

Objetivo: realizar consultas sobre contenido XML

- realizar consultas que devuelven colecciones de objetos XElement
- acceder a los valores de los elementos de dicha colección

Ejemplo resuelto

Trabajaremos a partir de un documento XML cargado en memoria y manipularemos sus datos. En primer lugar, contamos con un documento XML al que se ha llamado "`xmlfile1.xml`" que contiene la siguiente información:

```
<?xml version="1.0"?>
<datosGenerales>
  <datosPersona>
    <nombre>Fernando</nombre>
    <edad>18</edad>
  </datosPersona>
  <datosPersona>
    <nombre>Rosa</nombre>
    <edad>17</edad>
  </datosPersona>
  <datosPersona>
    <nombre>Carlos</nombre>
    <edad>21</edad>
  </datosPersona>
</datosGenerales>
```

El siguiente código muestra cómo cargar XML en memoria desde el archivo y cómo una vez cargado, efectuar una selección de sus datos. El ejemplo en ejecución, dará como resultado los valores "Fernando" y "Carlos".

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
    'obtener origen de datos
    Dim doc As XDocument = XDocument.Load("xmlfile1.xml")

    'crear consulta
    Dim Selección = _
        From datos In ^doc...<datosPersona>
        Where datos.<nombre>.Value Like "??r*"
        Select datos.<nombre>.Value

    'ejecutar consulta
    For Each s In Selección
        MessageBox.Show(s)
    Next
End Sub
```

¹ Origen de datos: los descendientes <datosPersona>

Entendiendo el ejemplo:

La expresión `doc...<datosPersona>` usa el llamado "eje de descendientes" y básicamente dice: **"busque todos los descendientes denominados `<datosPersona>`".**

En XML, un descendiente es un elemento anidado **en uno o más niveles** por debajo del elemento actual. Observa que el XML del ejemplo tiene un nodo raíz denominado `datosGenerales` y que contiene los elementos `datosPersona`. La sintaxis de **tres** puntos se traduce básicamente en `doc.Descendants("datosGenerales")`.

La expresión `datos.<nombre>.Value:` usa la propiedad `Value`. Puede que te estés preguntando por qué aquí necesitamos llamar explícitamente a **`.value`** antes de hacer la comparación. La razón de ello es que **`datos.<nombre>` devuelve una colección de objetos `XElement`**; un `IEnumerable(Of XElement)`. La propiedad de `.value` proporciona acceso al **valor del primer elemento de dicha colección**.

Las siguientes consultas proporcionarán el mismo resultado:

```
'Dim selección =  
'    From datos In doc...<nombre>  
'    Where datos.Value Like "??r*"  
'    Select datos.Value  
  
'Dim selección =  
'    From datos In doc.Root...<nombre>  
'    Where datos.Value Like "??r*"  
'    Select datos.Value  
  
'Dim selección =  
'    From datos In doc.Root...<datosPersona>.<nombre>  
'    Where datos.Value Like "??r*"  
'    Select datos.Value
```

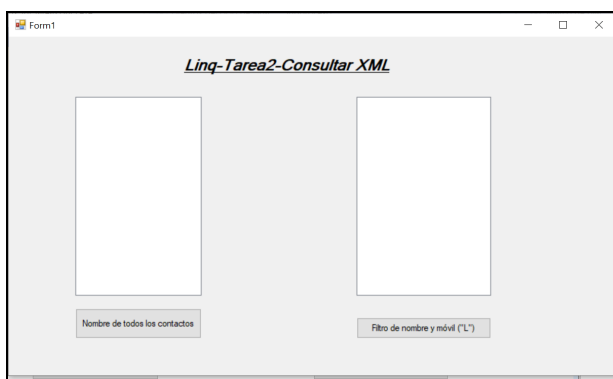
Puesta en práctica

A partir de los datos de **Contactos.xml** creado en Linq-Tarea1 mostrar:

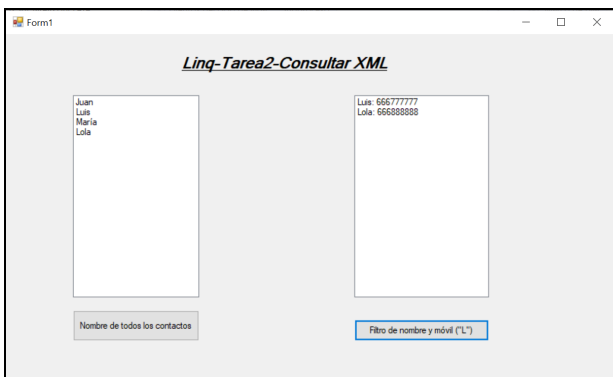
- en un **listBox** los nombres de todos los contactos del archivo y
- en otro **ListBox** los nombres y el teléfono móvil de aquellos cuyo nombre comience con "L".

Ojo!, móvil es un atributo del elemento <Telefono>

(<Telefono Móvil="66666666" Trabajo="911111111" />)



The screenshot shows a window titled "Form1" with the subtitle "Linq-Tarea2-Consultar XML". It contains two list boxes. The left list box is labeled "Nombre de todos los contactos" and is currently empty. The right list box is labeled "Filtro de nombre y móvil ('L')" and is also empty.



The screenshot shows the same window as above, but now the list boxes are populated. The left list box, labeled "Nombre de todos los contactos", contains the names: Juan, Luis, Maria, and Lola. The right list box, labeled "Filtro de nombre y móvil ('L')", contains the names and mobile numbers: Luis: 656777777 and Lola: 656300000. The right list box is highlighted with a blue border.