TEMA

48

SISTEMAS Y APLICACIONES INFORMATICAS

Desarrollo de los temas

Lenguajes de alto nivel en entorno gráfico.

elaborado por
EL EQUIPO DE PROFESORES
DEL CENTRO DOCUMENTACIÓN

CEDE

1. INTRODUCCIÓN

En la actualidad todos los fabricantes de software de programación enfocan sus compiladores al desarrollo de programas en entornos gráficos. El motivo es la alta implantación de dichos entornos.

Diseñar y elaborar un programa para entornos gráficos puede ser una tarea imposible si se trabaja con herramientas convencionales, en las que habría que diseñar los elementos del entorno y la interacción con el sistema. Por ello han aparecido (y están apareciendo nuevos) compiladores de desarrollo de programas que incorporan librerías con las herramientas necesarias para utilizar los elementos de un entorno gráfico. Estas herramientas están diseñadas en base a alguno de los entornos gráficos más difundidos (X-Windows bajo Unix, Windows de Microsoft...).

2. CLASIFICACIÓN DE LOS LENGUAJES

Los lenguajes que incorporan funciones para desarrollo de programas en entornos gráficos se pueden clasificar en dos grandes grupos en función del modo de operar:

- Lenguajes visuales.
- Lenguajes no visuales.

Los lenguajes visuales funcionan solo en entornos gráficos y tienen herramientas gráficas para crear de forma automática todos los elementos básicos del entorno (ventanas, botones...) sin necesidad de escribir código para ello. En estos lenguajes, la escritura de código queda limitada a la codificación de las respuestas del sistema ante los diferentes eventos que provoque el usuario o el propio sistema. Estos lenguajes son muy válidos para el desarrollo de prototipos. Entre estos lenguajes se encuentran Visual Basic 4.0 y Delphi (pascal visual). Más orientados a la multimedia, pero dentro de este grupo, se encuentran ToolBook, Director...

Los lenguajes no visuales pueden funcionar en entornos de texto (aunque los programas generados por ello no). La inclusión en el programa de cualquier elemento visual implica la escritura de código más o menos complejo. No implica la creación de los elementos (objetos) sino la personalización de los mismos utilizando las librerías que incorpora. Como ejemplo de este tipo está Borland C++.

Además de esta clasificación, los lenguajes, en función de su organización interna y de las posibilidades que prestan al programador los lenguajes se pueden clasificar en:

- Orientados a objetos.
- Basados en objetos.

Los primeros, sin ser lenguajes puros de POO, permiten la creación de clases y objetos. Permiten además la creación de subclases de las clases propias del entorno gráfico (que vendrán suministradas en librerías específicas). En este grupo se encuentran Borland C++ y Delphi.

Los segundos basan la programación en la utilización de objetos predefinidos, permitiendo tan solo modificar valores de las propiedades y el código de algunos métodos (una serie de eventos permitidos). No permiten la creación de clases ni tan siquiera la creación de subclases de estas clases. Esta característica hace que sean lenguajes limitados en cuanto a prestaciones pero hacen mucho más sencilla la codificación de programas.

Veamos ahora dos de los lenguajes más difundidos para el desarrollo de programas en entornos gráficos.

3. VISUAL BASIC

Como se ha comentado, Visual Basic es uno de los lenguajes visuales. Funciona bajo Windows 3.1 o 3.11 (hasta la versión 3 inclusive) o bajo Windows-95 (a partir de la versión 4.0). El software que se puede desarrollar es de 16 bits con las versiones hasta la 4 inclusive, y software de 32 bits con las versiones 4 y posteriores; la versión 4 es la única que permite desarrollar software para ambas plataformas (16 y 32 bits). Se puede adquirir el software con diferentes prestaciones: Versión básica, versión profesional o versión empresarial (la versión empresarial solo apareció a partir de la 4), siendo la diferencia entre ellas, a parte del precio, las prestaciones que incluye (más limitadas en la versión básica).

En este capítulo se va a hacer un estudio de como se crea un programa sin entrar en detallar características del lenguaje (el lenguaje es el QBasic de Microsoft). Se pretende con ello dar al lector una visión de como se organiza y realiza un programa en un lenguajes visual, sin entrar en los detalles específicos de la codificación (tipos de instrucciones, gestión de ficheros...). Cualquier otro lenguaje visual funciona de una forma muy similar.

Empecemos viendo el entorno de trabajo.

EXPLORACIÓN DE LA PANTALLA

Después de que arranques Visual Basic aparecerán cinco ventanas en la pantalla, como se ilustra en la figura (En la pantalla inicial de Visual Basic las ventanas pueden solaparse; en la figura se ha modificado el tamaño y la posición de las ventanas en la pantalla, de manera que todas se puedan ver claramente.) En la parte superior de la pantalla está la ventana principal. Contiene los menús File (Archivo) y Edit (Edición) que son habituales y otros menús de Visual Basic, así como una barra de herramientas. En el centro de la pantalla está la ventana del formulario, una gran ventana en blanco con el título Form1. Directamente a la izquierda, puedes ver una ventana en forma de paleta de opciones denominada caja de herramientas. A la derecha de la ventana del formulario está la ventana Properties (propiedades) y debajo de ésta aparece la ventana Project (proyecto).

