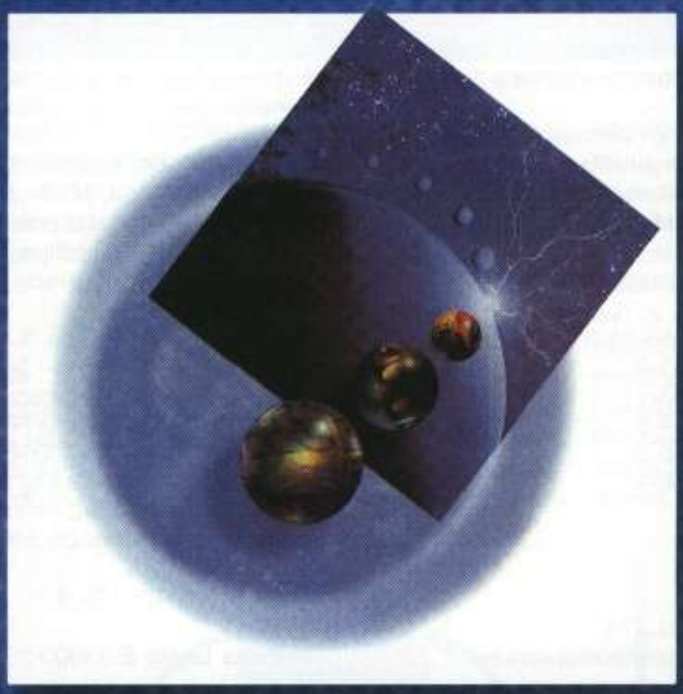
CURSO IBM

PROGRAMAR es fácil



5

IBM WebSphere Studio

Programación en Internet

Multimedia Ediciones, S.A.

Conceptos de programación (5)

Funciones y control de flujo

La utilización de funciones y expresiones de control de flujo constituyen la base de una buena programación estructurada. Las funciones, debido a que permiten gestionar tareas repetitivas concretas y las expresiones de control de flujo por su influencia en la toma de decisiones.

Las funciones y las subrutinas son de las ayudas más valiosas a la programación, que podemos encontrar, ya que permiten descomponer fácilmente tareas complejas en otras más sencillas, mucho más fáciles de controlar y depurar.

FUNCIONES Y SUBRUTINAS

La manera más eficaz de programar es mediante el dicho, "divide y vencerás": tomamos el problema, lo dividimos en partes y vamos resolviendo cada una de ellas. Además, si esas partes se repiten frecuentemente, podemos desarrollar un pequeño programa que ejecute ese trabajo y luego usarlo cuando sea necesario. Por ejemplo, en un programa de "marcianos" hará falta saber si un disparo ha dado en un blanco, y repetir esa operación para cada uno de los "marcianos" de la pantalla. A esta pequeña parte del programa que desarrolla una tarea concreta la denominaremos *subrutina* o *procedimiento*.

Un procedimiento tiene un nombre que le asignamos; estos nombres suelen seguir las mismas reglas que los de variables. En Java (y otros lenguajes similares como C y C++) un procedimiento o función lleva detrás un paréntesis. Los procedimientos y funciones han de declararse en la mayoría de los lenguajes, al igual que las variables. Nombres válidos de procedimientos en Java serían:

```
imprimir()
VerSiguienteRegistro()
```

A un procedimiento se le pueden pasar *parámetros*, es decir, unos valores que el programa principal le envía a esa subrutina para que ésta actúe sobre ellos. Así, siguiendo con nuestro hipotético juego, la subrutina EvaluarImpacto, que verifica si nuestro disparo ha dado a un "marciano", necesita saber qué "marciano" hay que evaluar. El "marciano" (mejor dicho, el conjunto de todos los datos necesarios para evaluar la situación y estado de uno de ellos) es el parámetro que le mandamos a la rutina. Internamente, esta rutina usará esos datos como quiera para llevar a cabo su tarea. En muchos lenguajes los parámetros se pasan como valores dentro de los paréntesis que acompañan al nombre del procedimiento.

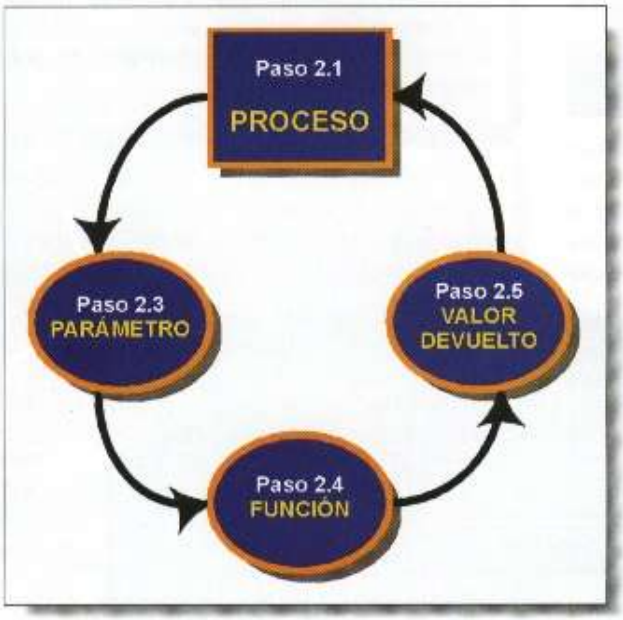


FIG 1. Un parámetro es un valor que se le pasa a un procedimiento para que éste lo use.

Para usar un procedimiento en estos lenguajes, escribimos su nombre (paréntesis incluidos) y lo terminamos con punto y coma. Por ejemplo, para imprimir diez veces la palabra "Hola" podríamos usar algo parecido a lo que describimos en el siguiente pseudocódigo:

```
entero L;
L = 0;
Repetir
Si L < 10
drawString(«Hola»);
      L = L + 1;
En caso Contrario
            END
```

En este ejemplo, *drawString ("Hola")* es una llamada al pro-

FIG 2. Gracias a las librerías se facilita el reciclaje de los procedimientos y funciones que tanto trabajo nos ha costado crear.

cedimiento *drawString* al que le pasamos como parámetro la cadena de caracteres "Hola" y que realiza la impresión de dicha cadena. La diferencia fundamental entre un procedimiento y una función es que el procedimiento no devuelve ningún valor, mientras que la función sí, y por tanto puede ser utilizada en el contexto de expresiones más complejas. Por ejemplo, imaginemos la función *Coseno* que devuelve el valor del coseno del ángulo que se le pasa como parámetro. Podría usarse así:

```
Altura = Coseno(Lado) * 3;
```

Donde *Altura* y *Lado* son variables que suponemos definidas en otras partes del programa. Como es fácil ver, la función tendrá también, como las variables, un tipo, el tipo de dato del valor que devuelve (en este caso un número en coma flotante) y debe ser declarada según las reglas del lenguaje concreto de que se trate.

RUNTIME

Un concepto asociado al de las bibliotecas estándar es el conocido como *librerías de tiempo de ejecución* o, empleando el término anglosajón, librerías *runtime*. Cualquiera que haya usado Visual Basic, seguramente habrá visto algún mensaje indicando que falta determinada librería "runtime". Cuando se compila un programa se debe añadir al código generado el de las librerías que utiliza internamente. Como resultado puede que un programa corto acabe siendo grande si hace uso de unas rutinas que, a su vez, lo hacen de otras. Esto puede constituir una desventaja.

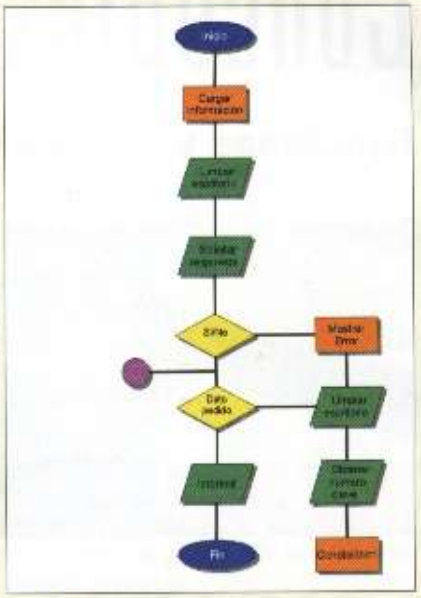


FIG 3. Los módulos o librerías *runtime* permiten a los programas reducir su tamaño y compartir funciones comunes.

La solución a este problema es reunir esas librerías en un único objeto binario, una librería de enlace dinámico o DLL, que puede estar instalada en el sistema del usuario. Al compilar los programas, estos hacen uso de esa librería, de manera que el programa compilado contiene enlaces a las funciones necesarias pero no las funciones en sí, ahorrando una considerable porción de espacio.

La parte negativa estriba en que dicha librería debe estar correctamente instalada en el sistema donde se va a ejecutar el programa. De lo contrario se producirá el error antes mencionado. Otra posible fuente de problemas es el control de versiones, ya que, en la práctica, pueden existir diferentes versiones de una misma librería no compatibles entre sí, por lo que si esta se emplea un programa y la librería instalada no es la correcta pueden darse toda suerte de resultados extraños.

FIG 4. Las sentencias de control de flujo permiten definir el orden en la secuencia de instrucciones de un programa.

DE REGALO... O NO

Lo bueno de desarrollar procedimientos y funciones es que, una vez creada una rutina que hace algo, no hay que volver a programar esa operación: basta con volver a usar ese procedimiento o función en todos nuestros programas posteriores. Resulta relativamente fácil crear *librerías* o *bibliotecas* de funciones y procedimientos, en las que se reúnen rutinas y más rutinas agrupadas generalmente en una cierta clase (por ejemplo, funciones matemáticas, de tratamiento de archivos, etc.).

De hecho, los compiladores e intérpretes ya incluyen una serie de librerías, más o menos extensas, con las

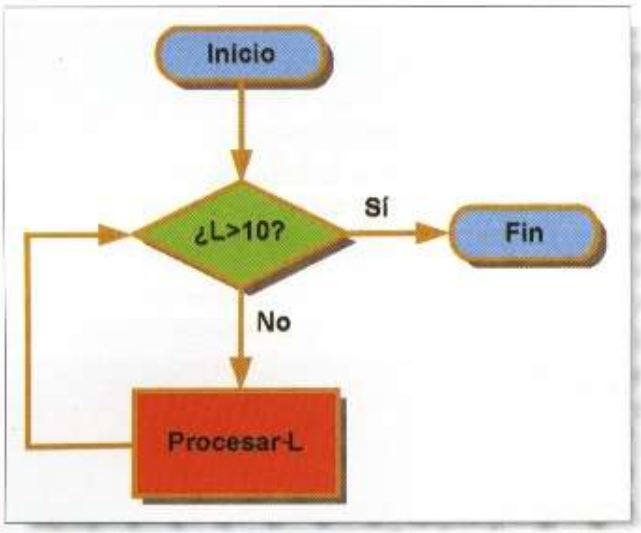


FIG 5. Una sentencia de salto condicional permite ejecutar unas instrucciones u otras en función de cierto resultado.

funciones necesarias para la programación habitual. Además de estas librerías suministradas por los fabricantes, podemos añadir otras, bien compradas, cargadas desde Internet o desarrolladas por nosotros mismos.

Cuando se programan procedimientos y funciones es importante definir cuidadosamente la *interfaz* con el resto de programas; es especialmente crítico elegir bien los parámetros y, sobre todo, no hacer uso de variables externas que a su vez sean accesibles desde otras partes del programa.

CÓDIGO SPAGHETTI

Las sentencias de salto incondicional, si se emplean en exceso pueden dar lugar a que el programa sea muy difícil de entender. Es precisamente ésta una de las causas por las que el lenguaje BASIC gozó de mala fama entre los profesionales. De ahí el término "código Spaghetti", usado para designar los programas que, de tan enrevesados a causa del abuso de esta sentencia, resultan incomprensibles. En general, y salvo casos particulares, puede sustituirse un GOTO por secuencias apropiadas de instrucciones (como bucles o saltos condicionales) más fáciles de leer.

FIG 6. El abuso de sentencias GOTO puede hacer incomprensible un programa, perdiéndose fácilmente el hilo de la ejecución.

CONTROL DE FLUJO

Existe un grupo de instrucciones que permiten controlar la secuencia de ejecución de las instrucciones, vigilar la repetición de determinadas secuencias y tomar decisiones en función de los datos disponibles en cada instante. Se las conoce como instrucciones de *control de flujo*.

La instrucción más sencilla es el salto incondicional, esto es, transfiere la ejecución del programa de un punto a otro.

Suele llamarse GOTO (del inglés, Go To, Ir a) o JUMP (saltar), y necesita un parámetro para indicar el punto al que hay que saltar.

