Programación con Delphi (IV)

© Francisco Charte Ojeda - http://www.fcharte.com

Sumario

La edición empresarial de las últimas versiones de Delphi, de la 5 a la 7, cuenta con más de doscientos componentes de todo tipo. Existe un grupo, no obstante, que son de uso habitual en la mayoría de las aplicaciones.

Al hablar de los componentes de Delphi, hay que tener en cuenta que no estamos hablando mas que de clases a partir de las cuales pueden crearse objetos, según se describió en la entrega previa. Esas clases, además, no son independientes, sino que se derivan unas de otras. Esto tiene para nosotros, que somos los usuarios de los componentes, una ventaja fundamental: la mayoría de los componentes cuentan con muchas características comunes.

La posición de un componente en el interior del contenedor donde está alojado, por ejemplo, siempre se almacena en las propiedades `Left` y `Top`. Éstas están definidas en la clase `TControl`, de la cual derivan otras como `TWinControl` que, a su vez, son la base de botones, cajas de texto, listas, etc. De forma análoga, existen métodos como `Show()` y `Hide()` o eventos como `OnClick` generales a la mayoría de los controles.

En esta cuarta entrega, va a tener oportunidad de conocer algunos de los controles usados con más asiduidad en el diseño de interfaces de usuario con Delphi, así como sus miembros más básicos: propiedades, métodos y eventos.

Contenedores de controles

Cuando uno se dispone a construir algo, pongamos por caso un automóvil, lo primero que hace es disponer los contenedores en los cuales se integrarán los demás elementos, como el chasis, el salpicadero, hueco-motor, etc. Para crear una interfaz de usuario ocurre más o menos lo mismo, aunque las posibilidades en cuanto a flexibilidad son mucho más amplias.

Nuestro chasis o contenedor principal será el formulario, como ya vimos en la segunda entrega. En el interior de éste es posible disponer otros contenedores, elementos que tienen por finalidad alojar a otros controles. Esto se hace, generalmente, para agrupar elementos funcionales o hacer más claro el diseño de la interfaz.

Los dos contenedores usados más habitualmente se encuentran en la página `Standard` de la Paleta de componentes, como puede verse en la Figura 1. En la Figura 2 se muestra el aspecto de tres contenedores: un `GroupBox`, un `Panel` y el propio formulario. Observe que el `GroupBox` muestra el título en su parte superior izquierda, el `Panel` en la parte central y el formulario en su barra de título. Independientemente de esto, el texto mostrado como título siempre se almacena en la misma propiedad: `Caption`. Puede comprobar, en el Inspector de objetos, que estos tres elementos cuentan con esa propiedad, que puede modificar simplemente introduciendo el texto que desee.

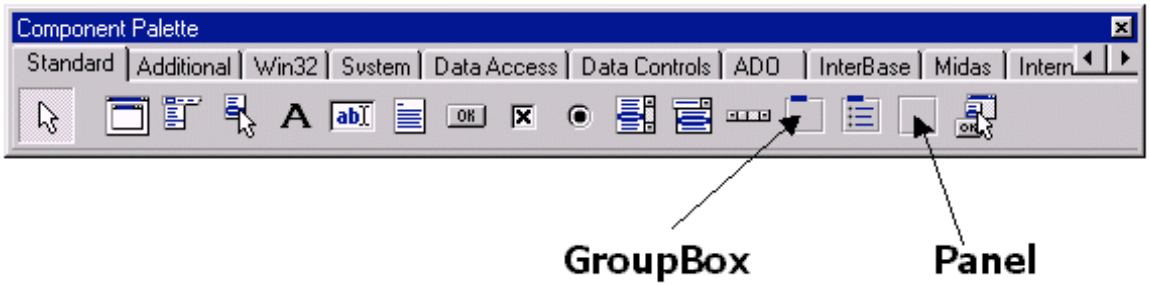

Figura 1. En la página Standard de la Paleta de componentes podemos encontrar dos de los contenedores con que cuenta Delphi.

