Programando con B4X

Tema 18 – Tipos de datos complejos

Version 1.0, mayo 2021

Autor original: Prokopis Leon

Traducido al español por José Miguel López

⏱4

Lo que los estudiantes aprenderán
• Declaración de un Tipo • Arrays de un Tipo • Listas de un Tipo • Mapas de un Tipo • Ficheros KVS • CustomListView • B4XComboBox

A veces un desarrollador quiere agrupar variables de distinto tipo que están relacionadas. Por ejemplo, un estudiante tiene nombre, apellidos, dirección, teléfono, etc. Sería posible crear variables por separado para cada propiedad, pero es más conveniente crear un nuevo tipo de dato que las agrupe todas.

La sentencia "type"

El tipo de dato que necesitamos se llama **type** y en el ejemplo anterior se declararía así:

> *Type Estudiante (Apellidos As String, Nombre As String, dirección As String, teléfono As String)*

Fíjate que la palabra reservada "type" es la primera que aparece y después se pone el nombre del nuevo tipo de variable que creamos. Seguidamente, entre paréntesis, indicamos todas las variables incluidas en el nuevo tipo de datos. La declaración de tipos se debe poner siempre en la zona **Class_Globals** y es **pública**.

En el ejemplo del estudiante, la sentencia "type" quedaría así:

```
Sub Class_Globals
  Type Estudiante(Apellidos  As String, Nombre  As String, _
                  Dirección As String,  Teléfono  As String)
End Sub
```

Ahora ya tenemos un nuevo tipo de dato llamado "Estudiante" que puede usarse para crear nuevas variables basadas en él.

La variable "Estudiante1" es ahora una variable del tipo "Estudiante" y para acceder a los datos que la componen pondremos el nombre de la variable seguido de un punto y el nombre de la propiedad especificado al crear el nuevo tipo:

```
Estudiante1.Apellidos = "Ioannidis"
Estudiante1.Nombre = "Alkinoos"
Estudiante1.Dirección = "Atenas, Grecia"
Estudiante1.Teléfono = "+303465854234"
```

Puedes crear tantas variables del tipo "Estudiante" que quieras, así como asignar unas a otras:

```
Private Estudiante2 as Estudiante
Estudiante2 = Estudiante1
```

Aquí, la variable "Estudiante2" contendrá exactamente los mismos datos que la variable "Estudiante1".

Mostrar Ítems

Podemos usar el método "Log" para mostrar los contenidos de "Estudiante1", aunque el resultado sería este:

```
Log(Estudiante1)
```

```
[IsInitialized=false, ID=FS23534X21, Apellidos=Ioannidis
, Nombre=Alkinoos, Dirección=Atenas, Grecia, Teléfono=+303465854234
]
```

Para mostrar o usar las propiedades de un tipo se puede usar el nombre junto el nombre del ítem.

Log(Estudiante1. Apellidos & " " & Estudiante1.Nombre)

Es útil también crear una rutina que acepte un tipo y que muestre o procese los elmentos de la variable recibida.

```
Private Sub LogEstudiante(est As Estudiante)
  Log(est.Nombre)
  Log(est.Apellidos)
  Log(est.Dirección)
  Log(est.Teléfono)
End Sub
```

Array de un Tipo

Se pueden crear arrays de un Tipo para poder gestionar más estudiantes. Por ejemplo:

```
Sub Class_Globals
  Type Estudiante(Apellidos  As String, Nombre  As String, _
                  Dirección As String,  Teléfono  As String)

  Public Estudiantes(10)  As Estudiante
End Sub
```

La sentencia Estudiantes(10) rea un array de 10 estudiantes. Cada ítem puede ser usado en un bucle o referenciándose mediante su índice.

Estudiantes(0). Apellidos = "Paul"

Estudiantes(0). Nombre = "Belmond"

Listas de un Tipo

Una lista puede contener variables de un Tipo creado.

```
Sub Class_Globals
  Type Estudiante(Apellidos  As String, Nombre  As String, _
                  Dirección As String,  Teléfono  As String)

  Public listaEstudiantes  As List
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
  ...
  ...
  listaEstudiantes.Initialize
  listaEstudiantes.Add(Estudiante1) 'Añadir estudiante a la lista

  'Obtener la lista de ítems y mostrarlos
  For i = 0 To listaEstudiantes.Size-1
    Private est  As Estudiante
    est = listaEstudiantes.Get(i)
    LogEstudiante(est)
  Next
End Sub

Private Sub LogEstudiante(est  As Estudiante)
  Log(est.Nombre)
  Log(est.Apellidos)
  Log(est.Dirección)
  Log(est.Teléfono)
End Sub
```

Mapas de un Tipo

También se puede usar un mapa para guardar datos de un tipo creado siempre que como clave se use un valor único.