Esta indicación suele hacerse por referencia, como un nombre de etiqueta o el número de la línea (como se hace en las versiones originales de Basic). Por ejemplo:

```
entero L;
L = 0;                          //
Inicializar la variable L
Aqui:                           //
Esto es una etiqueta de salto
Si L < 10                       //
Repetir 10 veces
drawString("Hola");     // Sacar
mensaje
L = L + 1;              //
Incrementar la variable L
GOTO Aqui;          // Ir al salto
para repetir todo
END                             // Fin
del programa
```

Las sentencias de *salto condicional* permiten ejecutar unas instrucciones u otras en función de una condición. Esta instrucción suele representarse con el comando IF (el si condicional del idioma inglés). Por ejemplo, en Java se usa la forma:

```
if (condicion)
Bloque de instrucciones;
```

donde *condicion* es una expresión. Si el resultado es distinto de cero se toma como verdadero, y se ejecuta la instrucción o bloque de instrucciones que le siguen.

Opcionalmente, a una instrucción *if* le puede seguir una cláusula *else* seguida también de un bloque de instruc-

ciones, que serán ejecutadas solamente si la condición del *if* resulta ser falsa (esto es, la expresión da como resultado cero). Por tanto, una secuencia *if...else* de Java tiene la estructura siguiente:

```
if (condicion)
      Bloque1;
else
      Bloque2;
```

Si *condicion* es distinto de cero (verdadero), se ejecuta *Bloque1*; en caso contrario, se ejecuta *Bloque2*.

Por ejemplo, el pseudocódigo para mostrar con seguridad el resultado de la división de dos números, *a* y *b*, sería:

```
if( b != 0)
{     resultado = a / b;
}
else
{     imprimir ("¡No se puede
dividir por cero!");
      resultado = 0;
      END;
}
```

El operador "!=" representa "distinto de", que compara dos valores y da "verdadero" si ambos son diferentes. Las llaves delimitan un bloque de instrucciones. El programa anterior podría modificarse así:

```
if(b)
{     resultado = a / b;
}
else
{     imprimir ("¡No se puede
dividir por cero!");
      resultado = 0;
      END;
}
```

VUELTAS Y MÁS VUELTAS...

Otro tipo de sentencia fundamental es la que permite controlar repeticiones de instrucciones; por ejemplo, a la hora de pintar un gráfico habrá que pintar un punto, luego el siguiente, luego el siguiente, y así sucesivamente. En realidad, esto podría conseguirse con secuencias de IF y GOTO adecuadas, pero el uso de instrucciones de *control de bucles* o *iteraciones* simplifica el programa tanto al escribirlo como al comprenderlo, dejando al compilador o al intérprete la tarea de decidir y calcular los puntos de salto que sean necesarios.

Analizaremos la sintaxis en Java para describir algunas de sus propiedades. La primera es la instrucción *while*, que adopta la forma siguiente:

```
while (condicion)
      Bloque;
```

donde *condicion* es una expresión. Si el resultado es distinto de cero se toma como verdadero, y se ejecuta la instrucción o bloque de instrucciones que le siguen. En caso contrario, se ignora dicho bloque y la ejecución continúa en la primera línea de programa que haya a continuación del mismo. Por ejemplo, un programa para mostrar en pantalla los números de uno a 10 podría ser así:

```
L = 1;
while( L <= 10)
{     Imprimir (L);
      L = L + 1;
}
```

Con un bucle *while* puede crearse cualquier tipo de iteración, pero como se ve, la condición se evalúa siempre antes de ejecutar el bloque de instrucciones asociado; por ello, el bloque se ejecutará entre cero e infinitas veces. En nuestro ejemplo, diez veces.

En ocasiones, sin embargo, interesa asegurar que el bloque se ejecute al menos una vez. Una solución sería copiar ese bloque antes del bucle para que se ejecute una vez, y luego entrar en el bucle una vez menos; por ejemplo, el caso anterior podría reescribirse:

```
L = 1;
Imprimir (L);
L = L + 1;
while( L <= 10)
{     Imprimir (L);
      L = L + 1;
}
```

La mayoría de lenguajes ofrecen una variante de bucles, los *do...while* cuya forma es la siguiente:

```
do
      Bloque;
while (condicion);
```

donde *condicion* es una expresión. La diferencia es que ésta se evalúa *después* de haber ejecutado el bloque de sentencias.

El ejemplo anterior podría ser reescrito de nuevo como:

```
L = 1;
do
{     Imprimir (L);
      L = L + 1;
}
while( L <= 10);
```

Existe un tercer tipo de bucles en Java, BASIC y muchos otros lenguajes: los bucles *for*, que vienen a ser un complemento de los *while*. Su estructura es la siguiente:

```
for (inicio;condicion;bucle)
      Bloque;
```

inicio es una instrucción que se ejecuta solo una vez al principio del bucle, antes de la primera iteración. *condicion* es, igual que antes, una expresión, y *bucle* es una instrucción que se ejecuta una vez en cada iteración después del bloque asociado al *for*. Un bucle *for* es igual a un bucle *while* de la siguiente forma:

```
inicio;
while(condicion)
{     Bloque;
      bucle;
}
```

La ventaja fundamental del bucle *for* es que las operaciones se representan de forma compacta; por ejemplo, el programa anterior podría reescribirse así:

```
for (L=1;L<= 10; L=L+1)
{     Imprimir(L);
}
```

Con todo lo repasado someramente, estamos ya en disposición de comprender las estructuras fundamentales de un programa.

Soluciones prácticas (5)

TechSmith SnagIt: capturas de pantallas

Cualquier usuario que haya documentado convenientemente uno de sus programas sabrá que crear un buen manual de uso o una ayuda en línea exige la realización de una tediosa tarea: la captura de pantallas. A lo largo de estas líneas va a conocer la aplicación TechSmith SnagIt, uno de los mejores capturadores de pantallas que existen en la actualidad.

Se dice que más vale una imagen que mil palabras, algo que es especialmente cierto en el caso de un buen manual de uso de una aplicación o de una ayuda en línea. Es por ello que en el momento de abordar la creación de la documentación que acompaña a una aplicación, el programador o grupo de programadores suele incluir un buen número de capturas de pantalla del programa en funcionamiento.

Ahora bien, capturar la pantalla es un proceso bastante tedioso, habida cuenta de que es necesario pulsar la tecla **Impr Pant** (o **Alt + Impr Pant**, para capturar cuadros de diálogo), acceder a una aplicación de retoque fotográfico (puede ser tan sencilla como Paint, por ejemplo), pegar el contenido del portapapeles con la combinación de teclas **Control + C** y, por último, guardar la imagen con un nombre. Si es necesario hacer tres capturas, este método es suficiente; si se deben hacer 300, hay que buscar otro procedimiento alternativo que permita ahorrar el máximo tiempo (y trabajo) posible. Aquí es donde entra en juego TechSmith SnagIt.

TechSmith SnagIt puede automatizar completamente el proceso de captura de pantallas. Para que se haga una idea, puede conseguir que una captura se guarde en el disco duro con el formato de imagen que sea necesario y con un nombre determinado, y todo ello con una simple pulsación de tecla. Es más, como en ocasiones le surgirá la necesidad de capturar un área concreta de la pantalla (un cuadro de diálogo, una zona de la pantalla de tamaño prefijado, etc.), TechSmith SnagIt le ofrece la posibilidad de definir qué quiere capturar en el momento de iniciar el proceso de captura, ahorrándole el tiempo de edición y modificación en un programa de retoque fotográfico.

Ahora que ya hemos comentado someramente las principales características de TechSmith SnagIt, está preparado para comenzar a explorar a fondo todas sus posibilidades.

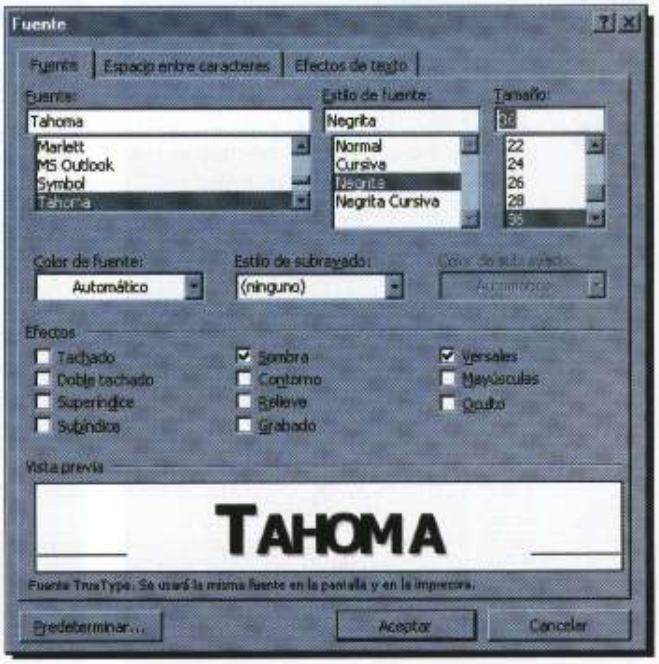


FIG. 1 Las capturas de pantalla son necesarias para informar al lector del estado del programa en un determinado punto de su ejecución o en el momento de realizar una tarea concreta.

INICIO

Cuando ejecute TechSmith SnagIt aparecerá un asistente llamado **"Quick Start" Wizard**, especialmente diseñado para facilitarle al máximo el proceso de captura de pantallas. Este primer paso del asistente le explicará (en inglés) las tres operaciones básicas de la captura de pantallas con la aplicación; a saber:

- Seleccionar lo que se desea capturar, es decir, definir qué elemento o elementos de la pantalla se deben capturar.
- Elegir dónde irá a parar la captura.
- Pulsar la tecla rápida de captura.

Como ya imaginará, los dos primeros pasos se llevan a cabo al comienzo de una

nueva sesión de captura de pantallas completamente distinta a la que se realizó con anterioridad (TechSmith SnagIt guarda los cambios en su configuración sin necesidad de realizar el proceso de forma manual).

Al pie del cuadro de la ventana encontrará una última opción, **Do not show me this wizard again**, es decir, *No me muestres este asistente otra vez*, que sólo deberá activar cuando esté familiarizado con TechSmith SnagIt y sus opciones de configuración. Para continuar, haga clic sobre **Siguiente**. La pantalla que está contemplando en estos momentos le permitirá configurar el primero de los apartados que hemos mencionado más arriba: qué capturar. Existen tres opciones, aunque más tarde veremos que podrá elegir bastantes más:

- **Screen**: captura la pantalla al completo, es decir, todo lo que ve en su monitor.
- **Window**: captura la ventana que usted mismo elija con el ratón. Es, con mucho, la opción de captura más versátil que posee TechSmith SnagIt, pues permite elegir en todo momento si se debe capturar la pantalla completa, un cuadro de diálogo o una paleta de herramientas determinada, por ejemplo. Si necesita hacer capturas variadas, use preferentemente esta opción. El proceso de captura es tan sencillo como pulsar la tecla o combinación de teclas elegidas (más tarde llegaremos a este tema), situar el ratón sobre el objeto gráfico a capturar (se enmarcará automáticamente para que pueda ver con precisión qué se va a procesar) y hacer clic con el botón izquierdo.
- **Region**: le permite capturar una zona concreta de la pantalla interactivamente. Para definir el área a capturar es necesario trazar un rectángulo con el ratón haciendo clic con el botón izquierdo del ratón y, manteniéndolo pulsado, crear un área antes de soltar el botón.

Si desea obtener más información sobre los distintos modos de captura que posee TechSmith SnagIt, puede hacer clic sobre el botón **More Info**.

SHAREWARE

Shareware es un tipo de distribución de software que permite su evaluación antes de la compra. Sus ventajas son evidentes: permite que el usuario compruebe si el programa que está probando es realmente el que le interesa, el que dispone de las funciones que está buscando. No obstante, el programa se puede usar libremente durante un espacio de tiempo determinado, normalmente dos semanas o incluso un mes; en el caso de este programa, el periodo de evaluación es de 45 días. Al término de este tiempo es necesario registrarse, es decir, comprar el programa, o desinstalarlo del disco duro (muchas aplicaciones dejan de funcionar transcurrido el tiempo de prueba). TechSmith SnagIt no es un programa caro, sobre todo si se tiene en cuenta la enorme cantidad de tiempo que permite ahorrar a los que aprovechan su potencia.

FIG. 2 Si hace una captura manualmente, deberá pegarla en un editor de imágenes para, posteriormente, guardarla en el disco duro con un formato determinado.

FIG. 3 TechSmith SnagIt permite automatizar el proceso de captura de pantallas.

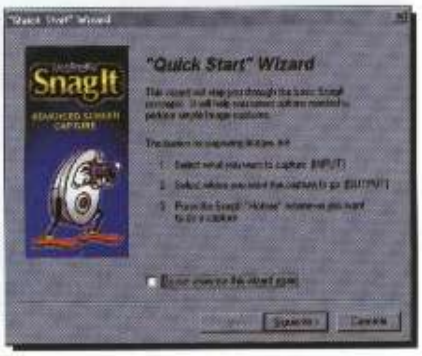


FIG. 4 Esta primera pantalla del asistente le explica cuál es el proceso básico de captura de pantalla.