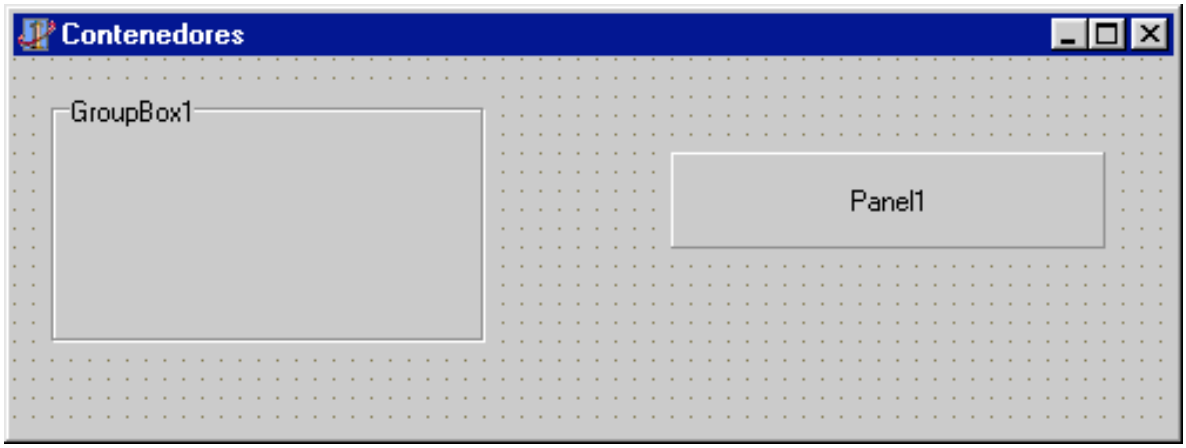

Figura 2. Aspecto que tienen los controles GroupBox y Panel al ser insertados en un formulario.

Exceptuando el caso de los formularios, que no pueden introducirse en otros contenedores, dentro de un contenedor puede alojar cualquier control, incluidos otros contenedores. Dentro un `Panel`, por ejemplo, puede disponer varios `Panel` más para crear varias divisiones.

Inicialmente, la posición de un contenedor dentro de otro vendrá determinada por sus propiedades `Left` y `Top`, mencionadas previamente. Asimismo, las dimensiones se indicarán con las propiedades `Width` y `Height`, ancho y alto respectivamente. Una alternativa a estas cuatro propiedades, en el caso de los contenedores, consiste en usar la propiedad `Align`, que facilita la alineación del control a uno de los márgenes del contenedor, o bien ocupando todo el espacio disponible. En la Figura 3 puede ver cinco controles `Panel`. El primero de ellos se ha insertado directamente sobre un formulario, dándose el valor `alClient` a su propiedad `Align` para que ocupe todo el espacio disponible. A continuación, dentro de éste se han insertado otros cuatro `Panel`, ajustando cada uno de ellos a un extremo.

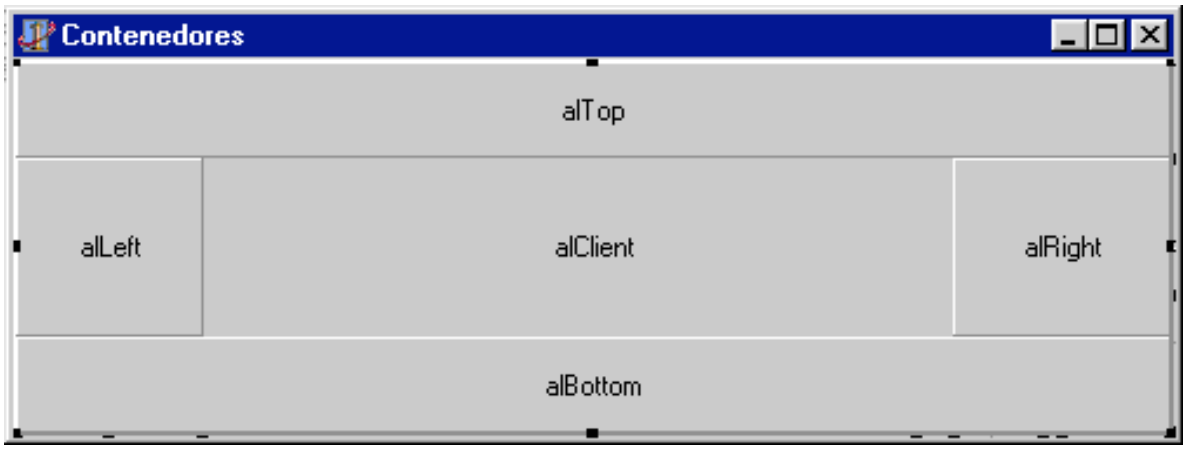

Figura 3. Mediante la propiedad Align es posible alinear unos contenedores dentro de otros, creando espacios independientes para agrupar los elementos de la interfaz.

