviar un patrón de teclas formado por una misma tecla presionada varias veces seguidas. Para ello, cree un argumento de cadena compuesta que indique la pulsación de tecla que desea repetir, seguida por el número de veces que desea repetirla. Para ello, utilice un argumento de cadena compuesta con el formato {pulsación número}. Por ejemplo, para enviar diez veces la letra "x", envíe el argumento de cadena "{x 10}". Asegúrese de incluir un espacio entre la pulsación de tecla y el número.
Nota Sólo es posible enviar como patrón de pulsación de teclas una única pulsación de tecla varias veces. Por ejemplo, puede enviar "x" diez veces, pero no puede hacer lo mismo con "Ctrl+x".
Nota No es posible enviar la tecla IMPRIMIR PANTALLA {PRTSC} a una aplicación.

Ejemplo:

```
' Enviamos “pulsaciones de teclado” a FileMaker, así que, de forma externa
' Podemos introducir datos y Ejecutar guiones

set WshShell = WScript.CreateObject("WScript.Shell")
'Inicio
'WshShell.Run "FileMaker Pro.exe"
'Llamamos a FileMaker (No es necesario indicar la ruta hacia el programa)
WScript.Sleep 100
'Vamos haciendo pausas

WshShell.AppActivate "FileMaker Pro"
WScript.Sleep 500
'Activamos FileMaker

WshShell.SendKeys "{ESC}5{enter}"
WScript.Sleep 500
' Si ejecutamos FileMaker, tenemos que “pulsar” Escape porque FileMaker muestra la pantalla primera


WshShell.SendKeys "{TAB}{tab}"
WScript.Sleep 500
' ”Tecleamos” tabuladores para meternos en un campo

WshShell.SendKeys "H"
WScript.Sleep 200
WshShell.SendKeys "O"
WScript.Sleep 200
WshShell.SendKeys "L"
WScript.Sleep 200
WshShell.SendKeys "A"
WScript.Sleep 200
‘ “Tecleamos” el texto que queramos

WshShell.SendKeys "{F3}+{F3}"
WScript.Sleep 200
‘ Hacemos un refresco de pantalla para salir del campo en el que estamos

WshShell.SendKeys "^1"
'Ejecutamos el primer guión (Crl+1)
WScript.Sleep 2500
```

Las posibilidades que nos ofrece esta función son múltiples, así, por ejemplo, podremos crear un tutorial, utilizando ayuda html, el cual nos controle lo que queremos mostrar o hacer con FileMaker, o con cualquier parte de nuestro desarrollo.

Funciones Externas

Una vez vistas las posibilidades que nos ofrece VBS, así como la posibilidad de controlar FileMaker, también podremos generar funciones externas a FileMaker, como generar alarmas, o realizar funciones que no se puedan hacer con FileMaker directamente. Para poder hacer esto, tendremos que generar una llamada a FileMaker a través del paso de guión "Enviar Mensaje" e incluyendo, como texto, por ejemplo, la llamada a nuestro archivo VBS (indicando la ruta completa, en caso necesario) y pasándole los parámetros separados por espacios.

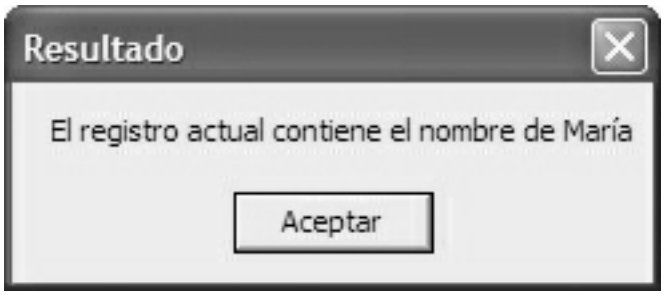
Por ejemplo, para poder mostrar un diálogo con un texto que le pasamos desde FileMaker, generamos el siguiente guión usando cualquier editor de texto:

```
set Args = Wscript.Arguments ' Obtenemos el parámetro pasado desde FileMaker
Nombre = args(0) ' Pasamos el valor dl parámetro (la primera) a la variable Nombre
Dim Mensaje
        Mensaje = "El registro actual contiene el nombre de " & Nombre ' Generamos el texto del mensaje
        MsgBox Mensaje, , "Resultado" ' Llamamos a la función que nos muestra el mensaje
```