Por último, elija la opción **Window** antes de hacer clic sobre **Siguiente**. Ahora está contemplando la pantalla donde puede elegir qué se hará con la captura de pantalla. Estas son las opciones disponibles:

- **Printer**: eligiendo esta opción, TechSmith SnagIt enviará la captura de pantalla a la impresora predeterminada de Windows.
- **Graphics file**: la captura se guardará como un archivo de imagen que podrá abrir posteriormente con programas de retoque de imágenes o insertarlos en documentos de Word, por poner dos ejemplos. Ésta es la opción más interesante de las tres prácticamente en cualquier caso, ya que permite disponer de la imagen a posteriori.

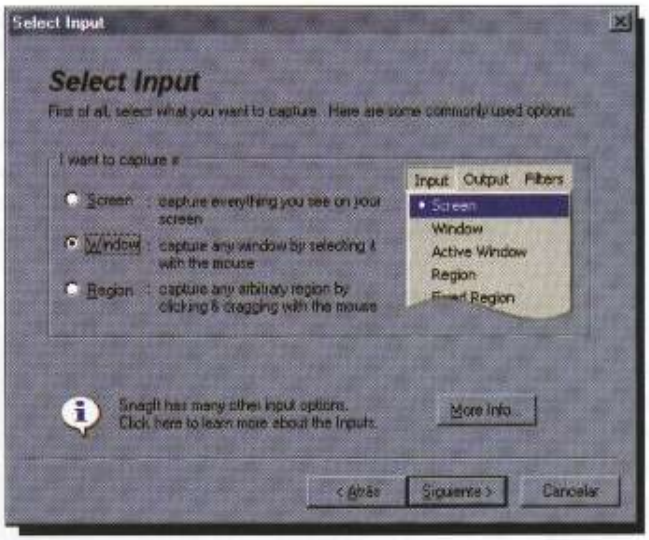


FIG. 5 Aquí puede definir qué zona de la pantalla se va a capturar. La opción Window que aparece en la lista es con mucho la más versátil.

FIG. 6 Con esta opción configurará TechSmith SnagIt para que guarde las capturas de pantallas como archivos de imágen.

FIG. 7 La combinación de teclas por omisión para iniciar el proceso de captura es Control + Mayús + P.

- **Catalog folder**: guarda la imagen en la carpeta de catálogo.

Una vez que haya elegido **Graphics file**, haga clic sobre **Siguiente**. Ahora podrá escoger la tecla o combinación de teclas que activará el proceso de captura. Por omisión, TechSmith SnagIt le muestra la combinación **<CTRL><SHIFT><P>**, es decir, la pulsación simultánea de las teclas **Control**, **Mayúsculas** y **P**.

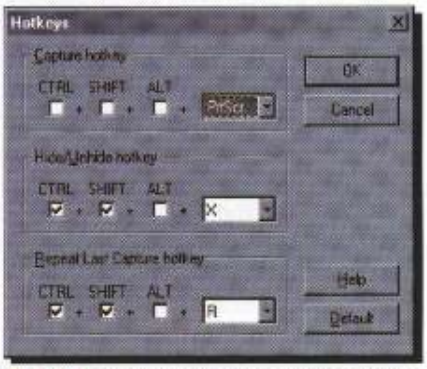


FIG. 8 Definir una nueva combinación de teclas es prácticamente un juego de niños.

Sin embargo, es posible que le sea más cómodo escoger otra combinación o incluso una única tecla: **Impr Pant.**

Para modificar la combinación de teclas actualmente seleccionada debe hacer clic sobre el botón **Change Hotkeys.**

De momento, la única combinación de las mostradas que le debe preocupar es la que se refleja en **Capture hotkey**; desactive **CTRL**, haga lo propio con **SHIFT**, abra el cuadro de lista desplegable que ahora muestra una **P** y, finalmente, elija **PrtScr** (la denominación en inglés de **Impr Pant**).

Antes de hacer clic sobre **OK** para aceptar los cambios, fíjese en que se puede definir prácticamente cualquier combinación de teclas que pase por su imaginación. Haga clic sobre **Siguiente**.

La pantalla que se muestra en estos momentos le permite configurar un apartado muy práctico de la aplicación. Una vez que se haya realizado la captura (por medio de la tecla o combinación de teclas elegida, en nuestro caso **Impr Pant**), TechSmith SnagIt ofrecerá una previsualización de la imagen obtenida, mostrándola en la pantalla para que el usuario pueda decidir si es correcta o no.

Si desea mantener esta característica de la aplicación, limítese a hacer clic sobre el botón **Siguiente**.

En caso contrario, active la opción **Off** para que la imagen se procese inmediatamente, sin visualización previa. En este ejemplo, no obstante, vamos a dejar activada la opción **On.**

FIG. 9 TechSmith SnagIt puede mostrar en la pantalla una previsualización de la captura de pantalla; de este modo el usuario podrá decidir si la captura es válida o no.

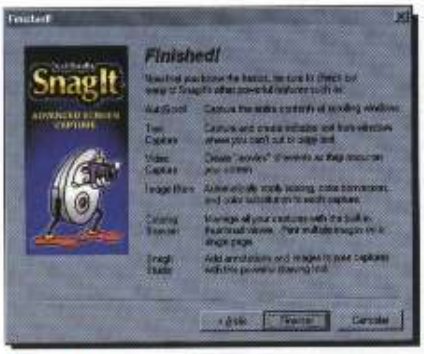


FIG. 10 TechSmith le recuerda en esta pantalla que la aplicación dispone de numerosas funciones que puede explorar. No deje de hacerlo.

FIG. 11 La interfaz de TechSmith SnagIt es sencilla, pero no tanto como para no mostrar la información de configuración.

FIG. 12 TechSmith SnagIt mostrará una previsualización de la imagen capturada.

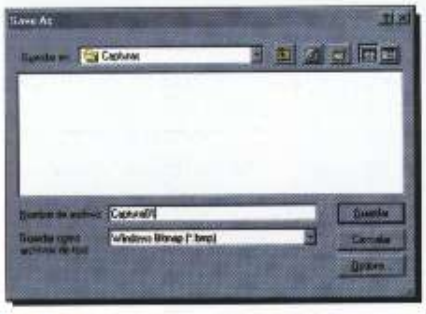


FIG. 13 Debe elegir la carpeta donde se guardará la imagen, así como su nombre.

Ha llegado al final del asistente. Esta pantalla le recuerda que no debe dejar de probar todas y cada una de las funciones de TechSmith SnagIt. Pulse sobre el botón **Finalizar** para dar por terminado el asistente de captura.

LA INTERFAZ

Al término del asistente, accederá a la interfaz de TechSmith SnagIt. Es bastante sencilla y prácticamente se limita a mostrar el tipo de acción a realizar (**Image Capture**, en la parte izquierda) y las opciones elegidas (**Input**, **Output**, **Filters** y **Options**, en la parte derecha). Además, como es habitual, en la parte superior encontrará la barra de menús y unos cuantos botones de acceso rápido a funciones.

Si repasa la configuración, comprobará que se va a realizar una captura de pantalla como imagen (**Image Capture**), que el objeto a capturar se va a elegir en el momento de la captura (**Window**), que se va a incluir el cursor en la imagen (**Include Cursor**), que la captura se guardará como una imagen con formato BMP en una unidad de disco (**File - BMP**) y que se debe previsualizar la captura (**[Preview]**).

Todo está preparado para comenzar a capturar. Para esta primera prueba vamos a capturar el escritorio de Windows. Minimice todas las aplicaciones abiertas excepto TechSmith SnagIt de la forma habitual y pulse la tecla **Impr Pant**. Verá que la aplicación se ha escondido y que el cursor se ha transformado en una mano de selección. Si la mueve hacia abajo, hacia la barra de tareas, podrá comprobar que el rectángulo sólo incluye este objeto.

Mueva el ratón de forma que el rectángulo de captura abarque el escritorio al completo y haga clic con el botón izquierdo del ratón. TechSmith SnagIt realizará inmediatamente la captura de pantalla y mostrará una previsualización.

Una vez que esté satisfecho con el resultado, haga clic sobre el botón **Finish** para dar por buena la imagen (o sobre **Cancel** para rechazarla). Tras escoger la carpeta donde se guardará la imagen y elegir su nombre, se habrá terminado el proceso de captura y el programa quedará a la espera de nuevas instrucciones.

EN DEFINITIVA...

Ha podido comprobar por sí mismo que realizar capturas con TechSmith SnagIt es un proceso sumamente sencillo y que puede ahorrarle un buen número de horas. No obstante, esta ha sido únicamente una primera toma de contacto con la aplicación. En la siguiente unidad avanzaremos más en el uso de esta magnífica herramienta.

TECHSMITH SNAGIT STUDIO

Cuando instaló TechSmith SnagIt, pudo comprobar que en su escritorio aparecieron dos accesos directos: uno a la propia aplicación y otro más llamado SnagIt Studio. Este último es un accesorio del programa principal que puede serle de gran utilidad si desea retocar las imágenes capturadas para incluir anotaciones, gráficos, resaltes, etc. Por ejemplo, con TechSmith SnagIt Studio puede recuadrar una determinada opción, resaltar en amarillo una parte de la imagen, escribir un texto y ligarlo a una zona de la captura mediante una flecha... Merece la pena echar un vistazo a esta estupenda aplicación, pues elimina la necesidad de disponer de un programa de retoque fotográfico (y, por supuesto, de comprarlo).

FIG. 14 Con este complemento podrá añadir información e incluso retocar las imágenes que haya capturado.

IBM WebSphere Studio (5)

IBM WebSphere Studio y Java

Desarrollado a principios de la década de los noventa, el lenguaje de programación Java se ha establecido como una referencia fundamental en el mundo de la programación, gracias sobre todo a su sencillez y a su independencia respecto a la plataforma en la que se ejecute. En esta unidad veremos cómo trabajar con Java dentro de IBM WebSphere Studio.

En el año 1991 un grupo de programadores de la empresa Sun Microsystems completaron un nuevo lenguaje de programación cuyas características más importantes se podrían definir como: simple, multiplataforma, orientado a objetos, portable, distribuido, interpretado, robusto, dinámico y seguro. De todas estas características, quizá la más importante sea la que indica su independencia de la plataforma en la que se está ejecutando. De esta forma se alcanzaba un viejo sueño de los programadores: conseguir que una aplicación escrita para un sistema determinado pudiera ser ejecutada en otro distinto sin tener que realizar cambios en el código. Esto es posible gracias a las librerías de Java, que son iguales con independencia de la plataforma en la que se esté ejecutando la aplicación, lo que permite que, por ejemplo, un programa escrito en un entorno Windows se pueda ejecutar sin ninguna clase de cambios en cualquier sistema Linux que soporte Java.

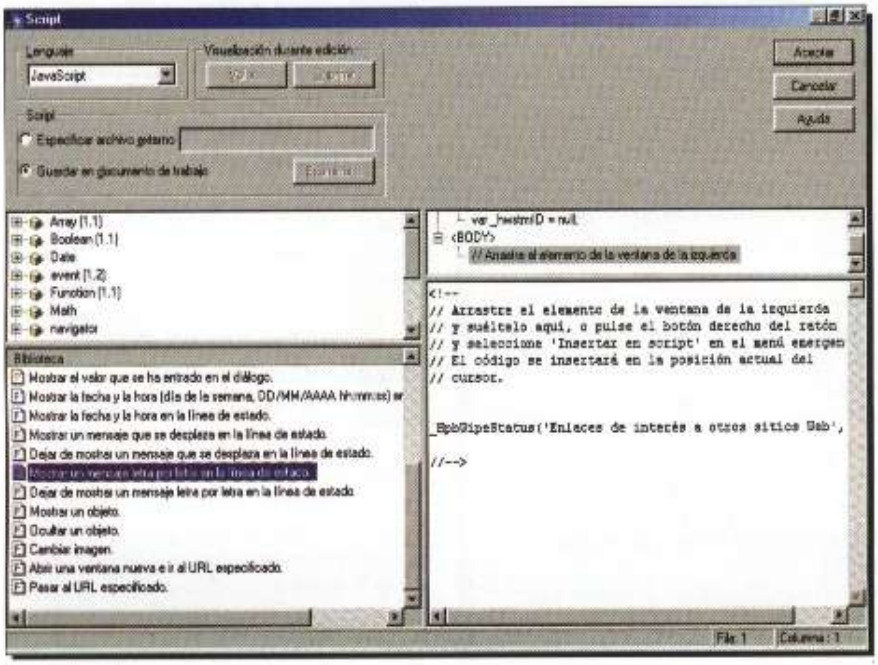


FIG. 1 Aspecto general de la ventana Script de IBM WebSphere Studio.

Esto, por supuesto, convirtió a Java en el lenguaje más apropiado para crear aplicaciones en Internet, dada la enorme variedad de entornos y plataformas diferentes que existen en la Red, aunque en los últimos años se ha forjado una buena reputación como un lenguaje de programación general, a la altura, incluso, de C++, con el que Java tiene muchas similitudes.