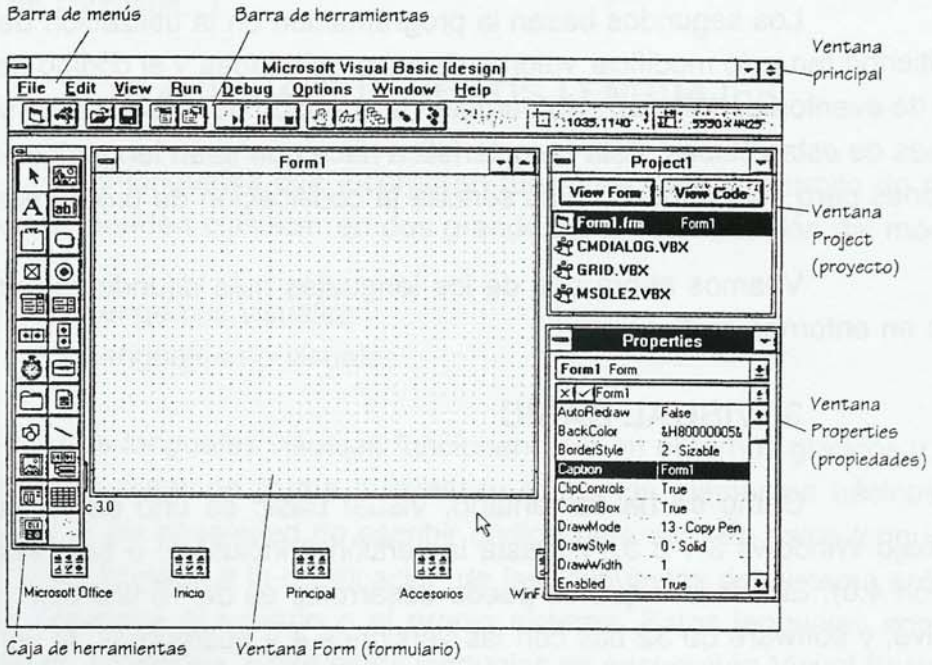

Figura 48.1. Pantalla de trabajo de Visual Basic

LA VENTANA PRINCIPAL

La ventana principal contiene la barra de menús con ocho menús desplegables. Uno de los más importantes, sobre todo mientras estés aprendiendo Visual Basic, es el menú Help (ayuda). Desde este menú puedes acceder a un tutorial que te presenta Visual Basic, puedes buscar la información necesaria para contactar con los servicios Microsoft de apoyo a sus productos y puedes explorar todo el sistema de Ayuda de Visual Basic.

Además, la Ayuda proporciona información sensible al contexto. Si se necesita saber más sobre un botón, un cuadro de diálogo, una ventana, un mensaje de error o sobre cualquier otro elemento con el que se esté trabajando en Visual Basic, simplemente hay que pulsar la tecla F1 cuando el elemento esté resaltado, y la Ayuda mostrará inmediatamente la información apropiada.

La ventana principal también contiene la barra de herramientas. Los botones de la barra de herramientas son atajos para los comandos que se usan con más frecuencia. Por ejemplo, en lugar de abrir el menú File (archivo) y seleccionar el comando Open Project (abrir proyecto) se puede, simplemente, hacer clic en el botón Open Project.

Finalmente, a la derecha de la barra de herramientas en la ventana principal se ven dos campos que sirven para indicar la posición y el tamaño del objeto que está seleccionado actualmente en la ventana del formulario.

LA VENTANA DEL PROYECTO

La ventana del proyecto (Project) contiene una lista de todos los archivos necesarios para ejecutar el programa de Visual Basic que estés creando. Aunque no se haya empezado todavía aparecen cuatro entradas en la ventana del proyecto. La primera es el nombre de archivo Form1.frm; su etiqueta (Form1) indica que el archivo está asociado con la ventana de formulario denominada Form1. Si se guarda el formulario en un disco sin cambiarle el nombre, Visual Basic usa el nombre de archivo por omisión Form1.frm. Una aplicación puede estar compuesta de muchos formularios, cada uno de los cuales se almacena en un archivo separado. (Como los archivos están separados, un formulario también puede ser compartido por otras aplicaciones.)

Las tres entradas adicionales que aparecen en la ventana del proyecto son los nombres de los archivos CMDIALOG.VBX, GRID.VBX y MSOLE2.VBX. La extensión VBX indica que esos archivos son archivos de extensión de Visual Basic. Cuando se carga un archivo de extensión se añaden herramientas suplementarias a la paleta de la caja de herramientas (que se trata en el siguiente apartado).

La ventana del proyecto también contiene dos botones, etiquetados View Form (ver formulario) y View Code (ver código). Por omisión, Visual Basic muestra el formulario correspondiente cuando se selecciona un archivo en la ventana del proyecto. Esta vista permite diseñar el interfaz de usuario para la aplicación: la parte de la aplicación que el usuario ve y con la que interactúa. Si se hace clic en el botón View Code de la ventana del proyecto, el código del archivo seleccionado aparecerá en una ventana diferente. El código se refiere

a las instrucciones en el lenguaje de programación. Cuando se crea un programa con Visual Basic, el trabajo se divide entre el diseño del formulario, el cual ve el usuario, y la creación del código, que controla el funcionamiento del programa. Para volver desde la ventana del código a la del formulario, simplemente hay que hacer clic sobre el formulario para activarlo o en el botón View Form en la ventana del proyecto.

LA VENTANA DEL FORMULARIO Y LAS HERRAMIENTAS

Un formulario es una zona de visualización que corresponde a una ventana que se verá cuando la aplicación esté funcionando. Cuando se empieza un proyecto nuevo, Visual Basic crea un formulario vacío y le da el título Form1. A medida que se diseña la aplicación, el formulario sirve como un lienzo en el que se puede dibujar diversas partes de la aplicación. Los componentes de la aplicación que se colocan en el formulario se denominan objetos o controles -cuadros de dibujo, botones de opciones y barras de desplazamiento, por ejemplo (de hecho, Visual Basic considera al propio formulario como un objeto)-.

Los controles se crean mediante la paleta de la caja de herramientas. Cada control está representado por un icono o herramienta de la caja de herramientas. La mayoría de las herramientas están construidas internamente en Visual Basic. Sin embargo, como ya se ha mencionado, la caja de herramientas puede ampliarse para incluir nuevas herramientas adicionales. Cada archivo con la extensión VBX de la ventana del proyecto proporciona una o más herramientas nuevas para la caja de herramientas. La herramienta Grid (cuadrícula), la herramienta OLE y la herramienta Common Dialog, por ejemplo, provienen de los archivos GRID.VBX, MSOLE2.VBX y CMDLALOG.VBX que se incluyen por omisión en tu archivo del proyecto. La Edición Profesional de Microsoft Visual Basic ofrece archivos VBX adicionales y proporciona la posibilidad de crear nuevas extensiones (si se es programador en C o en Pascal). También hay disponibles varios paquetes de software de otros fabricantes.