En el caso de los estudiantes, se podría usar como calve única un número identificativo (ID), un número de registro, un correo electrónico, etc.

```
Sub Class_Globals
  Type Estudiante(id As String, Apellidos As String, _
  Nombre As String, Dirección As String, Teléfono As String)

  Public mapaEstudiantes As Map
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)

  Estudiante1.ID  = "FXA47345S3"
  Estudiante1.Apellidos = "Ioannidis"
  Estudiante1.Nombre = "Alkinoos"
  Estudiante1.Dirección = "Atenas, Grecia"
  Estudiante1.Teléfono = "+303465854234"

  mapaEstudiantes.Initialize
  mapaEstudiantes.Put(Estudiante1.ID, Estudiante1)

End Sub
```

De este modo, es muy fácil encontrar un estudiante usando su número identificativo (ID) invocando el método "Get".

Guardar en ficheros KVS

Los ficheros comentados en el tema 17 son ficheros de texto que normalmente no sirve para guardar estructuras complejas como los tipos, listas y mapas. Existe otro tipo de ficheros en B4J llamados ficheros KVS (key value store).

La forma en que funcionan es similar a los mapas. Sólo se necesita un clave y la estructura que quieres guardar. Los ficheros KVS son esencialmente una base de datos, pero lo importante es que para el desarrollador esta gestión es completamente transparente y sólo debe usar los métodos adecuados.

Los métodos en un fichero KVS son los siguientes:

Declaración de un fichero KVS

Para declarar un fichero KVS se debe usar la biblioteca KeyValueStore que debes marcar en la pestaña de bibliotecas (librerías).

Seguidamente, crea un variable de tipo **KeyValueStore.**

```
Private ficheroKVS  As KeyValueStore
```

Inicializar el fichero KVS

En el método Initialize se indicará la carpeta donde se guardará el fichero y su nombre.

```
File.MakeDir(File.DirTemp, "tema18")

ficheroKVS.Initialize(File.DirTemp & "tema18", "kvsData.dat")

Log(File. DirTemp  & "lesson18")
```

La sentencia anterior crea un carpeta llamada "tema18" dentro de la carpeta temporal, crea un fichero llamado "kvsData.dat" y finalmente muestra la ruta al fichero en la pantalla de Log.

Recuerda

El fichero que crees no puede abrirse con otro visor (como Notepad++). Sólo puedes abrirlo con tu programa.

Insertar ítems en un fichero KVS

Se usa el método "Put" para guardar datos en un fichero KVS. Cara inserción se llama "Registro". Es un prerrequisito que se use una clave única para poder referirse a cada registro. El siguiente ejemplo inserta la variable "Estudiante1" usando el número identificativo del estudiante como clave.

```
ficheroKVS.Put(Estudiante1.ID,  Estudiante1)
```

De este modo puedes guardar cualquier tipo de dato como Listas, Mapas, Cadenas de texto, variable simpes (números), tipos y arrays (sólo arrays de bytes u objetos), así como combinaciones (una lista de mapas, por ejemplo).

La declaración de tipos debe hacerse siempre en B4XMainPage.

Los mapas también se pueden guardar usando el método **PutMapAsync**. Es además la mejor forma de guardar un mapa ya que usa la propia clave del mapa como clave para cada registro.

```
ficheroKVS.PutMapAsync(mapaEstudiantes)
```

Recuperar ítems de un fichero KVS

Se usa el método "Get" para obtener ítems de un fichero KVS. El valor devuelto es un objeto. Asegúrate de asignar el valor devuelto a una variable del mismo tipo.

El siguiente ejemplo lee un registro Estudiante de un fichero KVS:

```
Estudiante3 = ficheroKVS.Get("FS23534X21")
LogEstudiante(Estudiante3)
```

El valor "FS23534X21" es la calve del registro que queremos recuperar y el método LogEstudiante recibe un estudiante y lo muestra en pantalla.

Recuerda

Si la clave no existe, entonces tendrás un problema en tu programa. La existencia de un registro debe comprobarse antes de asignar el resultado a una variable. Ello puede hacerse con el método ContainsKey.