Otra característica importante de Java es su necesidad de ser interpretado en el momento de su ejecución. En la mayoría de los lenguajes de programación el código de la aplicación es compilado, de forma que ésta pueda ser ejecutada en el sistema en cuestión. En Java, no obstante, el primer paso consiste en compilar el código a un binario llamado *Bytecode* que se puede emplear en cualquier plataforma. Este binario será interpretado en el momento de su ejecución en la *máquina virtual de Java (Java VM)* de cada sistema, permitiendo de esta manera que cada entorno disponga de las librerías necesarias y adecuadas para el correcto funcionamiento de los programas creados, a la vez que se garantiza la portabilidad.

Visto todo esto no es de extrañar el gran alcance que Java ha tenido como el mejor medio para añadir una interactividad sin precedentes dentro de una página Web. Toda clase de efectos de animación, accesos a bases de da-

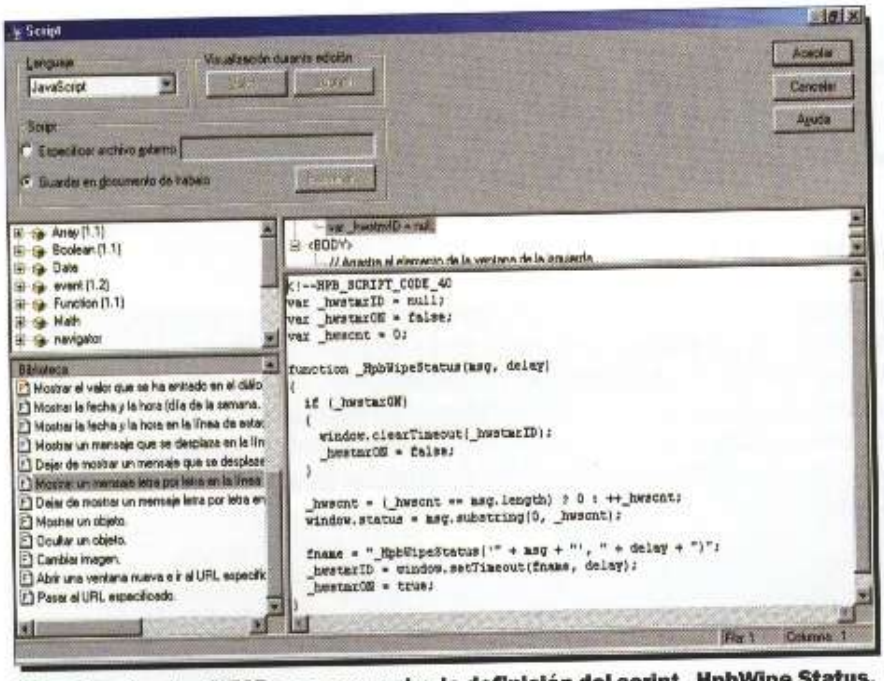


FIG. 2 Dentro de <HEAD> se encuentra la definición del script _HpbWipe Status.

tos, formularios e incluso juegos pueden ser implementados dentro de un navegador HTML que soporte dicha tecnología, haciendo que los sitios así creados ganen en cuanto a aspecto y eficacia en la navegación.

Como no podía ser menos, IBM WebSphere Studio soporta la tecnología Java, facilitando mediante ayudas y asistentes la creación de *scripts* de JavaScript, *Java Beans* y *applets* de Java. Como todas las demás herramientas de IBM WebSphere Studio, éstas se encuentran completamente integradas dentro del entorno del editor, por lo que irán apareciendo como opciones, ventanas y fichas adicionales según las vaya necesitando.

Antes de proseguir con lo que se puede hacer con Java dentro de IBM WebSphere Studio debe recordar que el hecho de no conocer todavía el lenguaje de programación Java o la creación de *scripts* o *applets* no será motivo para que no pueda usarlos, ya que la aplicación incluye una gran variedad de *scripts* preparados para su uso y que necesitan simplemente algunos parámetros para su correcto funcionamiento. Además, veremos que en Internet es posible encontrar toda clase de *applets* y otros programas en Java completamente gratis y que podrá añadir a su sitio sin excesiva complicación.

JAVA Y JAVASCRIPT

Es importante no confundir el lenguaje de programación Java con JavaScript, ya que, aunque ambos tienen un nombre similar y son parecidos en algunos aspectos, en otros muchos difieren notablemente. Básicamente, JavaScript es un lenguaje compacto basado en *scripts* o fragmentos de programa, que se usa en el desarrollo de aplicaciones en Internet. Recuerde que Java no es JavaScript: las maneras de trabajar de ambos son diferentes, ya que el segundo está basado en *scripts* y Java es un lenguaje de programación. JavaScript se parece a Java, pero no mantiene un estricto control de los datos como hace Java. No obstante, JavaScript soporta la mayor parte de las expresiones de sintaxis de Java y algunas de sus construcciones básicas de control de flujo de datos.

Frente a los *scripts*, Java puede crear *applets*. Un *applet* es un programa Java que puede ser incluido dentro de una página HTML, como si, por ejemplo, fuera una imagen. Usando un navegador compatible con Java, pues, se puede visualizar la página que contiene un *applet*, ya que el código del *applet* es transferido al sistema y ejecutado por dicho navegador. Así, un *applet* es compilado en el servidor y después interpretado en el navegador. Los programas en Java consisten exclusivamente de clases y métodos. Los requisitos de Java para la declaración de dichas clases y métodos y el tipo de seguridad usado hacen su programación mucho más compleja que con JavaScript.

DIFERENCIAS ENTRE JAVA Y JAVASCRIPT

En el siguiente cuadro puede observar las diferencias entre ambos lenguajes. Fíjese en que, aunque parecidos, ambos tienen características muy diferentes:

JavaScript	Java
Interpretado por el cliente.	**Compilado en el servidor y después interpretado en el cliente.**
Basado en objetos. Utiliza objetos incorporados, pero no clases o herencia.	**Orientado a objetos. Los *applets* consisten en clases objeto con herencia.**
Código integrado con HTML.	**Los *applets* son insertados como objetos dentro de las páginas HTML.**
Los tipos de datos no son declarados (pierden el tipo).	**Los tipos de datos pueden ser declarados (mantienen el tipo).**

JAVASCRIPT DESDE IBM WEBSPHERE STUDIO

En unidades anteriores vimos la forma de añadir un texto en la barra de estado del navegador que esté visualizando la página Web creada. En realidad esto se consiguió usando uno de los muchos *scripts* de Java que IBM WebSphere Studio incorpora. Para poder añadir uno de dichos *scripts* tendremos que abrir, desde la ventana principal de edición de la aplicación, el

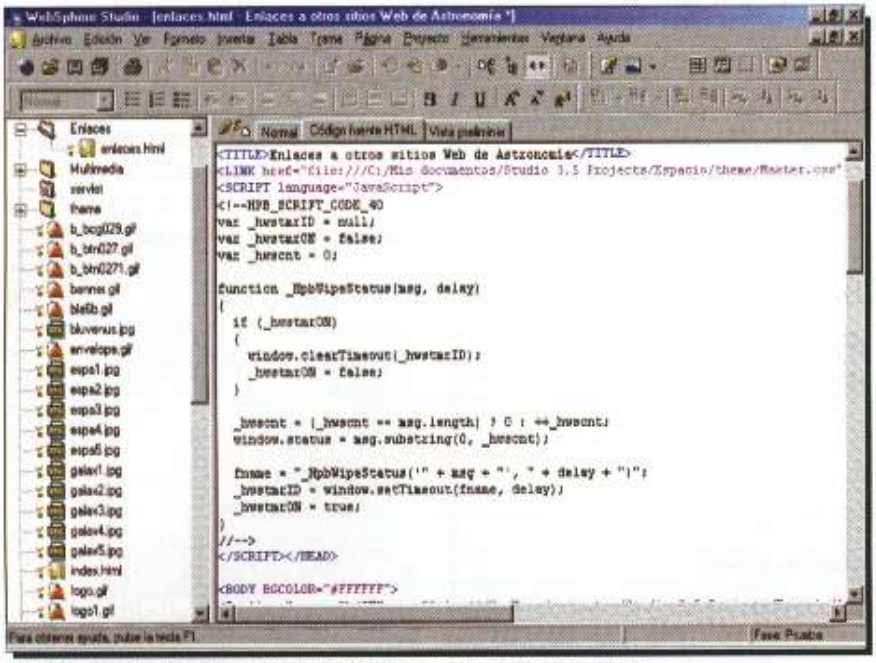


FIG. 3 El script se inserta directamente en el código HTML.

menú **Insertar**, para más tarde, desde el apartado **Otros**, seleccionar **Script...**. Aparecerá una nueva ventana desde la cual podrá crear *scripts*, utilizar los ya incluidos, o añadir otros que encuentre, por ejemplo, en Internet.

El proceso de trabajo desde la ventana **Script** es relativamente sencillo, sobre todo si ya tenía experiencia anterior con lenguajes de programación. La ventana está dividida en varias zonas donde aparecen el área de edición, la librería de *scripts*, las funciones del lenguaje y los botones y opciones de utilidades. Especialmente interesante es la sección denominada **Biblioteca**, la cual contiene *scripts* pregenerados para llevar a cabo acciones corrientes, como por ejemplo mostrar mensajes o animaciones de texto en la línea de estado, establecer un color de fondo, o incluso detectar el tipo de navegador que estemos usando, ya sea Internet Explorer o Netscape. Para añadir cualquiera de estos *scripts* simplemente debe señalar con el botón izquierdo del ratón el *script* deseado dentro de la lista **Biblioteca**, para después arrastrarlo hacia el área en blanco de la parte derecha de la pantalla. Observe que el *script* se insertará desde la posición actual del cursor de edición de texto de esa zona de la pantalla, por lo que habrá de tener esto muy en cuenta a la hora de concatenar *scripts* (por ejemplo, en el caso de un *script* que realice operaciones distintas en función de los distintos resultados de otro *script* anterior).

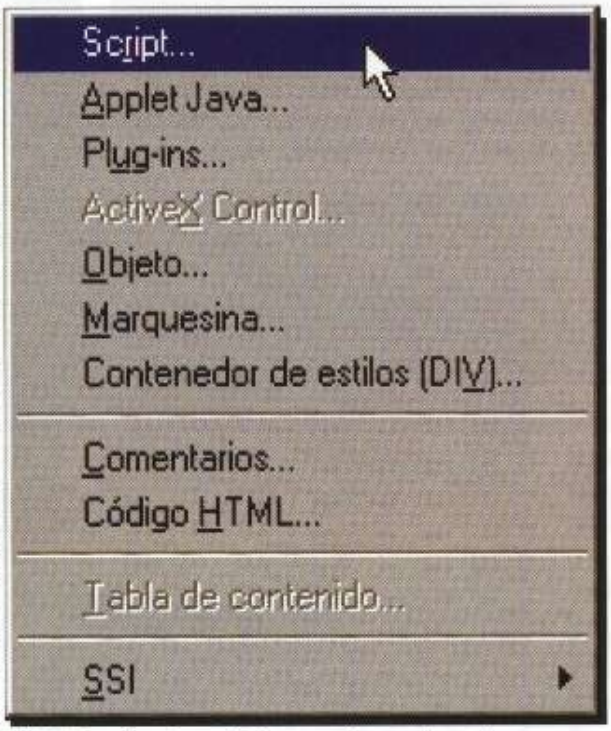


FIG. 4 Desde el menú Otros podremos insertar o crear nuevos scripts.

Otro aspecto muy a tener en cuenta a la hora de añadir *scripts* es el lenguaje de programación y la versión que deseamos para su creación. Desde la ventana **Script**, y en el apartado **Lenguaje**, encontrará una lista desplegable. Ábrala y encontrará una serie de opciones, básicamente diferentes versiones y revisiones de los lenguajes, como **JavaScript**, **VBScript** o **JSP Expression**. Elija la que más convenga para las necesidades de su futuro sitio Web y el código que contendrá. Generalmente, el valor por omisión, **JavaScript**, es válido para la mayoría de las situaciones, y no le dará problemas a la hora de insertar los *scripts* del apartado **Biblioteca**.

No obstante, y dependiendo de la clase de código introducido (por ejemplo, si encuentra una función interesante en Internet), debe tener muy clara la versión concreta del lenguaje que va a insertar, y sobre todo si los navegadores que usarán los futuros visitantes del sitio Web tendrán la capacidad de interpretar el código en cuestión. Tras comprobar que el valor de la lista desplegable permanece como **JavaScript** haga clic sobre el botón **Aceptar** para volver a la ventana principal del editor de IBM WebSphere Studio.

Para probar el funcionamiento de los *scripts* en nuestro sitio Web vamos a añadir algunos de los in-

FIG. 5 El efecto del script se verá en la barra de estado del navegador.