Si tenemos el guión VBS dentro del archivo MiScript.vbs en el mismo directorio donde se encuentra nuestra base de datos, podemos hacer un guión en FileMaker con un único paso, "Enviar Mensaje", con el texto:

```
MiScript.vbs María
```

Al ejecutar este guión desde fileMaker, obtendremos un diálogo personalizado que utiliza el parámetro "María".



TRUCO: Para poder pasar parámetros que contengan caracteres especiales, o espacios, se tendrá que incluir entre comillas

Funciones de FileMaker

FileMaker, a partir de la versión 5 incluyen controles ActiveX. ActiveX nos permite controlar ciertos parámetros de la aplicación, estos, en FileMaker están definidos en la misma aplicación.

Desde su introducción con la versión 5, no ha cambio más que en la versión 7, para poder adaptarlos a esta.

La documentación incluida en FileMaker sólo hace referencia a Visual Basic. En tanto que el propósito de este apartado está centrado en Visual Basic Script, dado su carácter gratuito, los datos y ejemplos mostrados no están probados en Visual Basic.

Plugin, y mis "plugout" con VisualBasicScript

La tendencia actual de FileMaker está encaminada hacia los plugin, que son pequeños programas creados en C, con cualquiera de las herramientas existentes, para poder desarrollar distintas acciones (para ver una lista detallada de plugin disponibles, vea la web de FileMaker (www.filemaker.com). En cambio, Visual Basi, tanto la versión entera, como la reducida Visual Basic Script están dejadas de lado (¿con la esperanza que, quien trabaje con FileMaker, se introduzca en el mundo del desarrollo profesional?) en contra de la interfaz y herramientas que convierten a FileMaker en una base de datos muy intuitiva a la vez de fácil de usar por gente nada experta en bases de datos ni en programación.

En muchos casos, el uso de un plugin para Windows estaría justificado cuando se implantan acciones que, bien por su rapidez, bien por la falta de medios, se obtengan unos beneficios amortizables con la inversión a realizar en la adquisición de este. Recordemos que, al igual que Visual Basic en un PC, se pueden implantar acciones semejantes con AppleScript en un Apple, aunque la versión para Mac, en cuanto a este punto se refiere, es más potente. (Este sería el caso donde se justificara el uso de un plugin ya que nos evitaría tener que desarrollar dos códigos distintos si se usara mac o PC, ya que el uso de un único código implementaría las acciones que deseamos utilizar.) La mala noticia es que la mayoría de los plugin, a dia de hoy, que se hacen para FileMaker está optimizado para Mac, por no hablar de los costes que tienen algunos, sobre todo si se adaptan en un sistema multipuesto.

Muchos de los plugin que existen ofrecen acciones que se pueden crear fácilmente, con Visual Basic Script. Por esta razón, yo he pasado a denominarlos "plugout".

Antes de empezar

La "sencillez" y facilidad en la lectura de nuestros guiones que teníamos al usar AppleScript, aquí ya no es tal. Visual Basic se parece más a un "telegrama". De hecho, hay, incluso, términos que serán algo difíciles de relacionar, a la hora de querer aprenderlos.

Así, por ejemplo, para declarar una variable, hay que utilizar el término **Dim**. **Dim** es la abreviación de "Dimension", y su origen, al parecer, se remonta a "hacer lugar para colocar una variable en memoria". Esto tiene cierta lógica cuando los ordenadores tenían unos recursos de memoria muy limitados, pero... ¿por qué usarlo ahora cuando la memoria es tan económica?.