Se puede leer un mapa completo usando el método **GetMapAsync.** Este método acepta una lista de claves como parámetro y devuelve un mapa con las claves y sus correspondientes valores.

```
Log("Mostrar claves”)
Private claves  As List = ficheroKVS.ListKeys
For i = 0  To  claves.Size-1
  Log(claves.Get(i))
Next
```

Finalmente, a continuación del código anterior, empleamos GetMapAsync:

```
Wait For (ficheroEstu.GetMapAsync(keys)) Complete (mapSt As Map)
```

Para guarar un mapa en un fichero KVS usaremos:

```
Wait For (ficheroEstu.PutMapAsync(keys)) Complete (Succes As Boolean)
```

Comprobar la existencia de un registro

Para comprobar la existencia de una clave en un fichero KVS se debe usar el método ContainsKey:

```
If ficheroKVS.ContainsKey("FS23534X21")  Then
  Estudiante3 = ficheroKVS.Get("FS23534X21")
Else
  Log("Clave ID errónea")
End If
```

ContainsKey devuelve True si encuentra la clave en el fichero.

Borrar un registro de un fichero KVS

El método “Remove” borra un registro con una clave indicada y el método “DeleteAll” borra todos los registros.

```
ficheroKVS.Remove ("FS23534X21")
ficheroKVS.DeleteAll
```

Recuerda

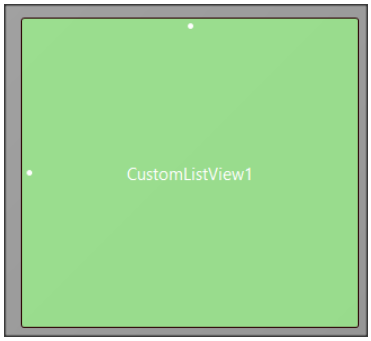
Es una buena práctica cerrar un fichero que ya no vas a usar más con el método .close

Más Views

Dos vistas importantes que te ayudarán a mostrar o elegir elementos de tipos de datos complejos, listas o mapas son **CustomListView** y **B4XComboBox**.

CustomListView

Crea una lista de ítems que puedes elegir con el ratón:



El valor devuelto es el que se haya indicado al crear el ListView.

```
Sub Class_Globals
  Private Root As B4XView
  Private xui As XUI
  Private lstItems As List

  Private CustomListView1 As CustomListView      1
  Private lblFecha As Label
End Sub

Private Sub B4XPage_Created (Root1 As  B4XView)
  Root = Root1
  Root.LoadLayout("MainPage")

  lstItems.Initialize
  lstItems.AddAll(Array As String("Lunes", "Martes",  _                2
"Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"))

  For i = 0 To lstItems.Size-1
    CustomListView1.AddTextItem(lstItems.Get(i), i)      3
  Next

End Sub

Private Sub CustomListView1_ItemClick(Índice As Int, Valor As  4
Object)
  lblFecha.Text = Valor
End Sub
```

1. Declarar una variable del tipo CustomListView.
2. Crear una lista que contendrá todos los ítems que aparecerán en la CustomListView

3. Recorrer la lista y colocar cada ítem en la CustomListView1 usando el método **AddTextItem.**
   AddTextItem recibe un valor que mostrará y un índice que corresponde con el valor que quieres mostrar. El ejemplo anterior corresponden a los valores 0 a 6 para representar los días del lunes al domingo.
4. Cuando se hace clic en un ítem de la lista, se dispara el evento **_ItemClick.** El ejemplo muestra el índice del ítem pulsado en la etiqueta lblFecha.

Otros métodos para CustomListView son:

- **Clear** As String
  Borra todos los componentes de la CustomListView.
- **GetValue** (Índice As Int) As Object
  Devuelve el valor que hay en el índice indicado como parámetro.
- **Size** As Int [sólo lectura]
  Devuelve el número de ítems.

Marcar un ítem de una lista

Como se indicó antes, al hacer clic en un ítem de una lista se dispara el evento _ItemClick. Para que el ítem se quede marcado, el programador debe usar alguna herramienta proporcionada por el lenguaje (por ej. La biblioteca CLVSelections) o escribir su propio código. El siguiente algoritmo supone que has creado una variable en **Class_Global** llamada **ítemElegido** de tipo Int.

```
Private Sub CustomListView1_ItemClick (Índice As Int, Valor As Object)
   If ítemElegido = -1 Then
      Private p As B4XView = CustomListView1.GetPanel(Índice)          1
      p.GetView(0).Color = xui.Color_Blue
      ítemElegido = Índice
   Else
      Private p As B4XView = CustomListView1.GetPanel(selectedItem)    2
      p.GetView(0).Color = xui.Color_White
      If ítemElegido = Índice Then
         ítemElegido = -1
      Else
         Private p As B4XView = CustomListView1.GetPanel(Índice)       3
         p.GetView(0).Color = xui.Color_Blue
         ítemElegido = Índice
      End If
   End If
End Sub
```

