

# **Programas de Aplicación III**

## **Tema 3. Creación de interfaces de usuario (Parte 3)**

Luis Rodríguez Baena y María Dorrego Luxán

**Universidad Pontificia de Salamanca (campus Madrid)**

Facultad de Informática

# Control de errores (I)

## ❑ Activar el controlador de errores.

### ● Declaración On Error.

- ✓ Se coloca antes de que se puede producir un error.
- ✓ Captura el error en tiempo de ejecución producido.
- ✓ `On Error Goto etiqueta`, bifurca la ejecución a la etiqueta que se indique.
  - ✗ Una etiqueta es un identificador seguido de dos puntos que empieza en la primera columna y sin nada más detrás.
- ✓ `On Error Resume Next`, ignora el error y pasa a la línea siguiente.
- ✓ `On Error Goto 0`, desactiva el controlador de errores del procedimiento actual.

# Control de errores (II)

## ❑ Detectar el tipo de error producido.

- El objeto `Err` recoge el número de error.
  - ✓ Propiedad `Number` (propiedad por omisión).
    - ✗ Guarda el número de error. Para ver los errores en tiempo de ejecución, buscar en MSDN "Errores interceptables".
    - ✗ Si no se ha producido un error vale 0.
    - ✗ Se restablece cuando acaba un procedimiento o se encuentra con `On Error` o `Resume`.
  - ✓ Propiedad `Description`. Una cadena con la descripción del error.
  - ✓ Método `Clear`. Limpia las propiedades del objeto `Err`.
  - ✓ Método `Raise`. Lanza un error.  

```
Err.Raise número [, source][, description][, helpfile][, helpcontext]
```
- Función `Error`. Devuelve la descripción del error.
- Instrucción `Error número` lanza el error especificado.

# Control de errores (III)

## ❑ Acciones a tomar si se produce un error.

### ● Instrucción Resume.

- ✓ `Resume`, vuelve a ejecutar la orden que ha producido el error.
  - ✖ Se puede utilizar si el usuario puede corregir el error.
    - Por ejemplo, no se ha introducido el disquete.
- ✓ `Resume Next`, salta la instrucción que ha producido el error y ejecuta la siguiente.
- ✓ `Resume etiqueta`, retorna a la etiqueta indicada.

# Colores en VB (I)

- ❑ Las propiedades BackColor, ForeColor, FillColor y BorderColor permite seleccionar un color de:
  - Paleta: color RGB personalizado.
  - Sistema: utiliza la definición de colores utilizada por el sistema operativo.
- ❑ El color se almacena en un entero largo.
  - Por comodidad se utiliza en formato hexadecimal.
  - El byte más significativo se utiliza para indicar que se trata de un color de paleta RGB (primer byte a 0) o del sistema (primer byte distinto de 0).
- ❑ Constantes de color.
  - El núcleo de VB dispone de constantes predefinidas para los colores.
    - ✓ En el examinador de objetos buscar ColorConstants para colores RGB y SystemColorConstants para colores del sistema (de &H80000000 a &H80000015).

# Colores en VB (II)

## □ Colores RGB.

- Del byte menos significativo al más significativo indica la cantidad de rojo, verde y azul del color.
  - ✓ Cada componente se expresa en notación hexadecimal de &H00 a &HFF.
  - ✓ &HAAGGRR
    - × &H0000FF, color rojo: nada de azul, nada de verde y 255 de rojo.
    - × &H00FF00, color verde: nada de azul, 255 de verde y nada de rojo.
    - × &H808080, color gris: valor medio (80) para el azul, el rojo y el verde.
- Función RGB.

`RGB(rojo, verde, azul)`

  - ✓ Devuelve un entero largo, correspondiente a los valores de 0 a 255 de la cantidad de rojo, verde y azul.
  - ✓ RGB(255,0,0) devuelve el número correspondiente al color rojo.

# Arrays de controles (I)

- ❑ Conjunto de controles que comparten el nombre que tienen algún valor no negativo en la propiedad `Index`.
  - Comparten los procedimientos de eventos, aunque los valores de sus propiedades o la acción de sus métodos son individuales.
- ❑ Creación de arrays de controles.
  - Asignar un valor no negativo en la propiedad `Index`.
  - Crear un control con el nombre de otro existente. VB preguntará si se desea crear un array de controles.
  - Seleccionar un control, pulsar `Ctrl+C` para copiarlo al portapapeles y `Ctrl+V` para pegarlo. VB preguntará si se desea crear un array de controles.