Todos los contenedores cuentan con las propiedades `ComponentCount`, `ControlCount`, `Components` y `Controls`, que sirven para saber cuántos componentes o controles hay en el contenedor, facilitando el acceso individual a cada uno de ellos mediante un índice numérico. De esta forma es posible efectuar, durante la ejecución, operaciones que afectan a múltiples elementos de la interfaz sin necesidad de hacer referencia directa a cada uno de ellos.

Introducción de datos

Gran parte de la información que se manipula habitualmente en las aplicaciones, independientemente de su tipo, debe ser introducida o editada por el usuario mediante el teclado. Para facilitar esta tarea existen diversos componentes, de los cuales conocemos uno por haberlo usado en la segunda entrega de este curso. En la Figura 4 se indican dos de los componentes disponibles para este fin: `Edit` y `Memo`.

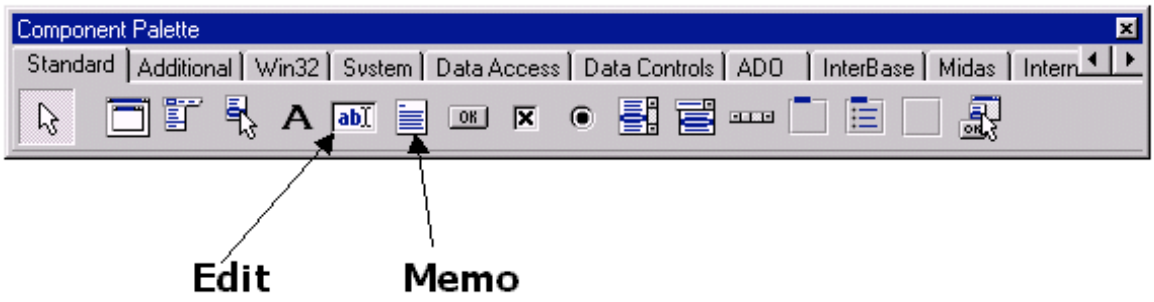

Figura 4. Los controles Edit y Memo, usados para facilitar la edición de texto, también se encuentran en la página Standard, como todos los controles más habituales.

El control `Edit`, usado en el citado ejemplo de la segunda entrega, se utiliza cuando el texto a introducir o editar está contenido en una sola línea. La propiedad `Text` es la que almacena el texto, pudiendo asignársele un valor, para facilitar un texto inicial, así como leerlo, para recuperar el dato introducido por el usuario.

A diferencia del anterior, el control `Memo` está preparado para contener múltiples líneas de texto, facilitando al usuario el desplazamiento y la edición. Las líneas de texto se almacenan en la propiedad `Lines`, que se caracteriza por ser una lista en la cual cada línea de texto está identificada por un índice numérico. No obstante, también existe la propiedad `Text`, como en el caso del control `Edit`, que puede utilizarse para asignar o recuperar todo el texto, en lugar de líneas concretas.

Aparte de la propiedad `Text`, estos dos controles comparten muchas otras características. La entrada de texto puede controlarse gestionando el evento `OnKeyPress`, generado con cada pulsación de tecla. Es posible usar las habituales combinaciones de tecla para copiar, cortar y pegar texto, existiendo además varios métodos para efectuar esas mismas operaciones desde programa.

Para mostrar un texto ya conocemos el control `Label`, que en lugar de la propiedad `Text` dispone de la propiedad `Caption`, como los contenedores que hemos conocido en el punto previo. `Edit` y `Memo`, no obstante, también pueden ser usados para mostrar un texto sin permitir su edición, simplemente dando el valor `True` a la propiedad `ReadOnly`. Esta propiedad, que inicialmente tiene el valor `False`, indica si el contenido del control puede o no ser modificado.