La apariencia del programa se diseña eligiendo controles de la caja de herramientas y poniéndolos en el formulario.

LA VENTANA DE PROPIEDADES

Las propiedades de Visual Basic son mecanismos formales que sirven para describir los atributos de un objeto.

Todo objeto de Visual Basic tiene propiedades específicas cuyos ajustes controlan la apariencia y comportamiento del objeto en una aplicación. Algunas propiedades están restringidas a ciertos valores. Por ejemplo, la propiedad Visible de un objeto puede ser

ajustada a True o False (verdadero o falso); es decir, el objeto es visible o invisible. Otras propiedades, como el título de una ventana de formulario puede ajustarse prácticamente a cualquier cosa. Los valores por omisión de muchas propiedades son, a menudo, perfectamente aceptables.

Aunque muchas propiedades pueden alterarse tanto en la fase de diseño como cuando la aplicación está funcionando, la ventana Properties, sólo está activa durante la fase de diseño.

Veamos ahora como se crean programas con Visual Basic. Todo programa constará de al menos un ventana de aplicación (formulario). Todos los elementos de un programa, salvo algunas rutinas de código, estarán incluidos en los formularios.

La creación de un programa en Visual Basic (y en cualquier lenguaje visual) tiene dos fases claras:

- Creación de la interfaz de usuario.
- Escritura del código.

3.1. CREAR UNA INTERFAZ DE USUARIO

La creación de la interfaz de usuario es la parte más rápida y fácil del programa. En esta fase se va a definir como se presentará nuestro programa al usuario:

- Se define la ventana de aplicación (formulario) y su aspecto.
- Se definen los objetos (controles) que van a incluirse en el formulario.

Como se ha comentado, Visual Basic presenta por defecto un formulario ya creado. Este nos servirá para empezar nuestra aplicación. Para definir su tamaño y ubicación bastará con realizar las operaciones típicas de Windows cuando queremos modificar una ventana (arrastrar la barra de título, arrastrar los bordes...). Para añadir otro formulario se utiliza la opción del menú 'Añadir nuevo formulario'. Un nuevo formulario puede definirse independiente del primero o integrado en el mismo. Si se define integrado, aparecerá incluido dentro del área de trabajo del primero.

Establecer las propiedades del formulario

Cuando se ejecute la aplicación, el formulario será presentado en una ventana de aplicación estándar. Si se quiere que esta ventana se parezca a las que muestran otras aplica-

ciones escritas para Windows, debe tener ciertos atributos. Por ejemplo, uno de los atributos compartidos con las aplicaciones escritas para Windows consistirá en que el nombre de la aplicación aparezca en la barra del título. En Visual Basic muchos atributos de ese tipo se controlan mediante las propiedades de los objetos.

A través de la ventana de propiedades ('Properties') se pueden definir los valores de puesta en marcha del formulario. Durante la ejecución del formulario el valor de la mayoría de estas propiedades podrá ser modificado. Entre las propiedades del formulario más utilizadas están:

- Caption. Define el texto que aparecerá en la barra de título de la ventana.

- Name. Todos los objetos en Visual Basic tienen una propiedad llamada Name (nombre). Cuando se establece la propiedad Name se le está dando al objeto una identidad que se puede usar dentro del programa para referirse a él. En una ventana del formulario se puede acceder a un objeto haciendo clic sobre él. En la parte del código del programa debemos referirnos al objeto por el nombre que se le hayas asignado con la propiedad Name.

Nota: Cuando se cambia el nombre de un formulario, Visual Basic refleja ese cambio en la lista de nombres de archivos que se muestran en la ventana del proyecto.

- BorderStyle (estilo del borde), la cual controla si el usuario podrá o no cambiar de tamaño la ventana cuando la aplicación esté funcionando. Por ejemplo el valor 3 - Fixed Double (fijo y doble) determina que la ventana tendrá un borde mediante el que no se podrá cambiar el tamaño de dicha ventana, y que no tendrá botones Minimizar ni Maximizar.

- Icon. Podemos definir que icono aparecerá si la aplicación es minimizada.

Además se dispone de propiedades para indicar si aparecerán los botones de control, definir los colores...

Añadir objetos

Una vez creado el formulario hay que incluir en él los objetos que queremos que contenga la aplicación. Para incluir un objeto hay que seleccionarlo de la ventana de herramientas (Tool Box). Los pasos a seguir para incluir un nuevo objeto son:

1. Hacer clic en el icono deseado de la caja de herramientas, y luego mover el puntero del ratón a la zona blanca de la ventana del formulario. El cursor se transforma en una cruz, lo que indica que está en el modo de dibujo.

2. Situar el cursor en la posición deseada del formulario. Mediante el procedimiento de "arrastrar" se definirá el tamaño del objeto. Al soltar el botón del ratón aparecerá la representación gráfica del objeto (un botón, una caja de texto, una barra deslizante...). Cuando se crea un nuevo objeto las propiedades (incluido el nombre -Name-) asumen una serie de valores por defecto, los cuales son los más idóneos para que mantenga una apariencia consistente con Windows. El nombre, aunque no sea necesario cambiarlo, es conveniente que sea sustituido por otro más significativo para nuestro programa.

Las propiedades de los controles varían según el tipo de control, aunque la mayoría son comunes a todos (Name, Captan, BackColor -color de fondo-, ForeColor -color de texto-, FontName -nombre de la fuente de texto-...).