# Arrays de controles (II)

## ❑ Ventajas:

- El uso de procedimientos de eventos compartidos permite ahorrar código.
- Se consumen menos recursos que los controles normales.
  - ✓ Evitan la restricción de VB de limitar el número de controles de un formulario a 256.
- Permiten añadir controles de forma dinámica en un formulario.

## ❑ Procedimientos compartidos.

- Cuando se crea un array de controles, en sus procedimientos de eventos aparece un nuevo argumento: Index.  

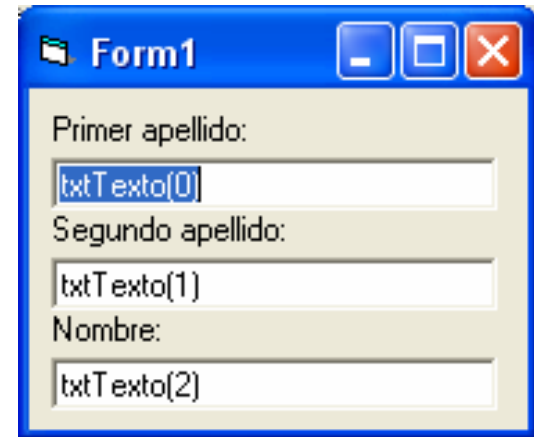
```
Private Sub ArrayControles_Evento(Index As Integer [,argumentos])
```
- El argumento Index indica el índice del control sobre el que se ha producido el evento.



# Arrays de controles (III)

```
Private Sub txtTexto_Change(Index As Integer)
    'Cambia a mayúsculas cualquier texto
    'introducido en cualquier control
    txtTexto(Index).Text = UCase(txtTexto(Index).Text)
    'Esto es necesario para poner el punto
    'de inserción al final del control
    txtTexto(Index).SelStart = _
        Len(txtTexto(Index).Text)
    'Index indica el control que ha cambiado
    Select Case Index
        Case 0
            'Ha cambiado el primer apellido
        Case 1
            'Ha cambiado el segundo apellido
        Case 2
            'Ha cambiado el nombre
    End Select
End Sub

Private Sub txtTexto_GotFocus(Index As Integer)
    'Selecciona el texto del control activo
    txtTexto(Index).SelStart = 0
    txtTexto(Index).SelLength = _
        Len(txtTexto(Index).Text)
End Sub
```



The screenshot shows a standard Windows application window titled "Form1". Inside the window, there are three text boxes arranged vertically. The first text box is preceded by the label "Primer apellido:" and contains the text "txtTexto(0)". The second text box is preceded by the label "Segundo apellido:" and contains the text "txtTexto(1)". The third text box is preceded by the label "Nombre:" and contains the text "txtTexto(2)". The text boxes are white with a thin border, and the labels are in a standard black font.

# Arrays de controles (IV)

## ❑ Propiedades de los arrays de controles.

- `LBound`, devuelve el índice inferior del array.
- `UBound`, devuelve el índice superior del array.
- `Count`, devuelve el número de elementos del array.

## ❑ Recorrido de un array de controles.

```
Dim i as Integer
For i = txtTexto.LBount to txtTexto.UBound
    'Procesar el elemento txtTexto(i)
Next
```

```
Dim txt As TextBox
For Each txt In txtTexto
    'Procesar el elemento txt
Next
```

# Colecciones (I)

- ❑ Agrupación de objetos relacionados.
- ❑ Diferencias con los arrays.
  - No deben dimensionarse (ni redimensionarse) para su utilización.
  - Los elementos pueden insertarse en cualquier lugar.
  - Pueden almacenar objetos heterogéneos.
    - ✓ En un elemento de una colección puede almacenarse cualquier cosa que pueda almacenarse en un dato de tipo Variant.
  - Permiten asociar una clave a cada elemento para su acceso posterior.
  - No es posible modificar un objeto insertado en una colección.
    - ✓ Para modificarlo habrá que borrarlo e insertar el nuevo valor.
  - El trabajo con colecciones es más lento que con arrays.