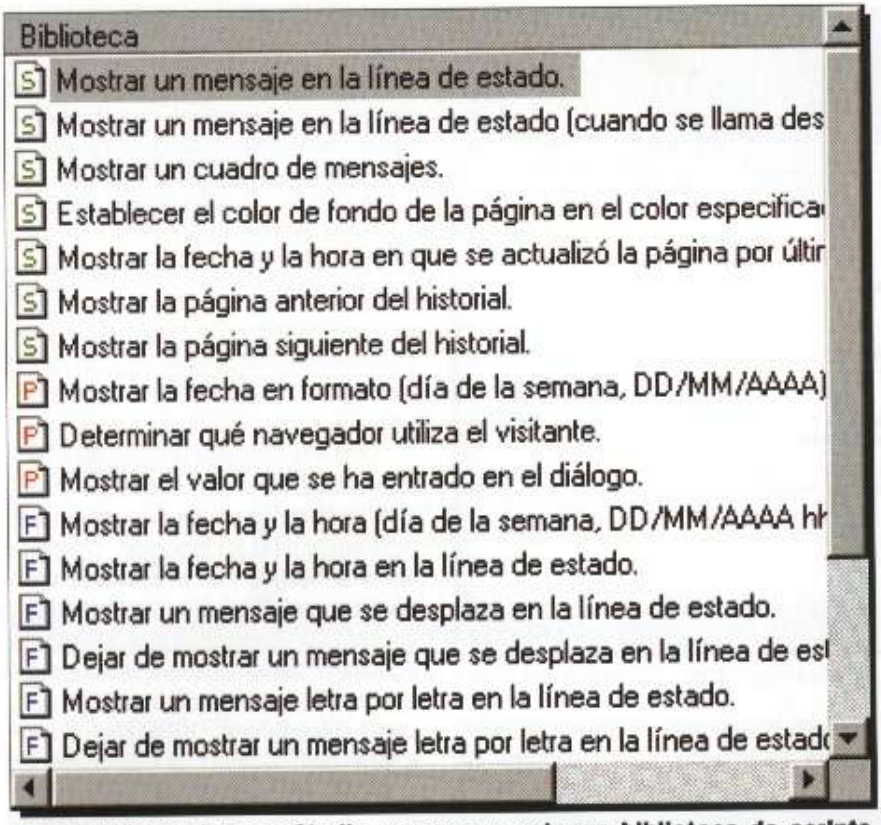


FIG. 6 IBM WebSphere Studio posee una extensa biblioteca de scripts pregenerados.

cluidos en la aplicación para darle un aspecto más profesional a nuestras páginas. Desde la ventana principal del editor haga clic en el archivo **enlaces.html**, situado dentro de la carpeta **Enlaces**, pase a modo de edición **Normal** si no lo estaba ya y vuelva a abrir el menú **Insertar**, para, dentro del apartado **Otros**, seleccionar la opción **Script...**. Le recomendamos que maximice esta ventana para su mayor comodidad y que amplíe el espacio para la sección **Biblioteca** haciendo clic y luego arrastrando hacia la derecha la barra de separación vertical que aparece en la parte central de la pantalla.

En este momento vamos a proceder a seleccionar el *script* deseado y a insertarlo en el documento.

Observe la lista de *scripts* que aparece a la izquierda de la pantalla y seleccione, de entre los marcados con el icono de una letra *F* azul (*función*), el que muestra el texto **Mostrar un mensaje letra por letra en la línea de estado**.

Ahora observe la parte derecha de la imagen.

Comprobará que aparece un texto que comienza por la etiqueta **<!—**, seguido por líneas que incluyen //, lo cual indica que se trata de un comentario.

Tras éste, y antes de la línea que muestra //—>, deben aparecer algunas líneas en blanco. Haga clic sobre cualquiera de ellas para situar el cursor en dicho lugar. Es precisamente ahí donde, a continuación, y tras hacer clic en el *script* que seleccionamos anteriormente, debe arrastrar y soltar el botón del ratón. Aparecerá la ventana **Establecer parámetros** donde, dentro del apartado **Parámetro**, debe introducir el mensaje que desea que aparezca en la línea de estado dentro de la caja **Mensaje**.

Marque el texto que aparece con el ratón e introduzca *Enlaces de interés a otros sitios Web*. En **Intervalo de visualización** podrá especificar el tiempo, en milisegundos, que tardarán los caracteres en aparecer en la pantalla. Usando las pequeñas flechas que aparecen junto a la caja, o marcando el texto, cambie el valor a *200*. Cuando haya finalizado, haga clic sobre el botón **Aceptar**. Tras volver a la ventana **Script**

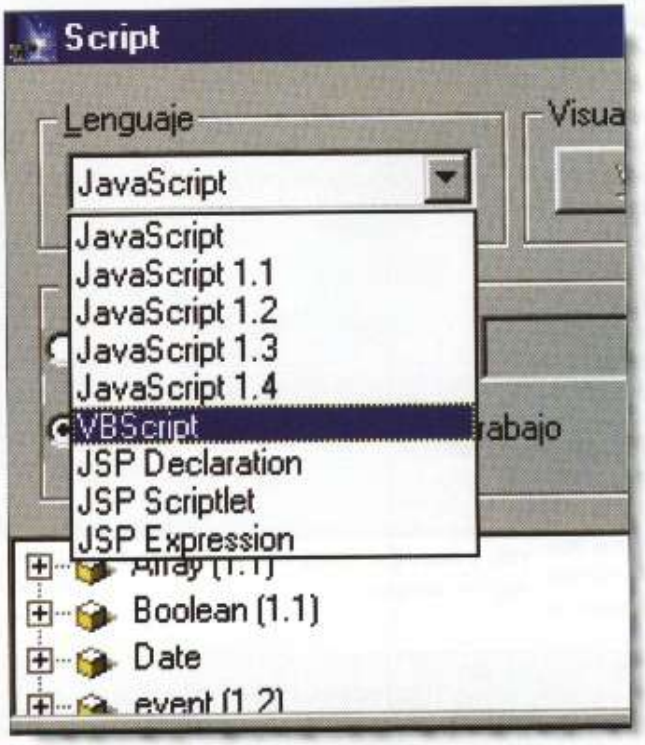


FIG. 7 Es importante tener en cuenta el lenguaje de programación que queramos usar en los scripts.

APPLETS Y JAVASCRIPT DESDE HTML

Desde el modo de edición Código fuente HTML podemos comprobar las etiquetas usadas para insertar *applets* y JavaScript en nuestros sitios Web. En el primer caso se utilizará la etiqueta <APP>, cuya sintaxis es <APP Class="Nombre de clase">. Para los *scripts*, la etiqueta respectiva será <SCRIPT>, seguida por el programa, que debe finalizar con </SCRIPT>.

observará que en el área derecha de ésta han aparecido nuevas líneas de código fuente. Si en la parte superior de esta zona hace clic sobre la línea **var _hwstmrID = null;** observará que aparece el siguiente código fuente JavaScript:

```
<!—HPB_SCRIPT_CODE_40
var _hwstmrID = null;
var _hwstmrON = false;
var _hwscnt = 0;

function _HpbWipeStatus(msg,
delay)
{
 if (_hwstmrON)
 {
 window.clearTimeout(_hwstmrID);
  _hwstmrON = false;
 }

 _hwscnt = (_hwscnt ==
msg.length) ? 0 : ++_hwscnt;
 window.status =
msg.substring(0, _hwscnt);

 fname = "_HpbWipeStatus('"
+ msg + "', " + delay + ")";
 _hwstmrID =
window.setTimeout(fname,
delay);
 _hwstmrON = true;
}
//—>
```

Observe que en primer lugar se definen las variables que se usarán en la función **_HpbWipeStatus**, para más tarde proseguir con el código fuente de la función.

Haga clic sobre el botón **Aceptar** para volver al editor de IBM WebSphere Studio. Asegúrese ahora de que se encuentra en modo de vista **Normal** y observará que en el lugar en que esté ubicado el cursor habrá aparecido un pequeño icono representando una letra S con fondo amarillo. No se preocupe por el diseño de la página, ya que dicho icono no aparecerá en la versión publicada y simplemente sirve para indicarle que se ha insertado una función *JavaScript* en el documento. Haga clic ahora sobre la pestaña **Código fuente HTML** y vaya a la parte superior del documento. Observe que aparece la etiqueta:

```
<SCRIPT
language="JavaScript">
```

Esta etiqueta indica que se ha insertado una función en lenguaje JavaScript. Fíjese en que, tras ella, aparece el código de la función, para acabar con la etiqueta **</SCRIPT>**. Desde ese momento el navegador sabrá que existe una función que tendrá que ser utilizada, pero no la interpretará hasta que no encuentre una llamada, que puede ver más adelante en el código en la línea:

```
_HpbWipeStatus('Enlaces de
interés a otros sitios Web',
200);
```

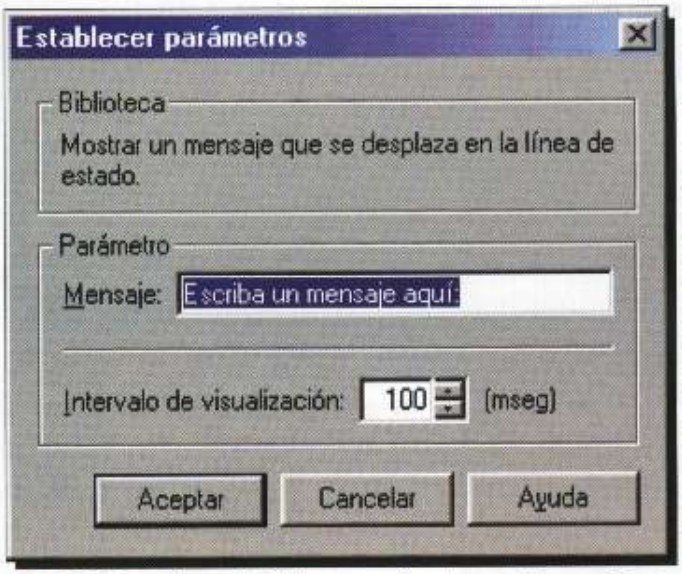


FIG. 8 En la ventana Establecer parámetros podremos introducir el texto deseado para el mensaje de la barra de estado.

Sólo en este momento el navegador (por supuesto, si soporta Java) interpretará la función, usando los parámetros **msg** y **delay**. Como es fácil adivinar, no habrá que volver a la ventana **Script** para cambiar el texto del mensaje o el tiempo que tarde en visualizarse; simplemente habrá que cambiarlos desde el modo **Código fuente HTML**,

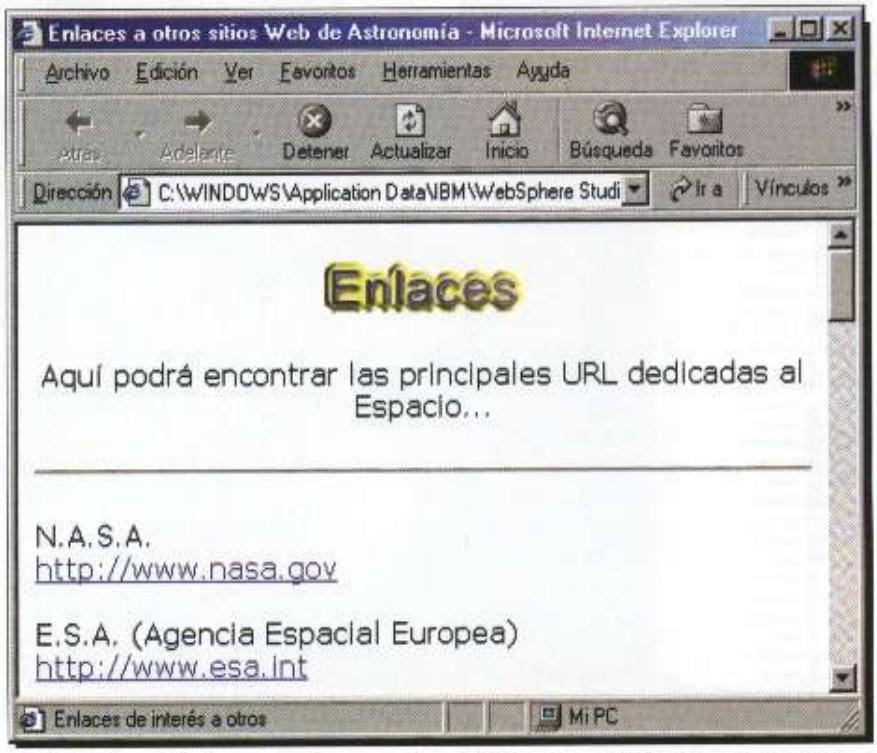


FIG. 9 Es importante comprobar que el script funciona en distintos navegadores.

```
Especifique cómo descubrió la Web &quot;El Espacio&quot;:<BR>
<BR>
<SELECT size="2">
  <OPTION>Buscador de Internet</OPTION>
  <OPTION>Enlace desde otra página</OPTION>
  <OPTION>Recomendación de un amigo</OPTION>
  <OPTION>Revista especializada</OPTION>
  <OPTION selected>Por casualidad</OPTION>
</SELECT><BR>
<BR>
<INPUT type="submit" name="SUB" value="Enviar"> <INPUT type="reset" name="RES" valu
</DIV>
<SCRIPT language="JavaScript">
<!--
// Arrastre el elemento de la ventana de la izquierda
// y suéltelo aquí, o pulse el botón derecho del ratón
// y seleccione 'Insertar en script' en el menú emergente.
// El código se insertará en la posición actual del
// cursor.


_HpbWipeStatus('Enlaces de interés a otros sitios Web', 200);

//-->
</SCRIPT></BODY>
</HTML>
```

FIG. 10 La llamada al script que hemos añadido se realiza al final del documento.

teniendo cuidado en respetar el resto del código de la función, especialmente las comillas simples del parámetro **msg**.