Inicialmente, ítemElegido se pone a -1 para indicar que no se ha elegido nada. El código emplea dos sentencias:

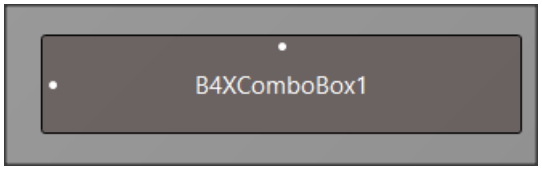
```
Private p As B4XView = CustomListView1.GetPanel(Índice)
p.GetView(0).Color = xui.Color_Blue
```

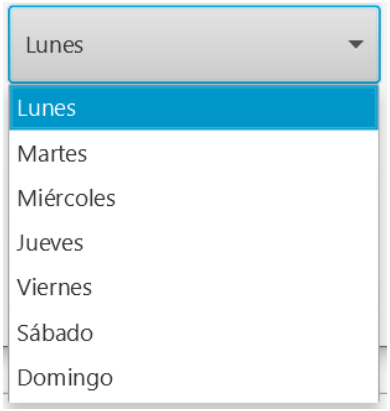
Cada ídem de una lista se crea dentro de una caja llamada "panel". Puedes fijar su color accediendo a la caka mediante el método **GetPanel(Índice)** donde Índice indica el valor de la línea que ha sido pulsada. La sentencia **p.GetView(0).Color** fija el color que el programador desea usar. Así:

1. Si se hace clic en un ítem de la lista, la rutina comprueba el valor de **ítemElegido** y, si vale -1, entonces pone de color azul el fondo de la línea elegida y guarda en **ítemElegido** el valor del índice que devuelve el evento **_ItemClick**
2. Si **ítemElegido** ya contiene un valor, se pone el color de fondo de la línea de color blanco y después tenemos 2 opciones:
    a. Si hemos elegido el mismo ítem, deseleccionamos el ítem y la variable **ítemElegido** se pone a -1.
    b. Si hemos elegido otro ítem, el nuevo ítem se pone de color azul y la variable **ítemElegido** se pone con el valor del índice.

B4XComboBox

B4XComboBox muestra una lista de ítems desplegable. El usuario podrá elegir cualquier de ellos. Al contrario que con CustomListView, el valor devuelto siempre es un número que indica la posición que ocupa el ítem dentro del ComboBox, con lo que el programador debe ser capaz de procesar ese correctamente.





```
Sub Class_Globals
   Private Root As B4XView
   Private xui As XUI
   Private lstItems As List
   Private B4XComboBox1 As B4XComboBox                                    1
   Private lblCmbFecha As Label
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
   Root = Root1
   Root.LoadLayout("MainPage")
   lstItems.Initialize
   lstItems.AddAll(Array As String("Lunes", "Martes",  _                  2
"Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"))

   B4XComboBox1.SetItems(lstItems)                                        3
End Sub


Private Sub B4XComboBox1_SelectedIndexChanged(Índice As  Int             4
   lblCmbFecha.Text  = Index
End Sub
```

1. Declarar una variable de tipo **B4XComboBox.**
2. Crear una lista que contenga los ítems que aparecerán en el B4XComboBox
3. Colocar los ítems en la lista B4XComboBox. Cuidado, porque no es necesario un bucle como en CustomListView
4. Si se elige un ítem, se dispara el evento **_SelectedIndexChanged** y devuelve el índice del elemento elegido.

Ejercicios

1. Crea un tipo llamado **Cliente** con las variables siguientes:

   ID, Nombre, Apellidos, Teléfono

2. Crea una lista de Clientes con los siguientes ítems:

Id	Nombre	Apellidos	Teléfono
A3473	John	Smith	4563454
B1753	Selim	Al Huarizmi	6532578
C6544	Mateo	Sandor	7345346
C6323	Lucía	Graham	1231345
F1462	Noam	Bell	6978323

3. Guarda la lista en un fichero KVS.
4. Crea un botón que al pulsarse rellene la CustomListView con la información de los clientes anteriores.
5. Crea un botón insertar que, al pulsarse, muestre un DialogBox que pida la información de un nuevo cliente y lo añada a la CustomListView.
6. Guarda el nuevo cliente también en el fichero KVS.
7. Cuando el usuario haga clic en un cliente se mostrará una ventana para confirmar que se desea borrar ese cliente. Si se confirma el borrado, se eliminará de la CustomListView y del fichero KVS.