# Colecciones (II)

## ❑ Creación de una colección.

```
`Declara un objeto de la clase Collection  
`y crea una nueva instancia  
Dim Nombres as Collection  
Set Nombres=New Collection
```

## ❑ Añadir elementos a una colección

### ● Método Add.

*ObjetoColección.Add elemento[,clave,[antes[,despues]]]*

```
Nombres.Add "Pepe","Empleado" `Crea un nuevo elemento  
Nombres.Add "Ana","Jefa" `Crea un elemento después del último  
Nombres.Add "Manolo","Encargado","Empleado"  
                `Crea un elemento antes de "Empleado"  
Nombres.Add "Jaimito","Secretario",,"Empleado"  
                `Crea un elemento después de "Empleado"
```

# Colecciones (III)

## ❑ Acceso a los elementos de una colección.

- Por el índice: Nombres(3), devuelve "Jaimito"
- Por la clave: Nombres("Secretario"), devuelve "Jaimito"
- Nombres!Secretario, devuelve "Jaimito"

## ❑ Eliminar elementos de una colección.

*ObjetoColección.Remove índice*

## ❑ Propiedad Count. Devuelve el número de elementos de una colección.

## ❑ Colecciones predefinidas en Visual Basic.

- Forms, guarda una referencia de cada formulario cargado.
- Controls, guarda una referencia de cada control del formulario.

# Creación dinámica de controles (I)

## ❑ Mediante arrays de controles.

- Crear en tiempo de diseño un control con el índice 0.
- Cargar un nuevo control con un índice mayor que 0.
  - ✓ Load *control*(1).
- Si es necesario, modificar sus propiedades.

```
'Crea un nuevo elemento del array Text1.  
'En el formulario existe un control Text1  
'con la propiedad Index a 0 y la  
'propiedad Visible a False  
Private Sub Command1_Click()  
    Dim max As Integer  
    max = Text1.UBound  
    Load Text1(max + 1)  
    'Lo sitúa justo debajo del anterior  
    Text1(max + 1).Move 0, Text1(0).Height * max  
    Text1(max + 1).Visible = True  
End Sub
```

# Creación dinámica de controles (II)

## ❑ Añadiendo un elemento a la colección Controls.

- El método Add de la colección Controls.

`Controls.Add progID, nombre [, contenedor]`

- ✓ progID, una cadena indicando el tipo de control.

- ✗ El tipo de control se puede obtener mediante el Examinador de objetos.

- ✗ Está formado por un nombre de biblioteca y el nombre del control.

- Control TextBox, "VB.TextBox"

- Control CommandButton, "VB.CommandButton"...

- ✓ nombre, nombre del control.

- ✓ Opcionalmente se puede indicar el objeto contenedor. Si se omite, el contenedor será el formulario.

# Creación dinámica de controles (III)

```
'Crea un nuevo control txtXXX
Private Sub Command3_Click()
    Dim txt As TextBox
    numTextBox = numTextBox + 1
    'El nombre del control será txt y un número correlativo
    Set txt = Controls.Add("vb.TextBox", "txt" & numTextBox)
    txt.Move 0, Text1(0).Height * numTextBox - 1, Text1(0).Width, Text1(0).Height
    txt.Visible = True
End Sub
```

## ❑ Eliminar controles en tiempo de ejecución.

- Arrays de controles.
  - ✓ Instrucción UnLoad: UnLoad Text1(1)
- Colección Controls.
  - ✓ Método Remove.
    - ✗ Controls.Remove "txt3"



# El control FlatScrollBar

- ❑ Control alternativo a HorizontalScrollBar y VerticalScrollBar.
  - Se encuentra en la biblioteca "Microsoft Common Controls-2 6.0"



- ❑ Propiedades (sólo las distintas a las otras barras de desplazamiento)