El número de controles de los que dispone Visual Basic es bastante amplio, pudiéndose además incorporar nuevos controles creados por el programador. Incluir la lista de controles y sus características seria demasiado extenso, enumeraremos solo algunos de los más significativos:

- Check box (cajas de chequeo). Para crear cuadros tipos de Windows para activar o desactivar opciones.
- Combo box (listas desplegables).
- Command button (botones). Se puede crear botones simples o que contengan imágenes y texto en 3D.
- Common dialog. Para crea cajas de dialogo típicas de Windows: Abrir archivo, Guardar como, Seleccionar fuente...
- Data control. Para manejar ficheros de base de datos.
- Directory, Drive y File list box. Para incluir listas de directorios, unidades y ficheros.
- Image. Para incluir imágenes almacenadas en ficheros.
- Label. Para incluir texto.
- OLE control. Permite realizar operaciones OLE con otras aplicaciones.
- Text Box. Permite la visualización de un texto que el usuario puede editar.
- Timer. Para ejecutar un código asociado cada cierto tiempo.

Una vez creada la interfaz se puede ejecutar para ver su apariencia real.

3.2. ESCRIBIR EL CÓDIGO

Crear el código escribir las instrucciones que controlan el funcionamiento del programa. La codificación no se realiza de la forma convencional de cualquier otro tipo de lenguaje, en el que existe un programa principal y los procedimientos y funciones internas contenidos en un mismo bloque de texto. La programación en Visual Basic es "por eventos", lo que significa que el código se ha de escribir de forma independiente para cada uno de los eventos a los que se quiera que responda nuestro programa (p.e.: hacer Click en uno de los botones creados). Esto supone que el código de nuestro programa se encontrará "disperso" en bloques separados. Además de estos bloques se puede crea código en:

- Declaración de variables y procedimientos globales a un formulario. En el formulario se incluye una sección ("General") que permite definir variables globales ("Declarations") y procedimientos o funciones comunes a todos los controles del formulario.

- Módulos Basic externos para declaración de variables, procedimientos y/o funciones globales al proyecto.

Por tanto, la tarea de codificación se resume en escribir el código (acciones) asociado a cada uno de los eventos a los que queremos responder. Cada uno de los controles creados en el formulario tiene predefinidos una serie de eventos a los que se puede asociar código. Los eventos predefinidos pueden variar de unos controles a otros pero por lo general corresponden a las acciones típicas que el usuario (o el sistema) puede realizar: Click, Double Click, Mouse Move...

Por ejemplo, si queremos que al pulsar un botón creado en el formulario (hacer Click sobre él) nuestro programa visualice una Caja de Mensajes, los pasos a seguir son:

- Hacer doble Click sobre el botón. Aparecerá una ventana para escribir código en la que podemos seleccionar la lista de eventos programables para este control.
- Seleccionar el evento a controlar (en este ejemplo "Click").
- Escribir el código correspondiente. En nuestro ejemplo:

```
MsgBox "Ha pulsado el botón".
```

Desde el punto de vista del lenguaje, el código escrito para cada evento es como si hubiésemos creado un procedimiento (subrutina) en Qbasic. De hecho, la ventana de código de un evento presenta la estructura:

```
Sub CommandButton_Click()
EndSub
```

lo que significa que podría ser invocado por su nombre desde cualquier punto del programa. En ese sentido se mantienen las normas básicas de programación en Qbasic respecto al ámbito de variables, paso de parámetros...

Ejecución del programa

No es necesario hacer más para ejecutar el programa. Eligiendo la Opción "Run" o pulsando <F5> el programa se pondrá en marcha y podremos comprobar el funcionamiento del mismo. Visual Basic no necesita realizar un proceso previo de compilación ya que realiza automáticamente la traducción del código y comprobación de la sintaxis de todos los módulos al ser solicitada la ejecución del programa.

Visual Basic permite la creación de un programa ejecutable, el cual se podrá incorporar a un grupo de programas mediante las herramientas que incorpora Windows para ello. El icono que representará al programa, si no se modifica, será el establecido en la propiedad Icon del formulario.

Almacenar un programa

En Visual Basic un programa es considerado como un proyecto en el que se incluyen formularios, controles y módulos BASIC. Cuando se almacena un proyecto se generan dos tipos de fichero:

- Ficheros de proyecto (.MAK) que almacena información de los formularios, tipos de controles (ficheros .VBX) y módulos BASIC que intervienen en el programa. Por cada programa se almacena un único fichero .MAK.

- Ficheros de formularios (.FRM). Se almacena uno por cada formulario creado para el programa. En cada uno de ellos se almacena:
  - Objetos creados.
  - Código asociado a eventos y global para el formulario.

3.3. ÚLTIMAS NOVEDADES

Interesante

Con las últimas versiones de Visual Basic han aparecido, además de mejoras en los controles, interfaz y rendimiento, una serie de controles nuevos denominados genéricamente

ActiveX. A este grupo pertenecen tipos de controles que permiten desarrollar aplicaciones totalmente compatibles con Windows 95. La tecnología ActiveX es utilizada también por otros lenguajes de programación para desarrollo de aplicaciones Windows: Visual C++, Delphi...

La diferencia clave entre las diversas familias de controles está en la interacción entre el control y el resto de la aplicación, la interfaz del control. Los controles típicos de Windows usan una interfaz basada en mensajes, los controles de VB usan propiedades y eventos, los objetos OLE automatizados usan propiedades y métodos y los controles ActiveX usan propiedades, métodos y eventos. Un control ActiveX usa el mismo mecanismo que los objetos servidores OLE, que son los objetos que podemos insertar en un documento OLE. La diferencia entre un servidor genérico OLE y un control ActiveX es que los servidores OLE se pueden implementar de tres maneras distintas:

- Como aplicaciones independientes (por ejemplo Excel).
- Como servidores fuera de proceso, es decir, archivos ejecutables que no se pueden ejecutar por si mismos, sino que sólo se pueden invocar por un servidor (por ejemplo MS Graph y similares).
- Como servidores en proceso, como las DLL abiertas en el mismo espacio de memoria que el programa que las usa.

Los controles ActiveX se pueden implementar sólo usando la última técnica, que es la más rápida. Además, estos controles son servidores OLE automatizados. Esto significa que podemos acceder a las propiedades de estos objetos y llamar a sus métodos.