Por otra parte, hay que tener en cuenta la jerarquía que se va a utilizar, ya que esta tiene suma importancia a la hora de crear el guión. El desarrollo de esta jerarquía se hace separándolo con puntos. Por ejemplo, para referirme a este capítulo, lo tendríamos que hacer partiendo del libro, como objeto que se sitúa en la parte superior de la jerarquía:

Libro.Capitulos.Antes

Donde **Libro** es el objeto del cual vamos a decir algo, **Capitulos** será el tipo de objeto contenido en el libro, y **Antes** sería la abreviatura que utilizaríamos para referirnos a este capítulo "Antes de empezar", definido como objeto.

NOTA: Visual Basic Script no admite el uso de nombres de objeto tales como "Antes de empezar" así como texto acentuado. Hay que tener esto en cuenta, sobre todo, a la hora de crear guiones que vayamos a utilizar con Visual Basic Script.

Ejecutar un guión de FileMaker

El primer ejemplo que vamos a aplicar es ejecutar un guión de forma externa a FileMaker. Aunque es un ejemplo muy sencillo, nos vale para ver cómo hay que iniciar las variables en Visual Basic Script.

La documentación incluida con FileMaker se refiere únicamente a Visual Basic. Por lo que es distinto que lo expuesto en la documentación que viene con FileMaker.

Para poder utilizar este ejemplo, primero vamos a crear una base de datos, que llamaremos MiArchivo.fp5 y en el que crearemos, como poco, un guión, que llamaremos Mi Guion (sin incluir el acento). En este guión podremos incluir los pasos que queramos, pero para el propósito de este ejemplo, sería válido que incluyéramos un paso de guión que nos mostrara un mensaje. Así, visualmente, tendremos una confirmación, por parte de FileMaker, de que nos están saliendo bien las cosas. Este archivo, para mayor sencillez, lo ubicaremos en c:

Por otro lado, vamos a crear, con cualquier editor de texto, un archivo que le podremos llamar, por ejemplo, guion.vbs. En este archivo vamos a incluir el siguiente guión:

```
' Iniciamos las variables
' Llevan el prefijo FM únicamente para acordarnos de su uso

' Variable de la aplicación
Dim FMApp

' Variable para manejar documentos
Dim FMDocumentos

' Variable para manejar la base de datos que está más al frente
DIM FMDocumentoActual

' Establecemos el valor de la variable del programa
' Si no está abierto FileMaker, se abrirá ahora
Set FMApp = CreateObject("FMPRO.Application")

' Establecemos el valor de la variable para manejar documentos
Set FMDocumentos = FMApp.Documents

' Hacemos FileMaker visible (al abrirlo pemanece oculto por defecto)
FMAPP.Visible = True
```

```
' Abrimos una base de datos indicando su ruta y la clave
' También establecemos, al mismo tiempo, el valor de la variable para el documento actual
' que será el que esté más al frente (el que estamos abriendo)
' Si usamos FileMaker 5 ó 6, tenemos que pasar 2 parámetro, la ruta y la clave
Set FMDocumentoActual = FMDocumentos.Open("c:\MiArchivo.fp5","")
' Si usamos FileMaker 7, tenemos que pasar 3 parámetros la ruta, el nombre de cuenta y la clave
Set FMDocumentoActual = FMDocumentos.Open("c:\MyFile.fp7","Admin","")
' Tenemos que borrar la línea que no corresponda a la versión que estemos utilizando

' Ejecutamos el guión.
' Atención con los caracteres altos, por ejemplo, acentos
FMDocumentoActual.DoFMScript ("Mi Guion")
```

Una vez hecho este guión podremos comprobarlo haciendo doble click sobre el archivo vbs que acabamos de crear.

Abrir FileMaker con una clave inferior

Vamos a desarrollar un ejemplo de uso de Visual Basic Script que, debido a su complejidad, resulta muy ilustrativo para poder repasar las funciones que podremos hacer usando FileMaker y VBScript.

Este ejemplo esperará un tiempo, que definamos nosotros. Tras este tiempo, se cerrarán todos los archivos de FileMaker que tengamos abiertos, y se volverán a abrir con una clave inferior.

