

Cap. 12 - Navegaciones en Flex 4

Después de explorar la estructura general de los diferentes componentes de flex, y como los eventos se relacionan con diferentes partes de la aplicación. Un aspecto importante en este enfoque de trabajo son las navegaciones; o sea, la forma en que interactúan los distintos componentes.

Componentes de Navegación:

Menú

MenuBar

ViewStack

ButtonBar

TabNavigator

Accordion.

12.1 Preparando Datos

Antes de mostrar cómo funcionan estos componentes, veremos distintas formas de preparar los datos para cargar en los componentes.

12.1.1 - Arrays Anidados

```
protected var menuData:Array =
[
    {
        label: 'New',
        children: [{label: 'Task'}, {label: 'Request'}, {label: 'Person'}]
    },
    {
        label: 'Import',
        children: [{label: 'Image'}, {label: 'Document'}, {label: 'Project'}]
    }
];
```

12.1.2 - ArrayCollection anidados

Similar a los array anidados pero no están anidados al tipo.

```
import mx.collections.ArrayCollection;
protected var menuData:Array =
[
    {
        label: 'New',
        children: [{label: 'Task'}, {label: 'Request'}, {label: 'Person'}]
    },
    {
        label: 'Import',
        children: [{label: 'Image'}, {label: 'Document'}, {label: 'Project'}]
    }
];
protected var menuCollection:ArrayCollection =
new ArrayCollection(menuData);
```

12.1.2.1 - MXML

La sintaxis para estos tipos de estructura es mucho más amigable en mxml.

Nota: cualquier nombre para los nodos hijos de <fx:Object> son válidos.

```
<s:ArrayCollection id="menuCollection">
  <fx:Array>
    <fx:Object label=" 'New' ">
      <fx:children>
        <fx:Array>
          <fx:Object label="Task" />
          <fx:Object label="Request" />
          <fx:Object label="Person" />
        </fx:Array>
      </fx:children>
    </fx:Object>
    <fx:Object label=" 'Import' ">
      <fx:children>
        <fx:Array>
          <fx:Object label="Image" />
          <fx:Object label="Document" />
          <fx:Object label="Project" />
        </fx:Array>
      </fx:children>
    </fx:Object>
  </fx:Array>
</s:ArrayCollection>
```

12.1.3 - Models

Models es otra forma de cargar los datos a un componente de navegación. Es muy similar al xml visto anteriormente pero éste tiene mucho menos niveles de anidamiento.

Algunas ventajas de utilizar models son:

- Bajo nivel de anidamiento.
- Fácil de codificar y visualizar
- Se puede configurar un archivo property para utilizar como fuente de datos.

```
<fx:Model id="menuData">
  <menuinfo>
    <menuitem label="Task">
      <children label="Request" />
      <children label="Person" />
    </menuitem>
    <menuitem label="Import">
      <children label="Image" />
      <children label="Document" />
      <children label="Project" />
    </menuitem>
  </menuinfo>
</fx:Model>
```

Nota: models, toman la ventaja de los XML pero no son XML en sí mismos. Existen algunas ocasiones en que se necesita utilizar XML.

12.1.4 - Componentes XML

Así como los arrays anidados, los XML no son componentes basados en colecciones de objetos, por tal motivo, cualquier cambio realizado en los datos no se propagarán por los eventos del sistemas a los cuales estén atados.

Unas de las ventajas que tienen estos componentes es que cada nodo se puede nombrar como el desarrollador desee.

Así como los models, también pueden usar como fuente de datos desde un archivo property.

```

<fx:XML id="menuData">
  <menuinfo>
    <menuitem label="Task">
      <submenu label="Request"/>
      <submenu label="Person"/>
    </menuitem>
    <menuitem label="Import">
      <submenu label="Image"/>
      <submenu label="Document"/>
      <submenu label="Project"/>
    </menuitem>
  </menuinfo>
</fx:XML>

```

12.1.5 – Componente XMLList

Otra de las formas de setear datos a un menú es usando XMLList. Este componente no tiene grandes diferencias con los XML vistos anteriormente, pero esta forma que nos provee Action Script de hacer lo mismo.

```

<fx:XMLList id="menuData">
  <menuitem label="Task">
    <submenu label="Request"/>
    <submenu label="Person"/>
  </menuitem>
  <menuitem label="Import">
    <submenu label="Image"/>
    <submenu label="Document"/>
    <submenu label="Project"/>
  </menuitem>
</fx:XMLList>

```

12.1.6 – Componente XMLListCollection

Este componente actúa como un encapsulador de XML pero a su vez se le suman las propiedades de una collection. Debido a que este componente está basado en colecciones, cualquier cambio en los datos, automáticamente son transmitidos a los diferentes componentes que están diseñados para utilizar esos datos.