En un control ActiveX, las propiedades se dividen en grupos diferentes: propiedades de almacén que la mayoría de los controles necesitan implementar; propiedades de entorno que ofrecen información sobre el contenedor; propiedades extendidas gestionadas por el contenedor, como la posición del objeto; y propiedades personalizadas, que pueden ser cualquier cosa.

La ventaja más interesante de estos es que permiten crear aplicaciones que aprovechan al máximo Internet.

Se pasa ahora a comentar algunos aspectos de Internet respecto a los controles ActiveX que facilitarán al programador el desarrollo de este tipo de aplicaciones, aunque superen las pretensiones de este temario.

Desde una perspectiva, la tecnología Internet simplemente proporciona otra zona para sus tareas de desarrollo. Por ejemplo, al desplegar componentes ActiveX junto con el Web,

puede hacerlo de forma distinta: incorporando código de VBScript y HTML, proporcionando características de seguridad, etc., pero todavía continúa llamando a métodos, estableciendo propiedades y tratando eventos. De esta manera, puede aplicar todo su conocimiento como programador en Visual Basic al contexto Internet.

Desde otra perspectiva, la aplicación de la tecnología Internet le permite ampliar sus tareas de desarrollo de maneras nuevas y excitantes. Por ejemplo, entre las ventajas de combinar componentes ActiveX con la tecnología Internet cabe citar las siguientes:

- Mantenimiento más sencillo del cliente: los componentes desplegados como parte de un Web se pueden transferir según sea necesario, sin tener que solicitar a los usuarios que lo actualicen ejecutando un programa de instalación.

- Capacidad de ampliar la funcionalidad hasta Internet: mediante protocolos comunes, el desplazamiento entre una Intranet e Internet puede ser prácticamente invisible para el usuario.

Servidores y clientes Internet

Una manera común de considerar el desarrollo en Internet es en términos de relaciones cliente/servidor en las que cualquier extremo del intercambio puede ser "estático" o "activo". Un cliente o un servidor activo es dinámico en lugar de estático y ejecuta una secuencia de comandos o cambia su comportamiento conforme varía su estado interno. Un cliente o un servidor estático simplemente reacciona a un estímulo.

Considerar de este modo el desarrollo en Internet produce cuatro situaciones, que se ilustran en la siguiente figura:

1. De un cliente estático a un servidor estático.
2. De un cliente estático a un servidor activo.
3. De un cliente activo a un servidor estático.
4. De un cliente activo a un servidor activo.

De un cliente estático a un servidor estático

Internet empezó con clientes estáticos que enviaban solicitudes a servidores estáticos, los cuales realizaban cada solicitud devolviendo un archivo. Una vez transferida de esta manera una página HTML al cliente, la conexión finaliza y el servidor no realiza ninguna otra acción

hasta que reciba otra solicitud. El mismo tipo de relación existe al transferir archivos mediante el Protocolo de transferencia de archivos (FTP) o un lector de noticias Usenet.

Incorporar la funcionalidad de clientes estáticos puede implicar el uso de controles ActiveX relacionados con la tecnología Internet incluidos con Visual Basic para crear una herramienta FTP, un explorador de Web, un lector de noticias, etc. Estas características también se pueden incorporar a una aplicación mayor, incluyendo así esta aplicación en el contexto Internet.

De un cliente estático a un servidor activo

Suponga que utiliza un cliente estático (por ejemplo, un formulario del explorador de Web adjunta a una aplicación existente) para transferir una página HTML con cuadros de texto para especificar criterios de solicitud y botones para hacer clic y enviar las entradas como una cadena al servidor. Al enviar el texto, el servidor devuelve una página HTML personalizada, quizás mediante llamadas a funciones de una DLL, con los resultados de la solicitud. Se trata de un ejemplo de un servidor activo en acción, procesando datos para devolver resultados personalizados según condiciones variables.

Una de las maneras en que puede hacer que un servidor sea más activo es mediante bibliotecas DLL que desarrolle con Visual Basic y el acceso a través de Oleisapi2.dll. Oleisapi2.dll ajusta la funcionalidad de la Interfaz de programación de aplicaciones para servidores de Internet (ISAPI), una característica Internet Information Server (IIS), incluido con Windows NT Server 4.0. Con IIS y Oleisapi2.dll, puede desplegar la DLL en el servidor Web y llamar a sus métodos correspondientes desde el equipo cliente utilizando información de una página HTML.

De un cliente activo a un servidor estático

Puede convertir al cliente en activo al proporcionarle capacidad de procesamiento. Los clientes activos pueden ofrecer ventajas de rendimiento realizando tareas de forma local, en lugar de enviar una solicitud al servidor para su procesamiento. Como en otras aplicaciones cliente-servidor, el trabajo realizado localmente no sólo agiliza muchas tareas, sino que ayuda a liberar el servidor para otras solicitudes.

Las situaciones que representan clientes activos son tan variadas como las aplicaciones creadas con Visual Basic. Por ejemplo, los documentos ActiveX que cree con Visual Basic pueden contener casi toda la funcionalidad que las aplicaciones tradicionales pueden incluir. Al

cargar éstas en un contenedor como Internet Explorer, el cliente se convierte en activo con la funcionalidad propia de las aplicaciones tradicionales en Visual Basic.

Un cliente también puede ser activo mediante una página HTML que ofrezca controles ActiveX o bibliotecas DLL cuyos métodos se llaman con VBScript.

De un cliente activo a un servidor activo

El despliegue de aplicaciones que se benefician de clientes y servidores activos puede proporcionar una funcionalidad mucho mayor que la suma de las partes descritas en los ejemplos anteriores. Si diseña una aplicación para que tanto el equipo cliente como el servidor puedan confiar en la capacidad del otro para participar en intercambios de proceso, pueda distribuir la funcionalidad para obtener soluciones flexibles y escalables que se beneficien al máximo de los recursos del sistema y de la red.