Este guión va a usar un archivo de FileMaker, así como dos archivos de vbs. En este ejemplo, se va a aplicar "c:" como la ruta a utilizar. Lógicamente, a la hora de modificarlo para un uso propio, la ruta se podrá controlar y cambiar por aquella que más nos convenga.

Archivo de FileMaker

En FileMaker, vamos a crear un nuevo archivo, al que llamaremos "tiempo". En este vamos a definir sólo dos campos:

- Tiempo, como campo global y datos de tipo numérico.
- cTiempo, como campo de cálculo con el resultado: `"wscript.exe C:\tiempo.vbs " & Tiempo`

El campo Tiempo es en el que vamos a introducir el tiempo

Igualmente, vamos a crear, con ScriptMaker, tres guiones. En cada uno de estos guiones sólo vamos a tener un paso de guión, que será "Enviar Mensaje". Para estos guiones utilizaremos los siguientes nombres y valores (usados en el paso "Enviar Mensaje"):

- Tiempo. Valor del campo "cTiempo"
- Sigue. Texto: "**wscript.exe C:\sigue.vbs 1**"
- No Sigue. Texto: "**WScript.exe c:\sigue.vbs 0**"

TRUCO: Para evitar que no se ejecuta nuestro guión vbs, podemos incluir, en el texto, el programa "**WScript.exe**" que es el que se encargará de ejecutarlo de manera correcta. Esto tiene más importancia con sistema el operativo inferior a Windows 2000. En este último caso, por ejemplo, si usamos Windows XP, se podrá omitir pudiendo indicarse, por ejemplo como: **C:\sigue.vbs 1**

Finalmente, vamos a crear dos claves, una, que puede ser la que queramos (y luego nos acordemos), que será la que tenga acceso a toda la base de datos. Crearemos otra clave, que será "123" sin ningún privilegio. De esta forma, creadas las dos claves, el archivo de FileMaker queda listo para poder usar en este ejemplo.

Archivos de VBS

Usando cualquier procesador de textos, vamos a crear dos archivos. Estos se encargarán de llevar a cabo toda la operatividad que vamos a implementar en este ejemplo. Es importante que, al guardarlos, incluyamos, al final del nombre, la extensión ".vbs"

En ambos ejemplos, en tanto que no sería deseable que un usario lo ejecutara desde el escritorio, por ejemplo, haciendo doble click sobre el archivo, se va a controlar que se pase, al menos, un parámetro (o argumento). En caso que no se pase ningún parámetro, se mostrará un mensaje advirtiendo que este guión sólo se puede ejecutar desde FileMaker.

SIGUE.VBS.
Este guión se va a encargar únicamente de escribir en un tercer archivo un texto. Este está compuesto sólo de un número 1 ó de un número 0, según queramos que nuestro guión principal (el que llamaremos al principio) sigua adelante, o no.

Este guión lo vamos a utilizar para "dar soporte" a nuetro guión principal y poder comunicarle si queremos seguir adelante o no. En tanto que el guión principal se va a quedar en una pausa, con el comando "**sleep**", además liberamos el motor de ejecución, por lo que no tendremos ningún inconveniente al ejecutar este guión de forma paralela al guión principal.

El archivo sigue.vbs será, por tanto:

```
' Este guión nos permitirá escibir un "0" ó un "1" en un archivo de texto,
‘ que llamaremos "sigue.txt"
' a través de este archivo, vamos a comprobar si tenemos que seguir esperando,
' con "1" (seguimos esperando)
' con "0" (ejecutamos un guión para cerrar FileMaker y lo abrimos con otra clave)

' Iniciamos las variables que usaremos en el guión
Dim fso, ElArchivo, ElTexto

Set fso = CreateObject("Scripting.FileSystemObject")

' Tomamos el parámetro pasado desde FileMaker, que será el texto a pasar al archivo
set Args = Wscript.Arguments

' Controlamos que se hayan recibido parámetros. Si se abre el guión con doble click
```

```
' no "se pasarían" parámetros, y, por tanto, mostramos un mensaje
if args.count < 1 then
    MsgBox "Este guión sólo funciona desde FileMaker",64,"A T E N C I O N"
    wscript.quit(1)
end if

ElTexto = args(0)

' Modificamos, o creamos el archivo, y le ponemos el texto que se ha recibido
Set ElArchivo = fso.OpenTextFile("c:\sigue.txt", 2)
ElArchivo.Write ElTexto     ' Pasamos el texto al archivo
ElArchivo.Close      ' Cerramos el archivo
```

Tiempo.vbs

El segundo archivo que vamos a crear contiene el guión principal, que será el que vamos a ejecutar desde el principio.