Selección de opciones

En ocasiones, los datos que precisa una aplicación pueden ser facilitados por el usuario mediante medios alternativos al uso del teclado. Suponga que dicho usuario debe rellenar un formulario con datos como el estado civil, sexo, si tiene automóvil, vivienda propia, etc. La mayoría de estas cuestiones tienen sólo unas pocas respuestas posibles, en ocasiones tan sólo dos: sí o no.

Los dos controles indicados en la Figura 5, conocidos como `CheckBox` y `RadioButton`, son los más usados cuando se quiere dar al usuario a elegir entre varias opciones. El primero se usa cuando la respuesta puede ser sólo sí o no, al tratarse de un control sobre el que el usuario puede simplemente pulsar para activar o desactivar. Un caso típico sería la respuesta a la pregunta "¿tiene hijos?", que sólo puede ser contestada con un sí o un no.

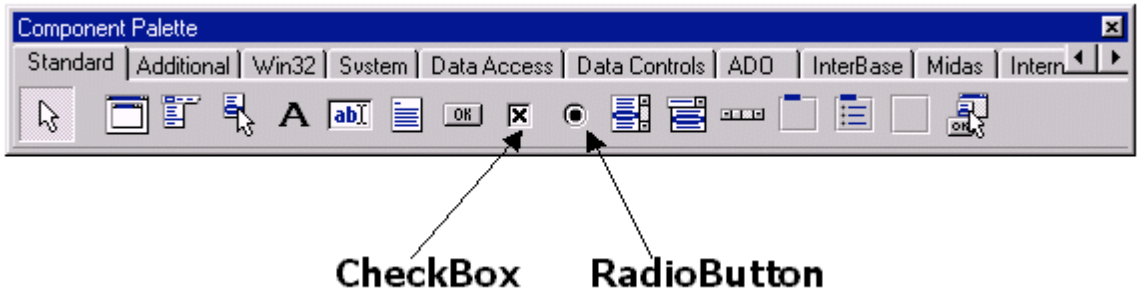

Figura 5. Existen controles para mostrar al usuario una serie de opciones entre las que elegir, en lugar de tener que introducir un texto.

Mediante el control `RadioButton`, que se utiliza habitualmente formando grupos, podríamos plantear cuestiones con más posibilidades. Planteemos la pregunta "¿Cuál es su estado civil?". En este caso las respuestas son varias, pudiendo elegirse sólo una de ellas puesto que no puede estarse casado y soltero al tiempo, por poner un ejemplo. Por cada respuesta posible se insertaría un `RadioButton`, que el usuario podrá seleccionar simplemente haciendo clic sobre él. Esta acción, además, desactivará la opción que existiese anteriormente elegida.

Ambos controles, al igual que otros muchos, disponen de la propiedad `Caption`, que nos servirá para establecer un título que, generalmente, será la pregunta u opción que el usuario puede elegir. Además, ambos disponen también de la propiedad `Checked`, que sólo puede ser `True` o `False`, según que la opción haya sido elegida o no.

Estado y visibilidad de los controles

Aparte de todas las propiedades mencionadas en los puntos anteriores, muchas de ellas comunes a un grupo de controles, existen otras genéricas, que afectan a todos ellos, y que rigen el estado en el que se encuentran y su visibilidad.

Un componente puede encontrarse activo o desactivo, según lo cual el usuario podrá o no interactuar con él y, además, aparecerá con una apariencia u otra en pantalla. Para conocer y modificar el estado se usa la propiedad `Enabled` que, al igual que `Checked` y muchas otras, tan sólo puede contener los valores `True` o `False`, al tratarse de una propiedad de tipo `Boolean`.

A diferencia de los componentes físicos, como las piezas de un motor o los ladrillos de una vivienda, los componentes software que usamos en Delphi pueden aparecer y desaparecer a voluntad, es decir, pueden hacerse invisibles. La visibilidad la controlaremos mediante la propiedad `Visible`, que es también de tipo `Boolean`. Inicialmente, dicha propiedad siempre tiene el valor `True`.