Estas soluciones pueden incluir, por ejemplo, aplicaciones desplegadas en entornos cliente-servidor de tres niveles. En esta opción, en la que componentes distribuidos por la red encapsulan servicios para proporcionar una interfaz de usuario, implementando reglas comerciales y teniendo acceso a datos, el acceso a los datos puede ser más sofisticado, ya que está disponible el procesamiento por parte del servidor y del cliente.

Un servidor activo puede hacer un seguimiento de los equipos cliente de los que recibe consultas, enviando las consultas y devolviendo los resultados. Un cliente activo puede desempaquetar y presentar los resultados devueltos al usuario y luego enviar los cambios al servidor. Además, mediante la transferencia de componentes en Internet, las reglas del equipo cliente para enviar consultas y procesar los resultados se pueden actualizar automáticamente siempre que el usuario visite el sitio Web del que se transfieren los componentes del cliente.

Todo esto contrasta con las aplicaciones que emplean las capacidades tradicionales (estáticas) y protocolos de Internet, en los que se pueden limitar las consultas enviadas y los resultados devueltos, por ejemplo, al alcance del envío de parámetros de instrucciones.

Actualizaciones por lotes mediante un documento ActiveX

Suponga una situación típica en la que un usuario envía una consulta, modifica las filas devueltas y vuelve a enviar los datos para actualizar la base de datos.

El usuario se desplaza a un documento ActiveX mediante un hipervínculo de una página HTML de la Intranet. Si el componente que proporciona el documento ActiveX no existe

en el equipo del usuario o no está actualizado, se transferirá automáticamente un nuevo componente. El documento ActiveX, contenido en Internet Explorer se asemeja a una aplicación estándar de Visual Basic.

Al hacer clic en el botón **Buscar** para abrir un cuadro de diálogo, el usuario especifica unos criterios de búsqueda, hace clic en **Aceptar** y luego en **Enviar** para transmitir la consulta. Una función del documento ActiveX empaqueta la consulta y la envía al servidor HTTP.

En el servidor, se desempaqueta el archivo y se envía a la base de datos de SQL Server. Un componente ActiveX del servidor hace un seguimiento del equipo cliente que envía la consulta y de las filas afectadas de la base de datos.

Las filas resultantes se empaquetan y se devuelven al equipo cliente, donde se desempaquetan y leen en un control de cuadrícula del documento ActiveX. El usuario actualiza las filas y luego hace clic en el botón **Actualizar** para enviar los datos modificados al servidor.

4. BORLAND C++. OBJECTWINDOWS

En este capítulo se va ha estudiar la librería que incorpora Borland C++ para la programación en Windows. No se van a describir ni instrucciones del lenguajes ni métodos de programación ya que han sido estudiados en temas anteriores.

UTILIZACIÓN DE OBJECTWINDOWS

¿Qué es ObjectWindows para C++? Básicamente, ObjectWindows es una interfaz C++ cuyo propósito es ocultar al programador el API Windows (Interfaz de programación Windows). Por supuesto, el motivo oculto es simplificar la programación Windows. ObjectWindows hace esto ofreciendo un nuevo interfaz orientado a objeto a través del cual los programadores pueden tratar fácilmente con el API Windows. Que ObjectWindows logre totalmente su propósito depende de los programadores que los usen, para los que hay tanto desventajas como ventajas en encapsular y ocultar todo el entorno que se quiere controlar. Las desventajas del uso de ObjectWindows parten del hecho de que se debe entender completamente los procesos ocultos (tanto en el API Windows como en el interfaz ObjectWindows) para controlar el comportamiento de los programas.

PRESENTACIÓN DE OBJECTWINDOWS PARA C++

La simple descripción de ObjectWindows como una interfaz de los programadores con el API Windows no explica completamente como trabaja ObjectWindows. Más

específicamente, ObjectWindows es una librería de clases C++ y de funciones miembros que se pueden llamar, que, nuevamente, hacen llamadas a las funciones del API Windows. Puede diseñar su programa Windows para incluir las funciones ObjectWindows, tanto mediante el linkado del fichero OWL.LIB como mediante la realización de llamadas dinámicas a la librería de enlazado dinámico OWL.DLL. La primera alternativa ofrece un poco de velocidad de ejecución extra, pero incrementa el tamaño final del fichero EXE. La segunda alternativa no incrementa el tamaño del EXE, pero reduce considerablemente la velocidad de ejecución debido a la sobrecarga del enlazado dinámico de las rutinas de librería. Ambas alternativas son normalmente aceptables. Ambas librerías contiene las funciones miembro de las clases ObjectWindows (bien en su formato enlazable bien en su formato llamable, respectivamente).

ENTENDIENDO LA LIBRERÍA OBJECTWINDOWS

El primer paso en el control de ObjectWindows es adquirir un imagen mental clara de las clases C++ en que esta estructurado ObjectWindows. La figura hace un diagrama de la estructura de clases ObjectWindows.

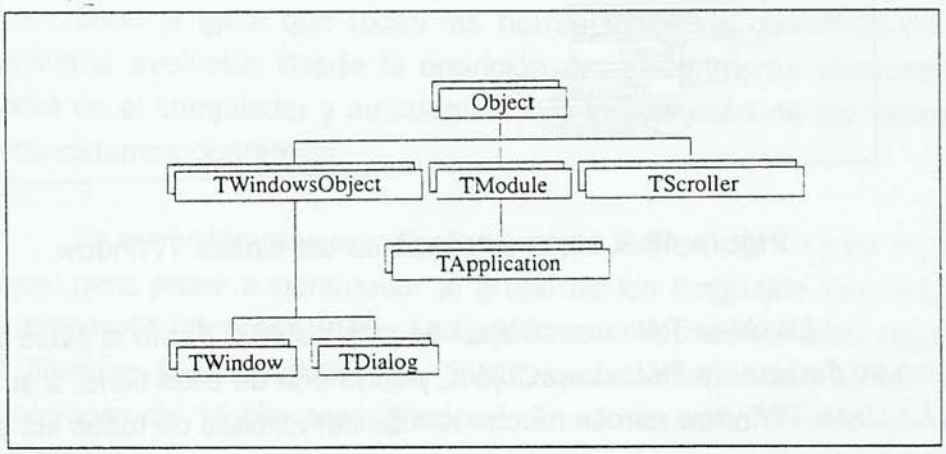

Figura 48.2. Vista resumen de la estructura de clases de ObjectWindows

Como muestra la figura, las clases de ObjectWindows se derivan de la clase base Object. La clase Object no es una clase ObjectWindows, sin embargo, Object es la clase base abstracta de la librería de clases Borland C++. Es la clase de la se derivan no sólo las clases ObjectWindows sino también las clases de objetos más generales (listas, pilas, y demás). Ninguno de los tipos de objetos de la librería de clases está ausente de ObjectWindows. Las tablas virtuales de trato dinámico (DDVTs), que se usan para dirigir los mensajes Windows a los manejadores de mensajes, son objetos list de la librería de clases C++. Por