Este guión le vamos a ejecutar desde FileMaker pasándole un parámetro, que será el tiempo que definimos para esperar antes que se cierre FileMaker y se abra con una clave inferior.

En tanto que los tiempos de espera, vamos a querer que sean variables, es por lo que se ha puesto la llamada a este guión en un archivo de cálculo, así obtendremos el tiempo que se ha definido, por ejemplo, para cada usuario.

El contenido de este archivo será:

```
' Este guión nos permitirá abrir un archivo de FileMaker con una clave distinta
' tras un tiempo determinado al ejecutar este VBScript

' Para ello, vamos a usar un archivo de texto, externo, que llamaremos "sigue.txt"
' a través de este archivo, vamos a comprobar si tenemos que seguir esperando,
' con "1" (seguimos esperando)
' con "0" (ejecutamos un guión para cerrar FileMaker y lo abrimos con otra clave)

' Iniciamos las variables que usaremos en el guión
Dim Tiempo
Dim fso, ElArchivo, Sigo
Dim CompruebaArchivo

Set fso = CreateObject("Scripting.FileSystemObject")
Set CompruebaArchivo = CreateObject("Scripting.Dictionary")

' Tomamos el parámetro pasado desde FileMaker, que será el tiempo de espera
set Args = Wscript.Arguments

' Controlamos que se hayan recibido parámetros. Si se abre el guión con doble click
```

```
' no "se pasarían" parámetros, y, por tanto, mostramos un mensaje
if args.count < 1 then
    MsgBox "Este guión sólo funciona desde FileMaker",64,"A T E N C I O N"
    wscript.quit(1)
end if

Tiempo = args(0)

' Si el archivo, donde controlamos si seguimos o no, existe
If (fso.FileExists("c:\sigue.txt")) Then
      Set ElArchivo = fso.OpenTextFile("c:\sigue.txt", 2)
      ' Ponemos un 1, ya que, psiblemente, el valor anterior sería el 0
      ElArchivo.Write "1"
      ElArchivo.Close      ' Cerramos el archivo
End If

' Hacemos un bucle para esperar el tiempo definido desde FileMaker
' Cada vez que pase este tiempo, comprobaremos si el valor del archivo sigue.txt es 1 ó 0
Do
      If (fso.FileExists("c:\sigue.txt")) Then
            ' Si existe el archivo, leemos su contenido
            Set ElArchivo = fso.OpenTextFile("c:\sigue.txt", 1)
            Sigo = ElArchivo.ReadLine    ' Devuelve el valor contenido en el archivo
      Else
            ' Si no existe el archivo, lo creamos con el valor 1
            Set ElArchivo = fso.CreateTextFile("c:\sigue.txt", true)
            ElArchivo.WriteLine("1")
            Sigo = 1
      End If

      ElArchivo.Close      ' Cerramos el archivo

      ' Hacemos una pausa durante el tiempo especificado, en milisegundos
      WScript.Sleep Tiempo

Loop Until Sigo = 0

' Si el valor del archivo sigue.txt es 0, seguimos con el VBScript

' Empezamos iniciando las variables para controlar FileMaker
' Iniciamos las variables
' Llevan el prefijo FM únicamente para acordarnos de su uso

' Variable de la aplicación
Dim FMApp
```

```
' Variable para manejar documentos
Dim FMDocumentos

' Variable para manejar la base de datos que está más al frente
DIM FMDocumentoActual

' Establecemos el valor de la variable del programa
Set FMApp = CreateObject("FMPRO.Application.5")

' Establecemos el valor de la variable para manejar documentos
Set FMDocumentos = FMApp.Documents

' Hacemos FileMaker visible (al abrirlo pemanece oculto por defecto)
FMAPP.Visible = True

' Cerramos todas las bases de datos abiertas
FMDocumentos.Close()


' Abrimos la base de datos indicando su ruta y la clave
' También establecemos, al mismo tiempo, el valor de la variable para el documento actual
' que será el que esté más al frente (el que estamos abriendo)
' Si usamos FileMaker 5 ó 6, tenemos que pasar 2 parámetros, la ruta y la clave
Set FMDocumentoActual = FMDocumentos.Open("C:\tiempo.fp5","123")
```

