

Resumen Capitulo 9 - Usando Listas Spark

Introduccion

En el capitulo anterior se vio como utilizar las listas simple de flex, ahora veremos como utilizarlas con la librería Spark.

La librería de Spark permite mas posibilidades de escalabilidad, por ejemplo, permite que un usuario avanzado de Flex modifique el diseño de la interfaz.

Genealogía

Como se vio en el capitulo 8 los componentes de la librería MX son derivados de ListBase y AdvancedListBase. En Spark el componente extiende de la ListBase de Spark ubicada en el paquete spark.components.supportClasses y esta a su vez extiende de SkinnableDataContainer.

Lo importante aquí es comprender que cada clase hereda de una clase ListBase propia de su librería sea MX o sea Spark.

Identificando el componente correspondiente

Flex automáticamente sabe cual de los namespaces usar, si en el mxml se encuentra el tag <mx:List/> usara la ListBase de la librería MX, si por el contrario utiliza <s:List/> usara la clase ListBase de la librería de Spark.

Elementos renderizados con las listas spark

En MX y Spark List extends ListBase, en MX ListBase extiende de ScrollControlBase la cual extiende de UIComponent. Por el contrario la ListBase de Spark extiende de SkinnableDataContainer la clase que es responsable de mostrar la informacion de los items usando item renderers.

Controles de las Listas Spark

No es difícil crear una custom List in flex 4, es tan fácil como versiones anteriores del SDK, la mejor forma es extender la clase ListBase de Spark o la clase SKinnableDataContainer.

Control ButtonBar

EL control ButtonBar es similar a el ToggleButtonBar de la librería MX, es basicamente una Lista ya que extiende de la clase ListBase, un menú es fácil de crear usando el control de Spark Button-Bar.

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/mx"
```

```

minWidth="1024" minHeight="768">
<s:layout>
    <s:VerticalLayout paddingLeft="20" paddingTop="20"/>
</s:layout>
<s:ButtonBar>
    <mx:ArrayCollection>
        <fx:String>Button One</fx:String>
        <fx:String>Second Button</fx:String>
        <fx:String>Button Three</fx:String>
        <fx:String>Last Button</fx:String>
    </mx:ArrayCollection>
</s:ButtonBar>
</s:Application>

```

Control Spark List

Cuando se declara una lista de spark, la siguiente declaracion debe ser `<s:dataProvider>`. como en flex 4 se la mayoría de las propiedades de los componentes se pueden declarar usando tags MXML se puede incluir la propiedad en la definición de la lista sin tener que poner tag embebido.

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/mx"
    xmlns:comp="com.danorlando.components.*"
    minWidth="600" minHeight="450">
<s:layout>
    <s:VerticalLayout gap="1" useVirtualLayout="true" />
</s:layout>
<s>List>
    <s:dataProvider>
        <mx:ArrayCollection>
            <fx:String>Menu Item 1</fx:String>
            <fx:String>Second Menu Item</fx:String>
            <fx:String>Third Menu Item</fx:String>
            <fx:String>Fourth Menu Item</fx:String>
            <fx:String>Last Menu Item</fx:String>
        </mx:ArrayCollection>
    </s:dataProvider>
</s>List>
</s:Application>

```

Control DropDownList

Como los otros componentes de Spark basados en lista, la informacion de estos puede especificarse

usando la propiedad data provider, a continuacion se muestra un codigo de ejemplo para el uso del DropDownList:

```
<?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/mx"
    minWidth="1024" minHeight="768">
    <s:layout>
        <s:VerticalLayout paddingLeft="20" paddingTop="20"/>
    </s:layout>
    <s:DropDownList width="200">
        <mx:ArrayCollection>
            <fx:String>Item One</fx:String>
            <fx:String>Second Item</fx:String>
            <fx:String>Item Three</fx:String>
            <fx:String>Last Item</fx:String>
        </mx:ArrayCollection>
    </s:DropDownList>
</s:Application>
```

Interactuando con las Listas Spark

La interacción de los componentes basados en listas de spark son similares a los componentes de MX porque los controles de Spark originan de la clase UIComponent, clase base de la librería MX.

Evento por defecto en selección de items

La interacción es similar a la librería MX, las principales diferencias son que los nombres de los eventos y los tiempos en los cuales estos son lanzados, por ejemplo, cuando un item es seleccionado en una lista MX, se lanzan los eventos ListEvent.CHANGE y ListEvent.ITEM_CLICK. En Spark la selección de un elemento resulta en tres eventos distintos: selectionChanging, itemFocusChanged y selectionChanged

Objeto 'IndexChangedEvent'

El objeto que encapsula estos tres eventos que se lanzan para Spark se llama IndexChangedEvent, además un evento itemFocusChanged es disparado después que el foco cambia de un item a otro. El objeto para itemFocusChanged es también un IndexChangedEvent. El tercer evento IndexChangedEvent es disparado cuando un nuevo ítem es seleccionado, llamando al evento selectionChanging. Este ultimo evento es enviado antes que la información del ítem actual cambia. Esta evento da la posibilidad de llamar al método preventDefault() en el event handler para prevenir la selección cuando cambia.

Comprendiendo la arquitectura de las Listas

La clave para comprender la arquitectura de los componentes de Spark es primero comprender que estos están hechos de la misma infraestructura que los componentes de MX.