tanto, para obtener un dominio total en ObjectWindows, se deben entender otras tres disciplinas: programación Windows, la estructuración C++ de los programas en el entorno Windows, y la librería de clases C++.

ObjectWindows deriva tres clases bases de la librería de clases abstracta Object: TWindowsObject, TModulo, y TScroller. TScroller es la más simple de las tres porque no tiene más clases derivadas. TModule es un poco más complicada, porque únicamente se deriva una clase de ella: TApplication.

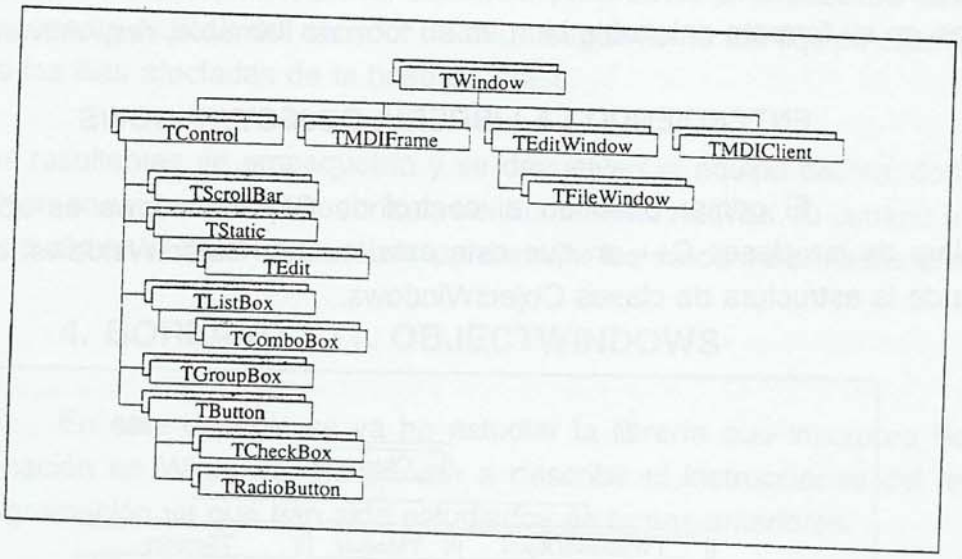

Figura 48.3. Vista detallada de las clases TWindow

La clase TWindowsObject es otra historia. Tanto la clase TWindow como la clase TDialog se derivan de TWindowsObject, y cada una de ellas tiene, a su vez, otras clases derivadas. La clase TWindow es con mucho la más complicada de todas las clases ObjectWindows. TWindow puede servir bien directamente como una clase con objetos propios, o bien como clase base para cuatro clases derivadas más. Las cuatro clases adicionales, que son TControl, TMDIFrame, TEditWindows, y TMDIClient, pueden tener sus propios objetos ventana o servir como clase base para objetos que cubren la mayor parte del completo rango de los tipos de ventana posibles. El desglose detallado de la jerarquía de clases se muestra en la figura anterior.

La clase base TDialog también puede ser desplegada en otras tres clases derivadas. Sin embargo, la jerarquía de la estructura de clase de diálogo no es tan complicada como la estructura de Twindow. La jerarquía de las clases de diálogo se muestran en la figura siguiente.

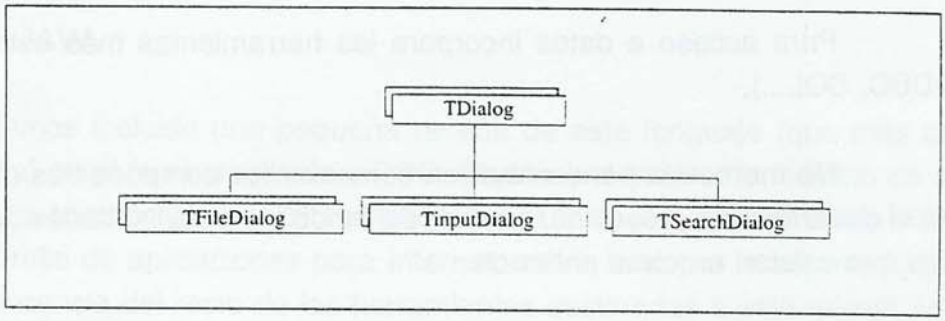

Figura 48.4. Vista detallada de las clases TDialog

El estudio de la organización de las clases ObjectWindows, como se muestra en las figuras, le pueden ayudar a formar una imagen mental general de la forma en que ObjectWindows distribuye la funcionalidad de sus clases. Sin embargo, para entrar en más detalle desde el principio probablemente no le ayudará mucho.

ACTUALIDAD DE C++

C++, al igual que todas las herramientas de desarrollo de software, ha sufrido una constante evolución desde la aparición de sus primeras versiones. A parte de sucesivas mejoras en el compilador y su adaptación a la evolución de los microprocesadores y entornos de los sistemas operativos.