El ejemplo que sigue, muestra un XMLListCollection encapsulando un XMLList para poblarlo con datos.

```

<s:XMLListCollection id="menuData">
  <fx:XMLList>
    <menuitem label="Task">
      <submenu label="Request"/>
      <submenu label="Person"/>
    </menuitem>
  </fx:XMLList>
</s:XMLListCollection>

```

A su vez podemos utilizar el anterior XMLListCollection como fuente de datos:

```

<fx:XML id="config">
  <configData>
    <appData>
      <appName name="My Application"/>
    </appData>
    <menuData>
      <menuitem label="Task">
        <submenu label="Request"/>
        <submenu label="Person"/>
      </menuitem>
    </menuData>
  </configData>
</fx:XML>
<s:XMLListCollection id="menuData" source="{config.menuData.menuitem}"/>

```

Lo particular del código anterior, es que se podría guardar en un archivo separado y únicamente necesitaríamos cargar nuestro menú con los datos de ese archivo en particular.

```

<fx:XML id="config" source="config.xml" />
<s:XMLListCollection id="menuData" source="{config.menuData.menuitem}"/>

```

12.2 Trabajando con Menus

Después de explorar las diferentes formas de cargar y preparar los datos, estamos capacitados para explorar los componentes de menus.

Los que tipos de menus que podemos encontrar en Flex 4 son:

- **Menu**
- **MenuBar**
- **ViewStack**
- **ButtonBar**
- **TabNavigator**
- **Accordion**

12.2.1 – Creando el menú.

Un menú puede contener desde una simple ventana con algunas opciones, o hasta una gran cantidad de submenús con diferentes niveles que desplieguen más y más opciones. Para poder cubrir algunos de estos puntos, comencemos con la creación de un menú.

El primer menú que veremos se encuentra en MXML y utiliza un XMLListCollection como proveedor de datos.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
    xmlns:mx="library://ns.adobe.com/flex/halo">
    <s:layout>
        <s:VerticalLayout gap="0" />
    </s:layout>
    <fx:Declarations>
        <mx:XMLListCollection id="menuData">
            <mx:source>
                <fx:XMLList>
                    <menuitem label="Tasks">
                        <submenu label="Add Request"/>
                        <submenu label="Add Person">
                            <submenu label="Customer"/>
                            <submenu label="Employee"/>
                        </submenu>
                    </menuitem>
                </fx:XMLList>
            </mx:source>
        </mx:XMLListCollection>
    </fx:Declarations>

    <s:Button label="Display Menu" click="menu.show()"/>
    <mx:Menu id="menu" showRoot="true" labelField="@label" dataProvider="{menuData}" />
</s:Application>
```

El resultado sería algo así:



Además de los datos generados que se agregan al menú (ya vistos en el capítulo anterior) , en el ejemplo se puede ver claramente como se crea el menú (mx:menu) y los parámetros que se le pasan a la clase:

Id – es el identificador del objeto menú.

showRoot – boolean que indica si se desea mostrar el elemento raíz dentro del menú.

labelField – indica la etiqueta. El símbolo de @ denota el nodo del XML. Para el caso de los **arrays** no debería usarse este símbolo.

Otra alternativa a la creación del menú sería utilizando `ActionScript` en lugar de XML. Para esto haríamos la siguiente invocación.

```
var menu:Menu = Menu.createMenu(null, menuData,true);
```

Los tres parámetros que recibe la operación son: `parent`(opcional), `dataProvider`, `showRoot`.

12.2.2 – Posicionando el menú.

Después de haber creado el menú, debemos posicionarlo. Para esto, la misma función `show` nos permite indicar parámetro que indican la posición en la cual se va a colocar el menú.

```
menu.show(10,20);
```

De manera más general podemos decir que los parámetros son:

```
menu.show(xShow,yShow);
```

12.2.3 – Customizando los ítems del menú.

Como cualquier software que utiliza un menú, podemos notar una cantidad de estilos diferentes en ellos. Con `flex` podemos customizar nuestros menus para darles un aspecto más atractivo y diferente.

A continuación veremos una lista de algunos de los atributos existentes:

enabled / Boolean : Habilita y deshabilita un ítem.

Icon / Class: no usar en radios, checks o separator. Para estos existen los `checkIcon` y `radioIcon`.

label /String: texto especificado en el control.

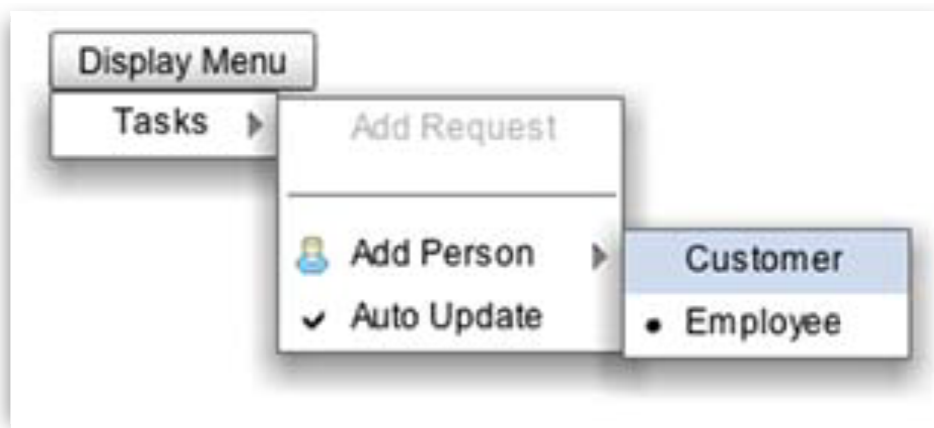
toggled / boolean: especifica si un check o radio es seleccionado.

type / String: especifica el tipo del ítem del menú.

A continuación veremos un ejemplo de un menú customizado:

```
<?xml version="1.0"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
    xmlns:mx="library://ns.adobe.com/flex/halo">
    <s:layout>
        <s:VerticalLayout gap="0" />
    </s:layout>
    <fx:Script>
        <![CDATA[
            [Bindable]
            [Embed(source="assets/icons/user.png")]
            public var userIcon:Class;
        ]]>
    </fx:Script>
    <fx:Declarations>
        <mx:XMLListCollection id="menuData">
            <fx:XMLList>
                <menuitem label="Tasks">
                    <submenu label="Add Request" enabled="false"/>
                    <submenu type="separator"/>
                    <submenu label="Add Person" icon="userIcon">
                        <submenu label="Customer" type="radio"
                            groupName="persons"/>
                        <submenu label="Employee" type="radio" groupName="persons"
                            toggled="true"/>
                    </submenu>
                    <submenu label="Auto Update" type="check" toggled="true"/>
                </menuitem>
            </fx:XMLList>
        </mx:XMLListCollection>
    </fx:Declarations>
    <s:Button label="Display Menu" click="menu.show()"/>
    <mx:Menu id="menu" showRoot="true" labelField="@label" iconField="@icon"
        dataProvider="{menuData}" />
</s:Application>
```

Resultado del código anterior:



12.2.4 – Interactuando con el menú.

Uno de los temas más importantes para desarrollar una aplicación es el cómo interactuar entre los diferentes componentes. Un menú necesita responder a acciones del usuario llamados eventos. Estos eventos, especialmente aquellos definidos por la clase Menu, son despachados como MenuEvents, los cuales llevan información de la acción del usuario para guiar a la función encargada de manejar el evento.

Para ver más claro esto describiremos un ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  xmlns:mx="library://ns.adobe.com/flex/halo">
  <s:layout>
    <s:VerticalLayout gap="0" />
  </s:layout>
  <fx:Script>
    <![CDATA[
      import mx.events.MenuEvent;
      private function onMenuClick(event:MenuEvent):void
      {
        var item:XML = XML(event.item);
        lastEvent.text = "Selection: " + item.@label + ", Position:
        " + event.index +
        " Type: " + item.@personType;
      }
    ]]>
  </fx:Script>
  <fx:Declarations>
    <mx:XMLListCollection id="menuData">
      <fx:XMLList>
        <menuitem label="Tasks">
          <submenu label="Add Request" enabled="false"/>
          <submenu label="" type="separator"/>
          <submenu label="Add Person" icon="userIcon">
            <submenu label="Customer" type="radio"
              groupName="persons"
              personType="32"/>
            <submenu label="Employee" type="radio"
              groupName="persons"
              toggled="true" personType="57"/>
          </submenu>
          <submenu label="Auto Update" type="check"
            toggled="true"/>
        </menuitem>
      </fx:XMLList>
    </mx:XMLListCollection>
  </fx:Declarations>

  <mx:Button label="Display Menu" click="menu.show()"/>
  <mx:Menu id="menu" showRoot="true" labelField="@label"
    iconField="@icon"dataProvider="{menuData}" itemClick="onMenuClick(event)" />

  <mx:Spacer height="10" />
  <mx:Label id="lastEvent"/>
</s:Application>

```

Después de hacer click en un ítem del menú, el itemClick se dispara y llama a la función onMenuClick pasándole por parámetro el evento del tipo MenuEvent. A partir de aquí, dentro de la función onMenuClick, podemos obtener información importante gracias al evento pasado por parámetro como por ejemplo qué ítem disparó el evento, que posición del menú ocupaba y cualquier otra información detrás del ítem.

Nota: Como podemos ver en el código, para poder acceder a atributos del item, debemos poner el parámetro @ antes del nombre de la propiedad debido a que el tipo es un XML.

El uso de MenuEvent nos permite reutilizar código para manejar los menús en nuestra aplicación. Este es un primer paso para el estudio de navegaciones en flex, pero a pesar de esto el componente Menu consta de limitaciones debido a que solo podemos procesar un solo nodo raíz. Para resolver este problema podemos utilizar el MenuBar.