Que los componentes cambien de estado, pasando de estar activos a no estarlo, o de ser visibles a no serlo, es algo que dependerá de la propia funcionalidad de la aplicación que se desarrolle. Dicho de otra manera: no hay una norma que nos indique cuándo debemos dar el valor `True` o `False` a las citadas propiedades `Enabled` y `Visible`.

Eventos genéricos

Todos los controles que hemos conocido, por el hecho de estar directa o indirectamente derivados de la clase `TControl`, cuentan con una serie de eventos genéricos, como `OnClick`, `OnDblClick`, `OnMouseDown`, `OnMouseUp` u `OnMouseMove`. El primero de ellos, que es el evento más usado, se produce al pulsar sobre un control, generalmente usando el ratón. Los otros cuatro tienen lugar al efectuar un doble clic, pulsar un botón, liberarlo y mover el ratón, respectivamente.

Podemos saber, por tanto, cuándo el puntero del ratón está desplazándose sobre un `Panel` o si se ha pulsado en un `CheckBox` cambiando su estado. Son eventos que pueden desencadenar una cierta acción, por ejemplo cambiando el estado de un elemento según que el `CheckBox` esté seleccionado o no.

Además de los anteriores, los controles que trabajan con texto, como `Edit` y `Memo`, cuentan con otros eventos adicionales relacionados con el teclado. El más representativo sería `OnKeyPress`, equivalente al `OnClick`, mientras que los eventos `OnKeyDown` y `OnKeyUp` serían similares a `OnMouseDown` y `OnMouseUp`, si bien en este caso indican la pulsación de una tecla en el teclado, y no de un botón del ratón.

Aprovechando el evento `OnKeyPress` es posible, por ejemplo, controlar la introducción de un dato en un control `Edit`, limitando los caracteres, realizando conversiones, etc.

Un caso práctico

En este artículo hemos conocido tres contenedores, dos controles para introducir texto, otros dos para facilitar la selección de opciones, un buen grupo de propiedades y algunos eventos. La mejor forma de saber cuándo y cómo utilizar estos elementos, consiste en introducirlos en un formulario, modificar propiedades, escribir código y, en resumen, hacer pruebas. Comenzaremos proponiendo un ejemplo relativamente sencillo, que le sirva como punto de partida para continuar con su exploración hasta la siguiente entrega.

Usando la técnica de arrastrar y soltar, diseñaremos un formulario similar al que puede verse en la Figura 6. Hemos insertado un `Panel` en la parte inferior, con un `Label` y un `Edit` en su interior. En la parte izquierda de la ficha tenemos un `CheckBox` y un `GroupBox` con tres botones de radio, mientras que a la derecha otro `GroupBox` sirve como delimitador para un control `Memo`. No hemos modificado los nombres de ningún control, dejando los que asigna Delphi por defecto para poder seguir mejor el ejemplo. En la práctica, no obstante, lo habitual es asignar un nombre lógico y descriptivo a cada elemento.