Una vez tengamos estos archivos, ya podremos usar nuestro ejemplo. Para ello, tenemos que revisar lo siguiente:
- Que los tres archivos, tiempo.fp5, tiempo.vbs, y sigue.vbs están en c:
- Que tengamos un valor, en milisegundos, en el campo global “Tiempo”

Una vez revisado lo anterior, podemos proceder a ejecutar el primer guión “Tiempo”.

Este primer guión va a ejecutar tiempo.vbs, que se encargará de mantenerse en espera el tiempo que le hayamos indicado. Así, por ejemplo, si en el campo “Tiempo” hemos indicado 4000, el vbs hará pausas de 4 segundos antes de comprobar si tiene que seguir haciendo las pausas o no.

Una vez comprobado que está en correcto funcionamiento, podremos ejecutar el guión 3. Este se va a encargar de poner un 0 en el archivo “sigue.txt” que maneja el vbs anterior. A su vez, cuando este último “lea” el archivo “sigue.txt” y vea que hay un 0, procederá a cerrar todos los archivos, o bases de datos abiertas, y procederá a abrirlas de nuevo.

Objetos ActiveX soportados por FileMaker

FileMaker Pro soporta tres objetos ActiveX: **Application**, **Documents**, y **Document**. A continuación se exponen los métodos y propiedades disponibles para los objetos.

Hay que tener en cuenta que, para poder usarlos con Visual Basic Script, es necesario ir asignándolos de forma jerárquica para poder manejarlos. Esto es debido a que, en vbs no se pueden declarar variables como públicas o privadas, porque vbs sólo tiene un tipo de datos (variables). Para que podamos utilizar los objetos ActiveX disponibles para FileMaker, antes se tendrán que declarar como variables.

Objeto Application
Aceso a FileMaker

Propiedades

Nota: Todas las propiedades son de sólo lectura, excepto "**Visible**."
Application: Dirige los comandos a la aplicación, que es el objeto "raíz" de la jerarquía.
Parent: Dirige los comandos a este objeto.
FullName: Devuelve el nombre de la aplicación, incluyendo la ruta.
Name( ): Devuelve el nombre de la aplicación - "FileMaker Pro."
Caption: Devuelve el nombre de la base de datos activa. Caption no se puede utilizar para obtener otro nombre.
DefaultFilePath: Devuelve la ruta por defecto para abrir bases de datos.
Documents: Dirige los comandos al conjunto de bases de datos, así se podrán manejar cada una de forma individual para abrirlas, acceder a estas, y utilizar los guiones.
Version: Devuelve la version de FileMaker.
Visible: Devuelve **TRUE** si FileMaker es visible. Hay que establecerla a **TRUE** para mostrar la aplicación, **FALSE** para ocultarla. Por defecto, al iniciar FileMaker a través de vbs, se inicia de forma oculta, por lo que habrá de establecer **Visible = True** si se quiere mostrar FileMaker.
ScriptStatus( ): Devuelve 2 si se está ejecutando un guión, 1 si hay algun guión en pausa, o 0 si no se está ejecutando ningún guión.