Nombre	Significado
Appearance	Apariencia del control <ul style="list-style-type: none"><li>● 0 - fsb3D, en tres dimensiones. La habitual de Windows.</li><li>● 1 - fsbFlat, plana</li><li>● 2 - fsbTrack3D, plana. Las flechas y el botón aparecen en tres dimensiones cuando se pasa el ratón por encima.</li></ul>
Orientation	<ul style="list-style-type: none"><li>● 0 - cc2OrientationHorizontal, horizontal.</li><li>● 1 - cc2OrientationVertical, vertical.</li></ul>
Arrows	Devuelve o establece si las flechas están habilitadas. <ul style="list-style-type: none"><li>● 0 – cc2Both, ambas flechas habilitadas.</li><li>● 1 – cc2LeftUp, habilitada la flecha izquierda o superior.</li><li>● 2 – cc2RigthDown, habilitada la flecha derecha o inferior.</li></ul>

# El control UpDown (I)

## ☐ Control de incremento y decremento.

- Se encuentra en la biblioteca "Microsoft Common Controls-2 6.0"



## ☐ Propiedades (nuevas o con distinto significado).

Nombre	Significado
Alignment	Determina la alineación respecto al control auxiliar. <ul style="list-style-type: none"><li>● 0 – cc2AlignmentLeft, se alinea a la izquierda del control auxiliar.</li><li>● 1 – cc2AlignmentRight, se alinea a la derecha del control auxiliar.</li></ul>
Increment	Establece o devuelve un valor que determina la cantidad en que varía la propiedad Value cuando se hace clic en los botones del control UpDown.
Max, Min	Valor máximo y mínimo de la propiedad Value del control.
Orientation	<ul style="list-style-type: none"><li>● 0 - cc2OrientationHorizontal, horizontal.</li><li>● 1 - cc2OrientationVertical, vertical.</li></ul>

# El control UpDown (II)

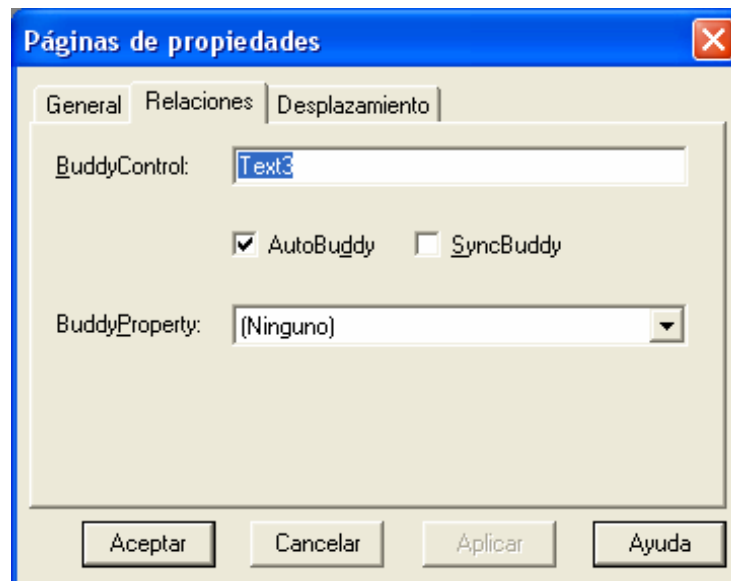
## ❑ Propiedades relacionadas con el control asociado.

- El control UpDown se puede asociar con otro control para sincronizar sus valores.

Nombre	Significado
BuddyControl	Un Variant con el nombre del control asociado (UpDown1.BuddyControl=Text1).
BuddyProperty	Un Variant con la propiedad relacionada del control asociado (UpDown1.BuddyProperty="Text")
AutoBuddy	Un valor lógico. Si es True utiliza como control relacionado el que tenga el número inferior en el orden de tabulación. Si es True utiliza el control de la propiedad BuddyControl.
SyncBuddy	Un valor lógico. Si es True sincroniza el valor del control UpDown con el control asociado.

# El control UpDown (III)

- ❑ Página de propiedades (menú contextual de UpDown, opción propiedades o Personalizado de la ventana de propiedades).
  - Cambia mediante un cuadro de diálogo las propiedades de aspecto, relación o desplazamiento.



- ❑ Nuevos eventos.

Nombre	Significado
UpClick	Se produce cuando se pulsa la flecha hacia arriba.
DownClick	Se produce cuando se pulsa la flecha hacia abajo.