En este momento puede proceder a probar la página con el *script* insertado. Para ello, en primer lugar haga clic sobre la pestaña **Vista preliminar** y observe como, al visualizarse la página, en la barra de estado situada en la parte superior de ésta va apareciendo el texto introducido en el parámetro **msg** de la función **_HpbWipeStatus**, carácter a carácter y con una velocidad en milisegundos definida por el parámetro **delay**.

Para comprobar que, efectivamente, puede visualizar el resultado en un navegador, abra el menú **Herramientas** y, dentro del apartado **Navegador WWW,** seleccione el que desee para hacer la prueba; en general **Internet Explorer** funcionará en la mayoría de los equipos. Se lanzará una ventana nueva del navegador seleccionado, en la que debe buscar la barra de estado y comprobar que el *script* funciona sin problemas (en *Internet Explorer*, la barra de estado se encuentra en la parte inferior izquierda de la ventana). Cierre esta ventana y vuelva al modo **Normal** de edición de IBM WebSphere Studio, donde no debe olvidar guardar los cambios efectuados.

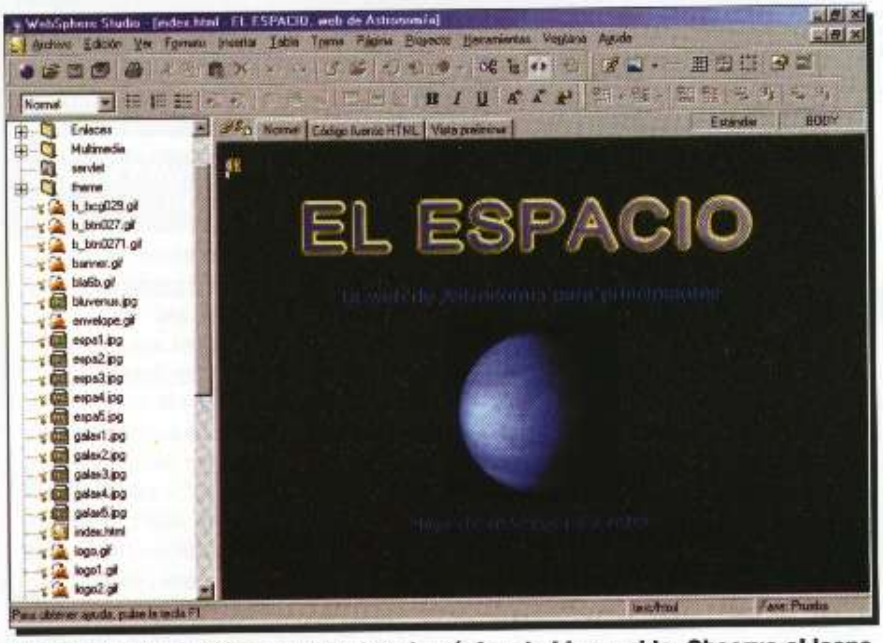


FIG. 11 Vamos a añadir un script en la página de bienvenida. Observe el icono con la "S" amarilla en la parte superior de ésta.

MÁS SCRIPTS

Como ya conocemos el procedimiento para añadir *scripts* en nuestras páginas, vamos a aprovechar algunos de los incluidos en IBM WebSphere Studio para realzar algunas de las secciones de nuestro sitio Web de ejemplo. En primer lugar cierre todas las ventanas de edición que pudiera tener abiertas dentro de la aplicación y abra el archivo **index.html**. Abra el menú **Insertar**, despliegue el apartado **Otros** y haga clic sobre la opción **Script...** para mostrar la ventana que ya conoce. Ahora, desde la sección **Biblioteca**, seleccione y arrastre a la parte derecha y en blanco de la ventana el objeto **Mostrar un cuadro de mensajes**, representado además por una letra *S* verde. Al soltar el botón del ratón volverá a aparecer la ventana **Establecer parámetros**. Dentro de la caja de texto **Mensaje** puede introducir el texto que desea que aparezca en el cuadro de mensajes. Introduzca *¡Bienvenidos a El Espacio!* en dicha caja y haga clic sobre el botón **Aceptar**. Recuerde que para introducir el mensaje, antes debe borrar el texto que aparece en la caja, o bien resaltar dicho texto antes de escribir el nuevo.

Observe que en este momento ha aparecido una nueva línea con la llamada al *script*. Sin embargo, si hace clic sobre **<HEAD>**, podrá comprobar la diferencia con la función que insertó anteriormente. Efectivamente, en este caso no se define la función para que el navegador pueda interpretarla, sino que se la llama directamente, en este caso el *script* en cuestión es **alert:**

```
alert('¡Bienvenidos a El Espacio!');
```

Haga clic ahora sobre el botón **Aceptar** para volver al editor de IBM

FIG. 12 Parámetros del cuadro de mensaje a insertar.

WebSphere Studio. Observe que, una vez más, ha aparecido en la pantalla un pequeño símbolo amarillo con el carácter *S* que representa la inclusión de un *script*. Haga clic sobre la pestaña **Vista preliminar** y compruebe como, efectivamente, antes de mostrarse la página aparece un pequeño cuadro de mensaje en una ventana independiente a modo de bienvenida. Así, los futuros visitantes del sitio Web únicamente tendrán que hacer clic sobre el botón **Aceptar** para entrar. También es importante destacar que dicho mensaje aparecerá independientemente del navegador que usemos (siempre, claro está, que soporte *Java*), por lo que, por ejemplo, un navegador *HotJava* funcionando bajo un sistema *Linux* mostrará también dicho mensaje, aunque adaptado a las características del entorno de ventanas usado en dicho momento (*Gnome*, *KDE*, *Enlightment* ...).

FIG. 13 Cuadro de mensaje creado mediante una sencilla función JavaScript.

Guarde los cambios ahora usando el botón **Guardar todo** y, si lo desea, compruebe que la página funciona en su navegador usando el procedimiento que ya conoce. Cierre el archivo **index.html** y abra **main.html**.

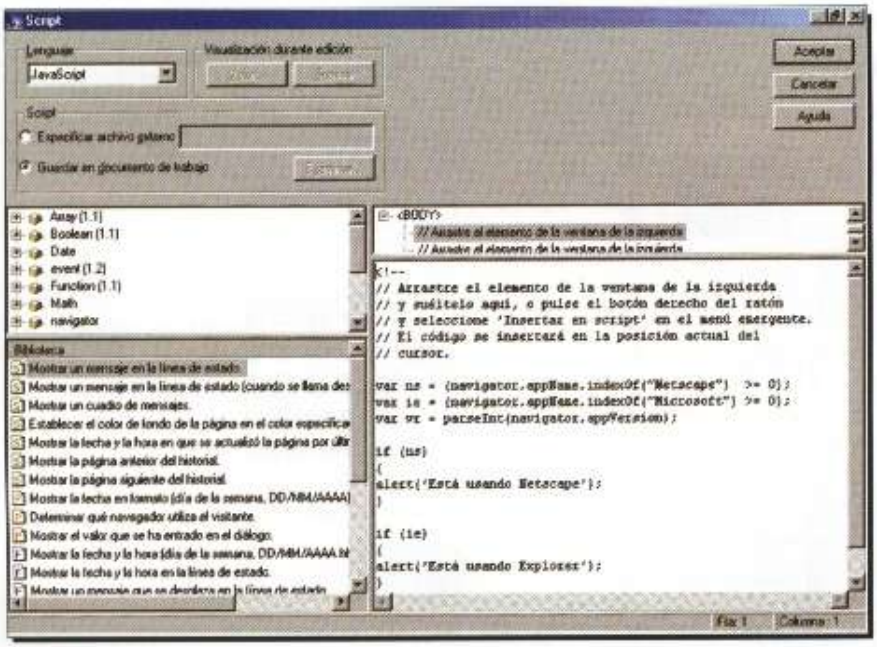


FIG. 14 Desde Script insertaremos una pequeña función que detectará el navegador usado por el visitante del sitio Web.

Vuelva a abrir la ventana **Script** y seleccione, en **Biblioteca**, la opción **Determinar qué navegador utiliza el visitante**, que se encuentra entre los elementos que muestran como icono una letra *P* roja.

Como ya hizo anteriormente, arrastre dicho elemento a la zona derecha de la pantalla. Observe que en esta ocasión no ha aparecido ninguna ventana pidiendo parámetros, sino que directamente se ha introducido el siguiente código:

```
var ns =
(navigator.appName.indexOf
("Netscape") >= 0);
var ie =
(navigator.appName.indexOf
("Microsoft") >= 0);
var vr =
parseInt(navigator.appVersion);

if (ns)
{

}

if (ie)
{

}
```

Este código por sí mismo no realiza ninguna acción, simplemente reserva dos áreas para introducir nuevos *scripts* que realicen acciones cuando se detecten los diferentes navegadores del sistema, *ns* en caso de *Netscape* e *ie* en caso de *Internet Explorer*.

Las acciones a llevar a cabo deberán ser definidas dentro de las llaves respectivas, esto es, entre los símbolos { y }.

Se podrían llevar a cabo todo tipo de acciones, como mostrar mensajes, cambiar las características de la ventana, etc., aunque en nuestro caso, y puesto que ya conocemos el *script* **alert**, el sistema mostrará una ventana mostrando un mensaje al visitante con el tipo de navegador que esté utilizando. Sitúe el cursor en la primera sección, la correspondiente a *ns*, justo entre los símbolos { y }, y escriba el siguiente *script*:

```
alert('Está usando Netscape');
```

A continuación, en la sección correspondiente a *ie*, escriba:

```
alert('Está usando Explorer');
```

Una vez introducidas estas dos líneas, el código final debe quedar así:

```
var ns =
(navigator.appName.indexOf
("Netscape") >= 0);
var ie =
(navigator.appName.indexOf
("Microsoft") >= 0);
var vr =
parseInt(navigator.appVersion);

if (ns)
{
alert('Está usando Netscape');
}

if (ie)
{
alert('Está usando Explorer');
}
```

Haga clic sobre el botón **Aceptar** para volver a la ventana principal de IBM WebSphere Studio y pruebe la página. Para ello, abra el menú **Herramientas**, seleccione el apartado **Navegador WWW** y finalmente haga clic sobre el elemento de su elección, por ejemplo **Internet Explorer**. Al lanzarse el navegador en una nueva ventana se interpretará el *script* introducido y se mostrará un mensaje u otro dependiendo del resultado de éste. Al volver a la aplicación no olvide guardar los cambios.

Para finalizar vamos a insertar una línea que informe al usuario acerca de la última vez que se actualizó el sitio. Para ello, sitúe el cursor en la última línea del documento y vuelva a abrir la ventana **Scripts**. Localice en **Biblioteca** el objeto **Mostrar la fecha y la hora en que se actualizó la página por última vez** y arrástrelo, como ya sabe, hacia la derecha. Aparecerá la siguiente línea de código:

```
document.write('Last updated:
', document.lastModified);
```

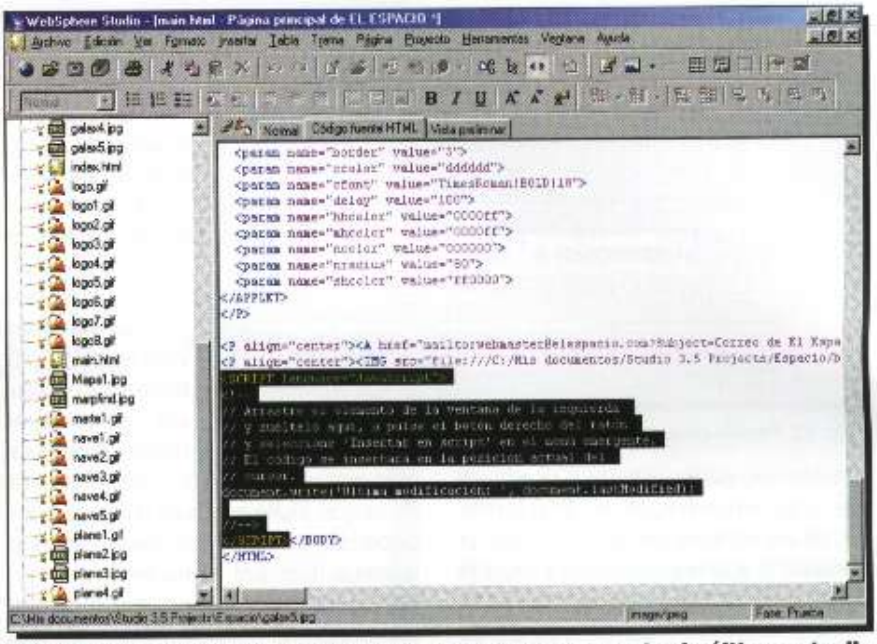


FIG. 15 Aspecto del código del script que detecta y muestra la última actualización de la página actual.