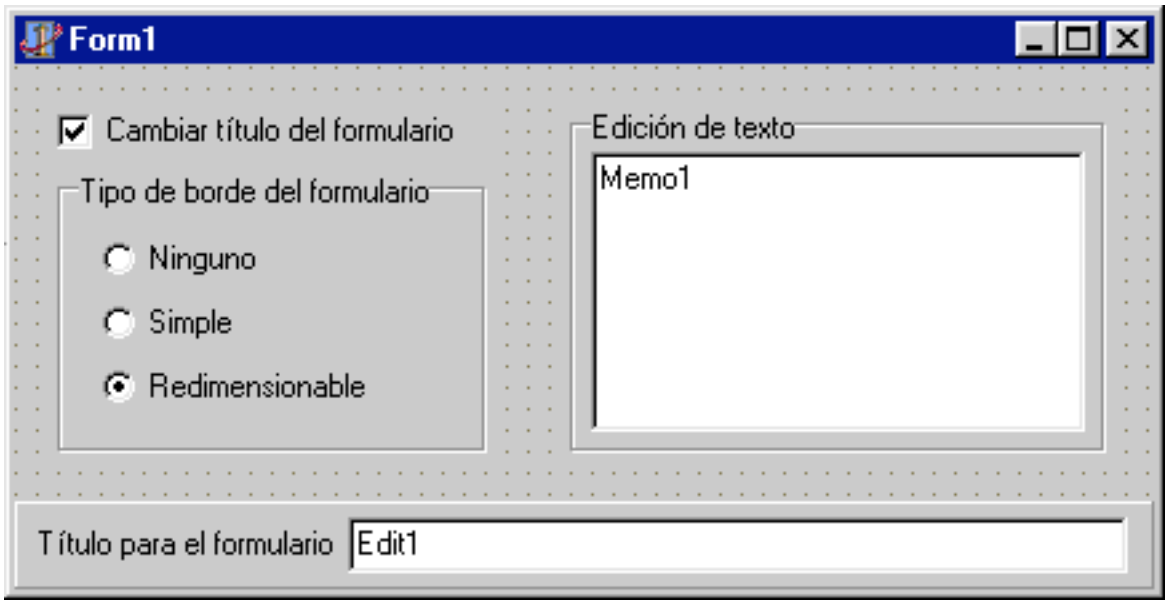

Figura 6. Aspecto del formulario que diseñaremos para probar el funcionamiento de algunos de los controles conocidos.

El panel que hay en la parte inferior podrá estar visible, como lo está inicialmente, u oculto, dependiendo de que el `CheckBox` que hemos dispuesto en la parte superior izquierda esté o no marcado. Tendremos, por tanto, que controlar la pulsación sobre el `CheckBox`, alterando la propiedad `Visible` del `Panel`. Puesto que la etiqueta de texto y el `Edit` se han insertado en el interior del `Panel`, al ocultar éste también se ocultarán aquellos.

Los botones de radio servirán para modificar el tipo de borde de la ventana, entre tres opciones posibles. El control `Memo`, por último, podemos usarlo para introducir texto, comprobar cómo podemos copiar, cortar y pegar, etc. Este control tendrá una característica *especial*, y es que no permitirá la introducción de dígitos numéricos. Con este fin, controlaremos el evento `OnKeyPress` comprobando qué carácter es el que se ha pulsado, anulándolo en caso de ser un dígito numérico.

El código, como puede verse en el Listado 1, es bastante sencillo, consistiendo básicamente sólo en algunas asignaciones. En el evento `OnKeyPress` del `Memo` se ha usado un condicional, un conjunto y un código de carácter. Son elementos que conocerá en la próxima entrega de este curso. Lo que se hace es comprobar si el parámetro `Key`, que contiene el carácter pulsado, almacena un dígito entre 0 y 9 y, de ser así, anularlo asignando al mismo parámetro el código `#0`.

Visto y por ver

Al finalizar esta cuarta entrega ya conocemos el entorno de Delphi, sabemos cómo crear interfaces de usuario insertando y personalizando componentes, cómo escribir código y cómo ejecutar nuestro programa. También sabemos algo acerca de los controles más básicos, y tenemos unos conocimientos generales sobre clases, objetos, propiedades, métodos, eventos y otros conceptos relativos a Object Pascal.

En la próxima entrega conoceremos algunos de los tipos de datos de Object Pascal, haciendo especial hincapié en los más específicos, como los conjuntos y enumeraciones. También tendremos ocasión de aprender las construcciones de control más básicas.

```
// Al cambiarse el texto del Edit1
procedure TForm1.Edit1Change(Sender: TObject);
begin
    Caption := Edit1.Text; // asignarlo al título de la ficha
end;

// Al pulsar sobre el CheckBox
procedure TForm1.CheckBox1Click(Sender: TObject);
begin // mostrar y ocultar el panel inferior
    Panel1.Visible := CheckBox1.Checked;
end;

// Según el botón de radio que se pulse
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
    BorderStyle := bsNone; // establecer un tipo
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
    BorderStyle := bsSingle; // u otro de borde
end;

procedure TForm1.RadioButton3Click(Sender: TObject);
begin
    BorderStyle := bsSizeable; // para la ficha
end;

// Controlar cada pulsación de tecla en el Memo
procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
begin
    if Key In ['0'..'9'] then // si es un dígito numérico
        Key := #0; // no permitirlo
end;
```

Listado 1. Código del programa de ejemplo