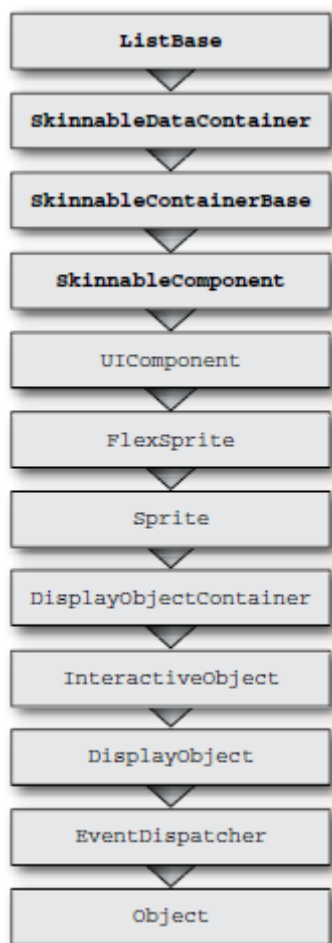
Jerarquía de clases

Flex 4 esta hecho como Flex 3 por lo tanto permite compatibilidad hacia atrás y simple portabilidad e las aplicación hechas con Flex 3 y convertirlas a Flex 4 con mínimo esfuerzo.

Esto es así porque los componentes en Flex derivan de UIComponent, y esta encapsula las funcionalidades básicas de Flex. Con Sparl sucede lo mismo ya que es la mejor opción al momento de crear los custom components basados en esa librería.

Clases que agrega Spark

Los mostramos en el siguiente diagrama:



Construir componentes Custom con Spark

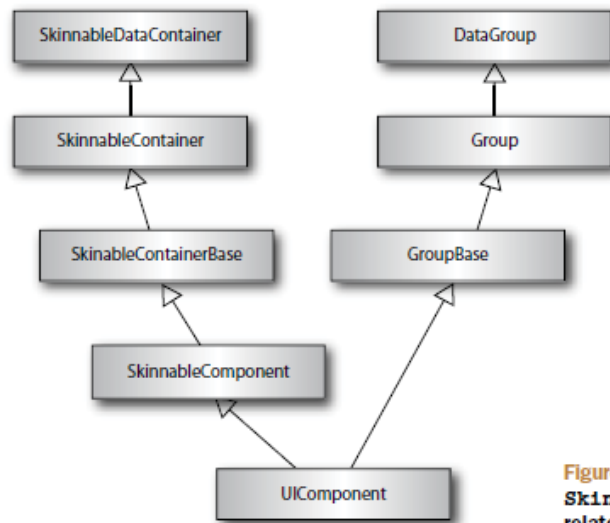


Figure 9.6 How Group and SkinnableComponent are related to UIComponent

Comprendiendo las clases Group y SkinnableContainer

El contenedor Group va a tomar cualquier componente que implemente la interfaz IUIComponent como hijo y cualquier component que implemente IGraphicElement. Use este contenedor cuando tenga que manejar visualizaciones secundarias, componentes visuales y componentes gráficos pero que no quiera sea personalizable.

Para eso utiliza la clase SkinnableConainer en lugar de Group, ambas clases son abstractas las cuales deben ser directamente extendidas en casos especiales.

Para los componentes de Spark estas existen pero se llaman SkinnableDataConainer y DataGroup.

Construyendo un componente custom

A continuación un ejemplo el cual presenta la funcionalidades descritas anteriormente:

```
<?xml version="1.0"?>
<s:Panel title="Spark List Component Example"
  xmlns:fx="http://ns.adobe.com/mxml/2009"
  xmlns:s="library://ns.adobe.com/flex/spark"
  width="75%" height="75%"
  horizontalCenter="0"
  verticalCenter="0">
  <fx:Script>
    <![CDATA[
      import mx.events.IndexChangedEvent;
      private function selectionChangingHandler(evt:IndexChangedEvent):void
      {
        var item:* = list.dataProvider.getItemAt(evt.newIndex);
        if (item.type != "travel")
        {
```

```

        evt.preventDefault();
    }
}
]]>
</fx:Script>
<s:VGroup left="20" right="20" top="20" bottom="20">
    <s:SimpleText text="Select a title to see the image."/>
    <s:List id="list"
        selectionChanging="selectionChangingHandler(event);"
        width="100%" height="100%" lineHeight="22">
        <s:layout>
            <s:VerticalLayout/>
        </s:layout>
        <s:itemRenderer>
            <fx:Component>
                <s:ItemRenderer>
                    <s:states>
                        <s:State name="normal" />
                        <s:State name="hovered" />
                        <s:State name="selected" />
                    </s:states>
                    <s:Rect left="0" right="0" top="0" bottom="0">
                        <s:fill>
                            <s:SolidColor color="0x999999" alpha="0"
                                alpha.hovered="0.2"
                                alpha.selected="0.4" />
                        </s:fill>
                    </s:Rect>
                    <s:SimpleText id="titleLabel" text="{data.title}"/>
                    <s:BitmapImage horizontalCenter="80" id="img"
                        source="{data.image}"
                        includeIn="selected" />
                </s:ItemRenderer>
            </fx:Component>
        </s:itemRenderer>
        <s:dataProvider>
            <s:ArrayList>
                <fx:Object type="travel" title="San Francisco"
                    image="images/san_fran.jpg" />
                <fx:Object type="travel" title="San Francisco Lightrail"
                    image="images/san_fran_lighrail.jpg" />
                <fx:Object type="travel" title="San Francisco Carriage"
                    image="images/san_fran_carriage.jpg" />
            </s:ArrayList>
        </s:dataProvider>
    </s:List>
</s:VGroup>

```

</s:Panel>