Cambie el texto *Last updated:* por *Última modificación:* , recordando respetar las comillas simples que aparecen y dejando un espacio en blanco al final del texto. Haga clic sobre **Aceptar** y volverá a la ventana principal de IBM WebSphere Studio, donde puede probar desde **Vista preliminar** que aparece una línea al final del documento indicando la fecha de la última modificación del documento en cuestión.

APPLETS DE JAVA

Como dijimos al principio, un *applet* es un programa escrito en Java que se puede incluir dentro de una página HTML como otro objeto cualquiera, por ejemplo como una imagen. Como ejemplo vamos a insertar en la página principal de *El Espacio* un reloj analógico, que al estar escrito en Java y no usar ninguna imagen añadida no dará problemas al descargar la página. En pri-

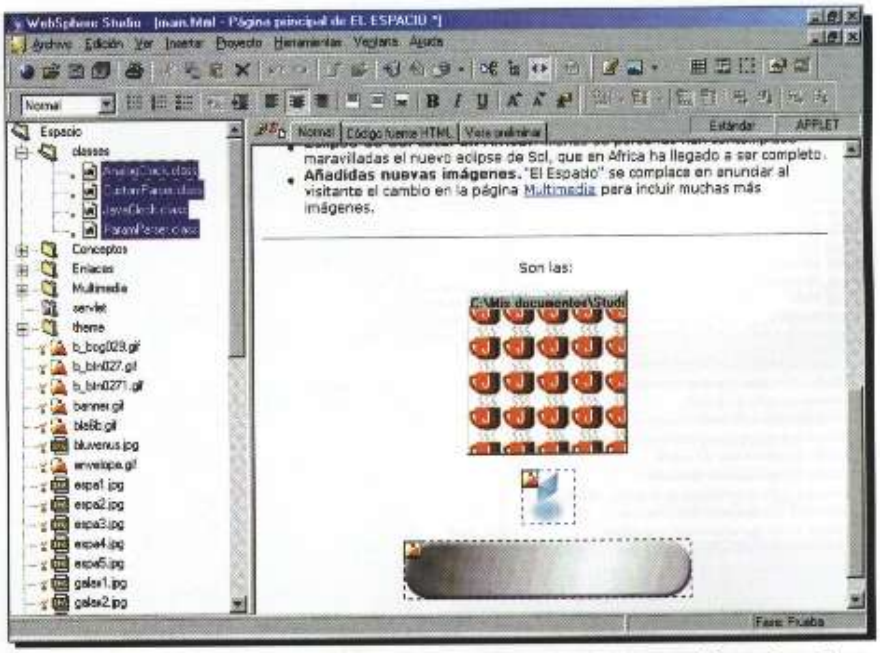


FIG. 16 Para crear el applet del reloj son necesarias 4 clases diferentes. Asegúrese de ubicarlas en la carpeta classes.

mer lugar vamos a crear una carpeta dentro de la estructura de nuestro sitio para que contenga el *applet*. Haga clic con el botón derecho en la raíz del árbol de carpetas de nuestro sitio, esto es, **Espacio**, situado en la parte izquierda del editor de la aplicación. En el menú contextual que aparece, seleccione la opción **Insertar** y haga clic a continuación sobre **Carpeta...**. En la ventana **Insertar carpeta** introduzca como nombre *classes* y haga clic sobre el botón **Aceptar**.

Al volver al editor podrá comprobar que, efectivamente, ha aparecido una nueva carpeta con dicho nombre en la estructura. Haga clic con el botón derecho del ratón sobre ella, seleccione **Insertar** de nuevo, y a continuación, **Archivo...**. En la ventana que aparece pase a la ficha **Utilizar existente** y haga clic sobre el botón **Examinar...** para mostrar la ventana **Abrir**. Seleccione el archivo **JavaClock.class** situado dentro de la carpeta **Ejemplos\Java** del CD incluido con esta entrega y haga clic sobre **Abrir** y más tarde sobre **Aceptar** para terminar su inserción. Siguiendo el mismo procedimiento, inserte en la misma carpeta los archivos **AnalogClock.class**, **CustomParser.class** y **ParamParser.class**. Abra el archivo **main.html** y, en modo **Normal**, sitúe el cursor encima del enlace a correo electrónico que se insertó al final de la página (el que representa un sobre animado). Abra el menú **Insertar**, seleccione el apartado **Otros** y haga clic sobre **Applet Java...**.

En la ventana **Atributos**, que ya conoce, aparecerán una serie de opciones nuevas relacionadas con el *applet* que pretende insertar. En **Código** debe escribir *JavaClock.class*, mientras que en **Base de código** introducirá, a su vez, *classes*. Vaya, en la parte inferior, al apartado **Tamaño** e introduzca el valor *150* en los parámetros **Anchura** y **Altura**. Haga clic ahora sobre el botón **Aceptar** para volver a la pantalla principal de IBM WebSphere Studio.

Seleccione el *applet* insertado (lo reconocerá por estar representado por una serie de tazas de café, símbolo del lenguaje *Java*) y céntrelo haciendo clic sobre el botón **Alinear al centro**.

Como parece que el reloj está ahí sin motivo, suba el cursor de edición encima del *applet* y teclee *Son las:*, centrando también el texto y haciéndolo un poco más pequeño haciendo clic sobre el botón **Reducir tamaño del font**. Guarde los cambios y haga clic sobre **Vista preliminar** una vez más para comprobar que, efectivamente, en el área en que aparecían los iconos en forma de taza de café figura ahora un reloj analógico mostrando la hora del sistema. Si pasa a la ficha **Código fuente HTML** podrá comprobar que se ha añadido la siguiente línea al código de la página, conteniendo la llamada al *applet*:

```
<P align="center"><APPLET
code="JavaClock.class"
codebase="classes"
width="150" height="150"></
APPLET></P>
```

Observe que en *code* se detalla la *clase*, esto es, el programa Java que se va a insertar, mientras que *codebase* representa el lugar donde buscar dicha *clase*.

En caso de no encontrar el parámetro *codebase* en la etiqueta *<APPLET>*, el navegador buscaría la clase en la misma carpeta en que se encuentre el archivo HTML.

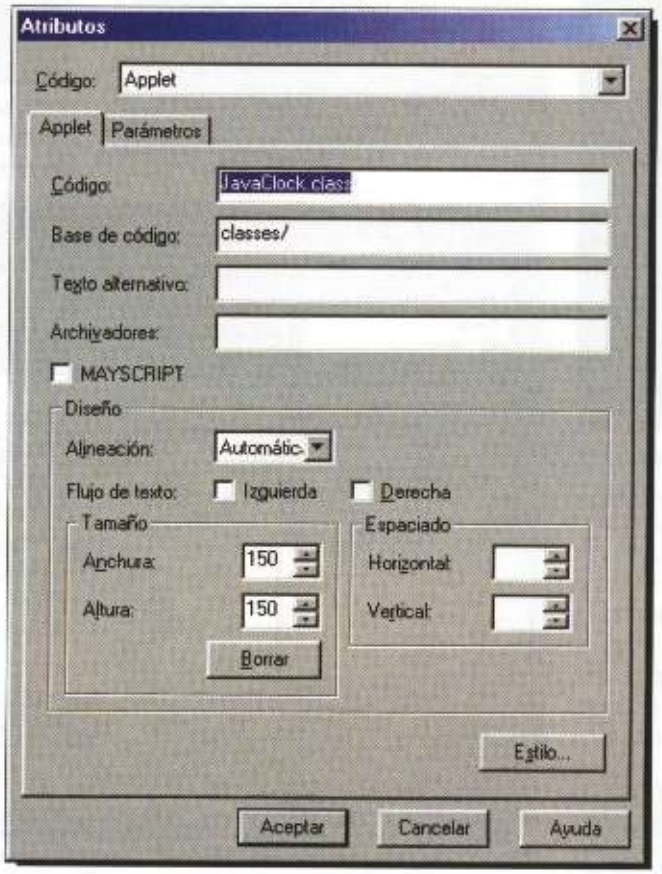


FIG. 17 Ventana de atributos del applet.

IBM WEBSPHERE HOMEPAGE BUILDER

En el CD-ROM adjunto a esta unidad se entrega una versión *trial* o de prueba durante 60 días de IBM WebSphere HomePage Builder V5.0. Como ya hemos visto, IBM WebSphere Studio es una completísima herramienta de diseño Web profesional, pero en muchas ocasiones los buenos resultados vienen dados por una gran cantidad de pasos y una concienzuda programación. Por ello, cuando se quieren resultados fáciles y rápidos, conviene usar programas como HomePage Builder.

PASO DE PARÁMETROS A UN APPLET

Los *applets* pueden ser configurados por medio de parámetros, caso de que se hayan definido para el *applet* en cuestión. Si no los hemos creado nosotros mismos, es conveniente disponer de una lista de los parámetros del *applet*, de forma que podamos cambiar rápidamente su funcionalidad. En el ejemplo anterior nos hemos limitado a insertar el reloj; no obstante, ahora lo modificaremos pasándole parámetros al *applet* que lo genera. Aunque IBM WebSphere Studio dispone de un sistema para crear parámetros desde la ventana *Atributos*, nosotros procede-

remos a añadirlos directamente en el modo *Código fuente HTML*. Vaya a esta ficha y localice la línea que incluye al *applet*. Le recomendamos que, por comodidad de lectura, sitúe cada etiqueta en una línea propia, dejando una línea en blanco delante y detrás del grupo, por ejemplo de este modo:

```
<P align="center">
<APPLET
code="JavaClock.class"
codebase="classes"
width="150" height="150">

</APPLET>
</P>
```

Escriba ahora, entre las etiquetas <APPLET> y </APPLET>, las siguientes líneas de código:

```
<param name="bgcolor"
value="ffffff">
<param name="border"
value="5">
<param name="ccolor"
value="dddddd">
<param name="cfont"
value="TimesRoman|BOLD|18">
<param name="delay"
value="100">
<param name="hhcolor"
value="0000ff">
<param name="mhcolor"
value="0000ff">
<param name="ncolor"
value="000000">
<param name="nradius"
value="80">
<param name="shcolor"
value="ff0000">
```

Observe que la etiqueta <*PARAM*> permite pasar un parámetro al *applet*, definido por su nombre (*name*) y por su valor (*value*).

Guarde los cambios y pase a modo **Vista preliminar** para comprobar que los cambios han surtido efecto y que el *applet* ha cambiado de aspecto. Experimente cambiando los valores de cada parámetro y comprobando los cambios que se producen.

Si abre ahora la ventana de atributos del *applet* (por ejemplo, seleccionando el *applet* y haciendo clic con el botón derecho del ratón sobre él para más tarde hacer clic sobre la opción **Atributos...**) y pasa a la ficha **Parámetros** observará que han aparecido los mismos parámetros que se introdujeron directamente en el código fuente. Si desea cambiar alguno de ellos, selecciónelo y use la casilla **Valor**. Haga clic sobre **Aceptar** para salir al editor. Una vez más, no olvide guardar los cambios hechos al documento.

PUBLICACIÓN PARA PRUEBA

Una vez completados los pasos recogidos en esta unidad, y tras haber grabado los cambios (recuerde que, como vimos en unidades anteriores, los archivos deben estar reservados para poder ser guardados), hay que proceder una vez más a la publicación de prueba del sitio Web en el disco duro.

Al publicar, IBM WebSphere Studio analiza todos los enlaces y vínculos del sitio, a la vez que ubica cada archivo y carpeta en su lugar correspondiente dentro de la estructura que hayamos ido creando.

Cierre ahora todas las ventanas de edición que pudiera tener abiertas y haga clic sobre el icono **Vista de publicación** para ver la estructura final que será creada por el sistema.

Haga clic con el botón derecho del ratón sobre **localhost** y seleccione la opción **Propiedades** en el menú contextual.

En la ventana **Propiedades de localhost** localice la lista desplegable **Destino de publicación** y asegúrese de que la opción **Espacio** se encuentra seleccionada.

En caso contrario, proceda a crear un directorio y a asignarlo al proyecto como se vio en la unidad anterior. Asegúrese también de que la opción **Publicación de sistema de archivos:** se

FIG. 18 Los parámetros del applet podrán ser modificados también desde la ventana Atributos.