La evolución más significativa es que hoy en día C++ ha dejado de ser un lenguaje no visual para pasar a pertenecer al grupo de los lenguajes visuales, presentando una interfaz de desarrollo integrado (IDE). Las primeras versiones visuales seguían utilizando el conjunto de librerías OBJETWINDOWS (OWL), que contienen las clases que permiten desarrollar aplicaciones de 16 bits para Windows. En el momento en que se escriben estas líneas acaba de aparecer una versión de este compilador que se pretende sea el unificador de todas las versiones que circulan por el mercado. Esta versión a aparecido con el nombre C++ Builder (va ya por la versión 3.0).

Además de las características típicas de un entorno de desarrollo visual, esta versión incorpora un nuevo conjunto de librerías para el desarrollo de software de 32 bits para Windows (MFC), aunque soporta todavía la "vieja" jerarquía de clase OWL.

Por supuesto C++ incorpora en la actualidad la tecnología ActiveX, para el desarrollo de aplicaciones orientadas a Internet.

Para acceso a datos incorpora las herramientas más estandarizadas del momento (ODBC, SQL...).

No merece la pena entrar en comentar los componentes del entorno y las técnicas para el desarrollo de proyectos, ya que, salvando las peculiaridades del lenguaje, son similares a los que existen en otros entornos.

5. OTROS LENGUAJES

5.1. DELPHI

Se puede definir Delphi como el lenguaje visual de Pascal, más exactamente la evolución a "lo visual" de Object Pascal (Pascal Orientado a Objetos). Está enfocado, sobre todo en las últimas versiones, para el desarrollo de aplicaciones para Windows.

Mantiene, por supuesto, las características de lenguaje Pascal, aunque evolucionado a POO y a componentes Windows. Tampoco en este caso pasamos a describir el entorno ni a explicar las técnicas de desarrollo ya que Delphi es muy similar a las demás herramientas en este sentido, aunque haya muchos autores que apuestan por esta herramienta como la mejor para desarrollo de aplicaciones en entorno Windows.

Pasamos a enumeras una pequeña lista de las principales prestaciones de Delphi:

- Dispone de los controles de Windows (tanto 16 como 32 bits) y algunos específicos.
- Permite creación de aplicaciones MDI (Interfaz de Documento Múltiple).
- Posibilidad de creación de aplicaciones con acceso a datos. Dispone de un gestor específico, aunque también es compatible con ODBC y SQL.
- Creación de componentes (a partir de nuevas clases).
- Acceso a API de Windows y creación de librerías DLL.
- Permite implantar OLE.
- Soporta tecnología ActiveX.
- Programación para Internet, herramientas para manejo de HTML.
- Incluye herramientas para desarrollo de aplicaciones multimedia.
- Soporte extendido para la red y aplicaciones distribuidas.

Como se puede observar, Delphi es una herramienta completa para el desarrollo en la actualidad de cualquier tipo de aplicación, con las ventajas que el lenguaje Pascal tiene, clara estructuración, fácil sintaxis...

5.2. JAVA

Ya hemos incluido una pequeña reseña de este lenguaje (que más que lenguaje es una tecnología) en el tema dedicado a POO. El principal motivo de incluirlo en este capítulo es el auge que ha tomado en los últimos años desde que se presentó como mejor herramienta para el desarrollo de aplicaciones para Internet, sobre todo por la característica multiplataforma que lo diferencia del resto de las herramientas destinadas a este mismo fin.

Se habla de multiplataforma cuando un único desarrollo de software permite que la aplicación resultante funcione bajo diferentes sistemas (UNIX, Windows...) sin necesidad de adaptaciones a las particularidades de cada sistema. Lo único que hará falta es la (o las) librería específica de cada sistema.

Al igual que C++, Java ha sufrido evoluciones hacia las herramientas visuales; pero a diferencia de este, siguen existiendo muchos programadores que siguen desarrollando aplicaciones utilizando las técnicas descritas en el tema de POO mencionado. El motivo no es que estos programadores no acepten la evolución a entornos visuales, sino que defienden la característica multiplataforma de Java.

Las diferentes herramientas visuales que han aparecido han tenido que romper con esta característica de multiplataforma (el entorno visual necesita adaptarse a la plataforma sobre la que funciona). No existen versiones multiplataforma de las herramientas visuales.

Las versiones visuales para PC (de las que destaca Visual J++ 6.0) incluyen las librerías de clases (WFC –Windows Fundation Classes–) necesarias para desarrollo de aplicaciones Win32 (aplicaciones de 32 bits para Windows), soportando las tecnologías para este entorno comentadas en las herramientas anteriores.

Desde el punto de vista de multiplataforma solamente decir que Java permite el desarrollo de un tipo especial de aplicaciones que funcionan a través de Navegadores (aplicaciones para Internet). Estas aplicaciones se denominan genéricamente Applets y funcionarán en cualquier Navegador que soporte las directivas Java (NetScape, Explorer...; aunque este último ha introducido modificaciones en estas directivas que a parte de romper la filosofía multiplataforma, le ha supuesto un contencioso con Sun MicroSystem –creadora de Java–).

RESUMEN

Los lenguajes que incorporan funciones para desarrollo de programas en entornos gráficos se pueden clasificar en dos grandes grupos en función del modo de operar:

- Lenguajes visuales.
- Lenguajes no visuales.

Además de esta clasificación, los lenguajes, en función de su organización interna y de las posibilidades que prestan al programador los lenguajes se pueden clasificar en:

- Orientados a objetos.
- Basados en objetos.

Visual Basic es un lenguajes visual basado en objetos que permite el diseño de programas para Windows de una forma fácil y eficaz.

Borlan C++ es un lenguaje orientado a objetos que trabaja en modo no visual. Contiene una librería (ObjetWindows) con las funciones necesarias para crear programas que funcionen bajo Windows. En la librería se incluyen las clases que utiliza Windows. Su manejo es mucho más complicado que el de Visual Basic ya que exige un conocimiento exhaustivo de la estructura interna de Windows y su funcionamiento.

EDITA Y DISTRIBUYE:

CENTRO DOCUMENTACIÓN DE ESTUDIOS Y OPOSICIONES
C/ CARTAGENA, 129 - TELS. 564 39 94 - 28002 MADRID