Métodos

Quit( ): Cierra el programa. Hay que tener en cuenta que si hay clientes todavía adheridos todavía, la aplicación se oculta hasta que todos los clientes salen de las bases de datos que se están compartiendo. Para prevenir un comportamiento impredecible, es recomendable utilizar el método **Quit** vaciando el objeto que lo usaba. Por ejemplo:

A continuación ejemplos de propiedades y métodos usando el objeto Application:

Se puede decir: Si se quiere:

```
FMproApp.Quit
Set FMProApp = Nothing
```

Objeto Documents
Se aplica al conjunto de bases de datos abiertas.

Propiedades

Nota: Todas las propiedades son de sólo lectura.
Application: Dirige los comandos al objeto **Application**.
Parent: Dirige los comandos al objeto **Application**.
Count: Devuelve el número de bases de datos abiertas (integral) en la colección **Document**.
_NewEnum: Devuelve un objeto numerador para utilizar todos los objetos **Document** en el conjunto actual. Esta no es una propiedad explícita del objeto, pero la accesibilidad es implícita cuando se usa en un bucle con "For".
Active: Devuelve la base de datos activa.
Item(variable): Devuelve un objeto **Document** de la colección. Este método es el miembro por defecto en el conjunto **Documents**. Toma como parámetro una variable que se puede especificar como:

Una cadena que represente el nombre del archivo (la ruta completa)
Un índice (Número integral) del conjunto document.
NULL (que devuelve todo el conjunto de documentos)

Metodos

Open(NombreArchivo, Clave): Abre una base de datos de FileMaker, crea un objeto **Document**, y dirige los comandos a este objeto **Document**. Los parámetros de archivo, usuario (a partir de FileMaler 7), y clave hay que pasarlos como texto.
Close( ): Cierra todas las bases de datos del conjunto actual y las quita de este conjunto. El método **Close** produce un cierre repentino de las bases de datos. Si hay otros usuarios conectados a estas bases de datos cuando se envía el método **Close**, se desconectarán inmediatamente sin previo aviso. Hay que asegurarse de permitir a los usuarios cerrar las bases de datos que están usando antes de usar este método.

Objeto Document
Un objeto "**Document**" es una base de datos, tabla de FileMaker.

Propiedades

Nota: Todas las propiedades son de sólo lectura
Application: Dirige los comandos al objeto **Application**.
Parent: Dirige los comandos al objeto del conjunto de bases de datos, **Documents**.
FullName: Devuelve el nombre de la base de datos que se está usando, incluyendo su ruta. Si se usa la propiedad **FullName** con una base de datos que esté abierta en un servidor remoto, **FullName** devolverá sólo el nombre del archivo, y no su ruta.
Path: Devuelve la especificación de ruta de la base de datos. Esta propiedad no devuelve el nombre de la base de datos ni la extensión.
Saved: Devuelve el estado del documento, es decir, si la base de datos se ha guardado. FileMaker Pro siempre devuelve **TRUE**.
Active: Devuelve **TRUE** si la base de datos del objeto **Document** está activa, si no está activa, o está minimizada, devolverá **FALSE**.

Metodos

Activate( ): Activa la base de datos o tabla asociada con el objeto **Document**.
Save( ): Vacía la cache de la base de datos.
Close( ): Cierra la base de datos y la quita del conjunto de bases de datos del objeto **Documents**.

DoFMScript(Guion): Ejecuta un guión de FileMaker que esté en el objeto **Document** seleccionado. El guión se especifica por nombre, debe existir en la base de datos. Hay que tener en cuenta que los caracteres altos (fuera de ASCII 128) no se reconocen, por lo que hay que tener cuidado con los guiones con caracteres acentuados, por jemplo, Mi Guión no funcionaría mientras que su nombre tuviera el caracter "ó", mientras que quitando el acento, es decir, Mi Guion sí funcionaría.

TRUCO: Cuando se tienen instaladas las versiones 7 y cualquier otra versión entre la 5 y la 7, se puede añadir ".7" ó ".5" al final de cualquier objeto para poder dirigir los comandos de forma correcta a la versión que queramos controlar, por ejemplo "FMPRO.Application.5", "FMPRO.Document.5, o "FMPRO.Documents.5" para las versiones anteriores a la 7.