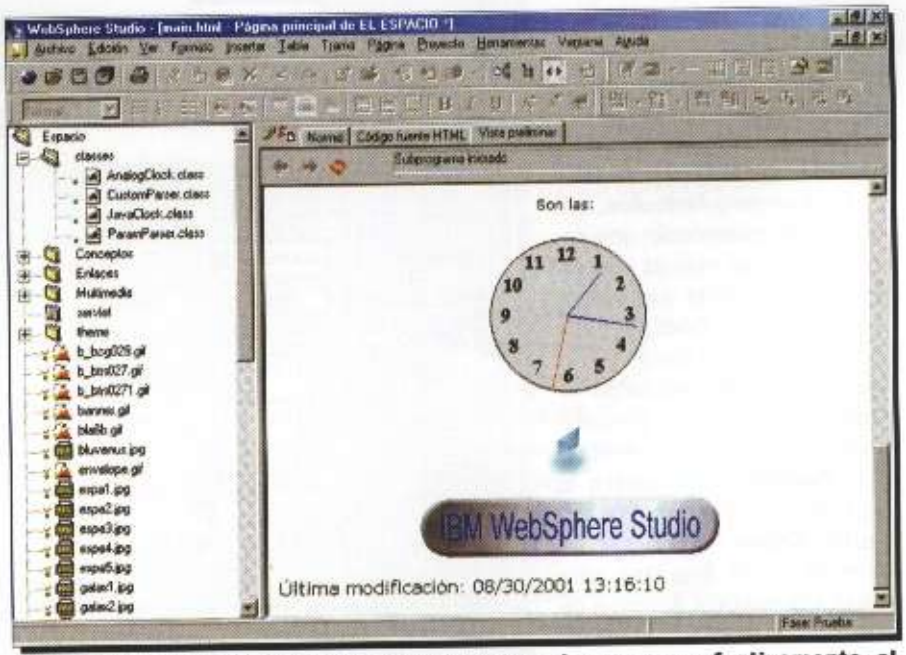


FIG. 19 Al pasar a modo Vista preliminar comprobamos que, efectivamente, el script de la última modificación y el applet de reloj funcionan.

FIG. 20 El aspecto de HomePage Builder es muy similar al de IBM WebSphere Studio.

encuentra seleccionada, y de que **Windows** aparece en la lista desplegable anexa.

Haga clic sobre el botón **Aceptar** cuando haya terminado. Vuelva a hacer clic con el botón derecho del ratón sobre **localhost** en la **Vista de publicación** y seleccione esta vez la opción **Publicar este servidor**. En la ventana **Archivos a publicar** preste especial cuidado a que todas las casillas de la lista se encuentren marcadas (puede usar el botón **Seleccionar todo** para tal fin). Deje sin marcar la casilla **Mostrar como vía de acceso de archivos locales** y haga clic sobre el botón **Aceptar** para comenzar la publicación del sitio Web en el disco duro.

Al intentar crear la estructura en nuestro sistema IBM WebSphere Studio mostrará una ventana donde se nos preguntará si deseamos crear las carpetas necesarias.

Marque la casilla **No volver a mostrar este diálogo** para que se creen las demás carpetas automáticamente y haga clic sobre **Sí** para continuar el proceso.

Tras unos instantes la publicación habrá finalizado. Abra entonces una ventana del **Explorador de Windows** y, dentro de la carpeta que contiene la publicación (la misma que en unidades anteriores), haga doble clic sobre el archivo **index.html** para abrirlo y compruebe que todo lo creado funciona sin problemas.

Es posible que el *applet* de reloj que se insertó no aparezca; esto puede ser debido a un problema a la hora de cambiar las rutas de los archivos. Para solucionar esto, abra el archivo **main.html** con cualquier editor y localice la línea que contiene la etiqueta *<APPLET>*.

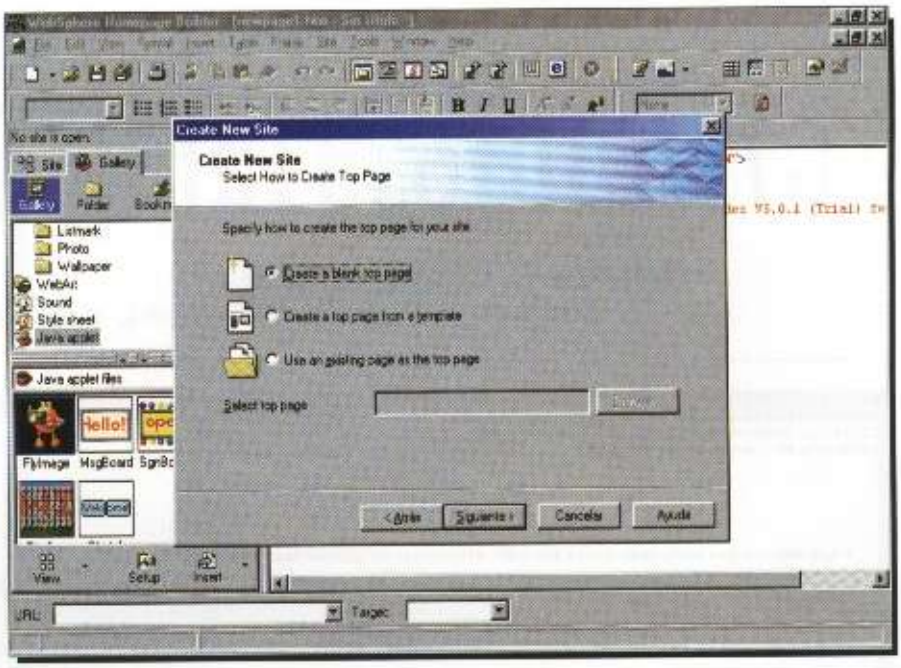


FIG. 21 La inclusión de asistentes hace mucho más asequible el trabajo con la aplicación.

Cambie el valor de **codebase** a *classes* en caso de que apareciera otro valor, guarde los cambios y compruebe de nuevo el funcionamiento del sitio. En este momento toda la nueva estructura y sus archivos estarían listos para ser ubicados en un servidor en Internet, por ejemplo, cargándolos mediante *ftp*. Para su instalación, introduzca el CD-ROM en la unidad y, en caso de que aparezca la pantalla Instalación de Web Services Toolkit, haga clic sobre el botón **Cancelar**. Abra el Explorador de Windows y vaya a la carpeta **Extras\Homepage \Install,** y una vez allí ejecute el archivo **hpb5tpgm.exe** haciendo doble clic sobre él.

FIG. 22 Las galerías en esta versión están llenas de objetos listos para ser utilizados en la creación de nuestras páginas.

El instalador se lanzará y le dará la bienvenida. Haga clic sobre el botón **Next >** para continuar, y tras leer el acuerdo de licencia de uso de la aplicación, haga clic sobre **Yes** para pasar a la ventana de elección de la carpeta de instalación. En esta ventana puede elegir una carpeta distinta de la señala-

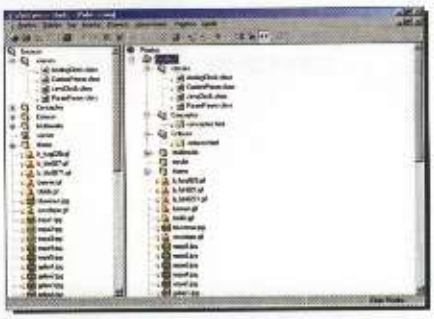


FIG. 23 Se publicará el servidor de prueba localhost directamente en el disco duro.

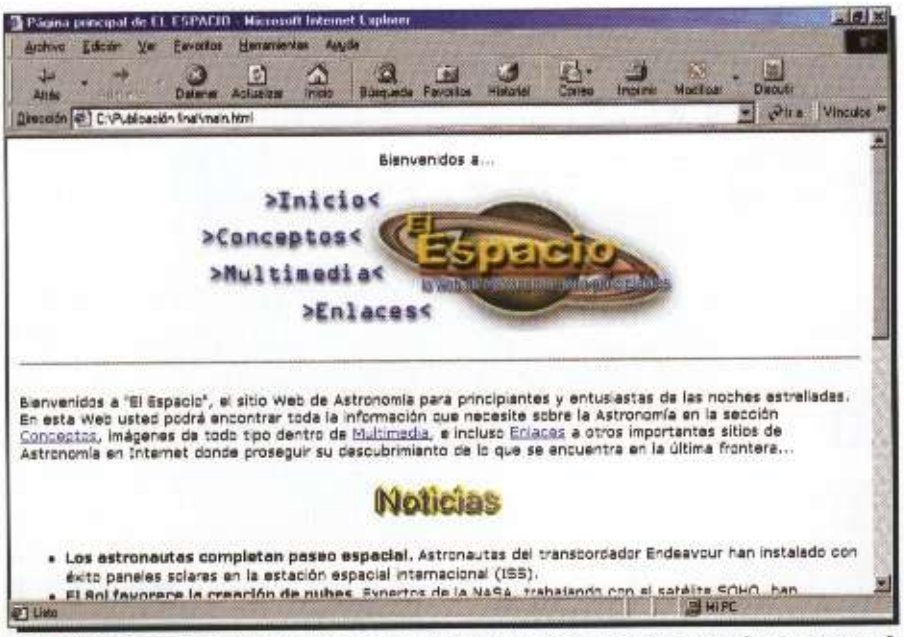


FIG. 24 Al hacer la publicación podremos comprobar el funcionamiento general del sitio Web.

da por omisión usando el botón **Browse...**, aunque le recomendamos que acepte la propuesta haciendo clic sobre **Next >**. Seguidamente, seleccione la opción **Typical**, vuelva a hacer clic sobre el botón **Next >** en las dos pantallas siguientes y, tras un breve proceso de copia de archivos, salga al sistema usando el botón **Finish**. Dentro del grupo de programas creado por el instalador, haga clic sobre el icono HomePage Builder para entrar en la aplicación.

Como observará, la interfaz de usuario es muy similar a la de IBM WebSphere Studio, por lo que no tendrá problemas para su uso.

Especialmente interesantes son las herramientas **WebArt Designer**, **Web Animator** y **File Transfer**, accesibles desde el menú **Tools** de HomePage Builder o directamente desde el grupo de programas creado en el menú **Inicio** de Windows.

Asegúrese de probar algunos applets únicos, como por ejemplo el fantástico **FlyImage**, que constituye una forma muy nueva para crear una galería de imágenes.

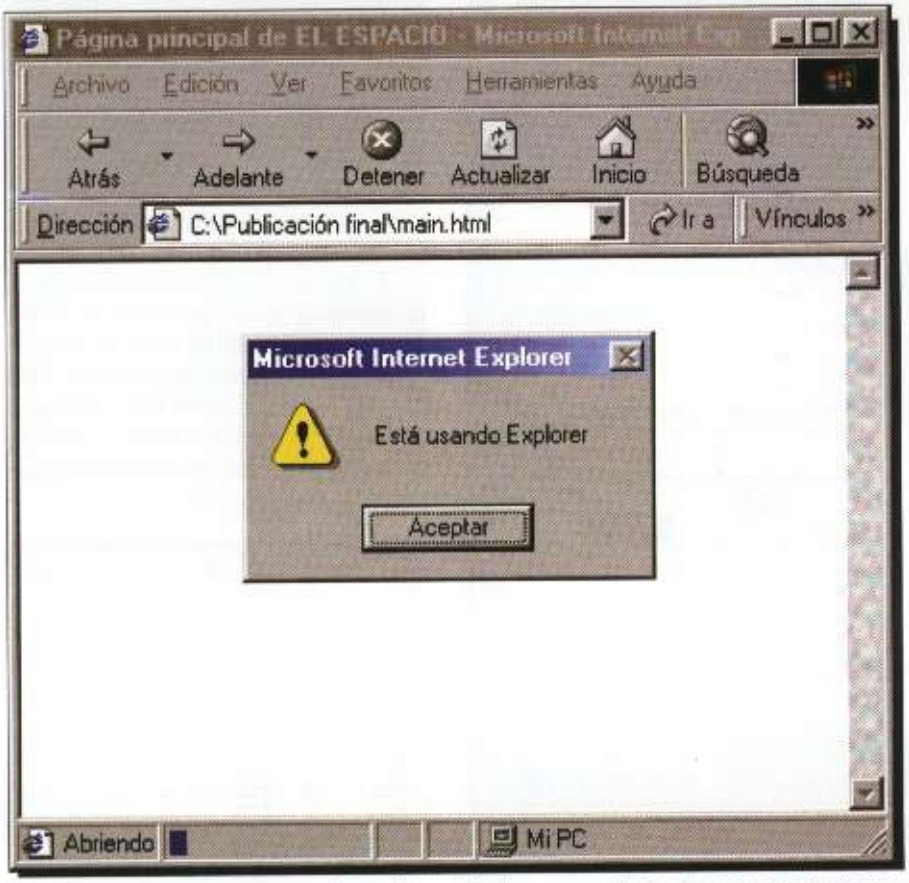


FIG. 25 Como puede comprobar, el script de reconocimiento de navegadores funciona correctamente.

RESUMEN

En esta unidad hemos visto como podemos añadir programas en los lenguajes Java y JavaScript a nuestras páginas Web, gracias a los cuales hemos podido hacer aparecer ventanas con cuadros de mensajes, animar un texto en la barra de estado del navegador o incluso conocer el tipo de navegador que está usando el visitante de nuestro sitio Web. Hemos visto además las notables diferencias existentes entre los applets y los scripts, para finalmente proceder a realizar una publicación de prueba del sitio y analizar la herramienta IBM WebSphere HomePage Builder V5.0, incluida en su versión de prueba en el CD-ROM adjunto. En la siguiente y última unidad dedicada a IBM WebSphere Studio veremos cómo finalizar el sitio Web "El Espacio", comprobando la sintaxis y ortografía en los documentos, los posibles vínculos rotos, o la falta de accesibilidad, así como la definitiva publicación del sitio en Internet.