

LISTAS PERSONALIZADAS

La mejor manera de trabajar con componentes basados en listas es aprendiendo el manejo del mapeo de datos de los campos provenientes de una colección de objetos (ArrayCollection o XMLListCollection) para luego poder manipularlos con las funciones de los campos (label functions).

-Propiedades de los LabelFields(Campo de etiquetas):

Los LabelFields indican que valores mostrar en el campo de la colección pedida.

-LabelFunctions (Funcion de etiqueta)

Son una propiedad de los componentes tales como: list, combo-box, datafields, con el fin de indicar la forma en la cual se despliegan los valores obtenidos de la colección y en las cuales también se puede agregar más funcionamiento que solo mostrar los datos como se recibieron.

- Tipos de Funciones de etiqueta

Existen dos tipos :

-**Los de columna simple**, son aquellas que aceptan un parámetro solo en la función: List, HorizontalList, TitleList and Trees.

-**Los de columnas múltiples**, aceptan dos parámetros en la función, uno para la fila y el otro para la columna: Listas Basadas en componentes, DataGrids, Advanced Data Grid, Print Data Grid, File System Data Grid y OLAP Data Grid.

-Usos de las Funciones de etiquetas:

*Concatenar datos de los campos de un objeto para crearlos y mostrarlos como un simple STRING.

*Dar formato a los datos en bruto.

*Convertir los datos en bruto en un formato legible para el usuario.

Item Renderers

Se encarga de darle al programador un control muy fino sobre la visualización de los datos en los elementos. Existen 3 tipos de elementos renderizables: **regular**, **inline** y **drop-in**.

- Regular Item Render:

Dentro de la librería de Spark se encuentran los objetos `ItemRenderer`, que pueden ser declarados como nodos de raíz de un elemento renderizado MXML.

CONSEJO: Con FLEX si deseas crear un elemento Spark renderizado en un MXML pero no quieres usar el objeto `ItemRenderer` como ruta del nodo del archivo MXLM, puedes usar `Group`, `BorderContainer`, `SkinnableComponents`, `SkinnableContainer`, `Hgroup`, `VGroup` u otro contenedor de Spark. Implementando `IListItemRenderer` (para renderers List-based) o `IDropInListItemRenderer` o `IItemRenderer` si no necesitas agregar funcionalidad a la lista.

Flex utiliza el control del campo como su elemento renderizado por defecto, pero cuando se especifica un elemento renderizado se sobrescribe el elemento por defecto por su personalización renderizada.

Lo logra de la siguiente forma: - Flex crea una instancia de renderización para cada uno de los objetos de la colección. - Como cada uno de los elementos renderizados es creado, el evento `creationComplete` es activado. - Para que una función que este escuchando (Listener Functions) verifique que el siguiente objeto tenga creada su renderización, sino lo fue, lo crea insertando sus datos apropiadamente al objeto, para que todo valor llamado pueda ser extraído como se necesite. Las instancias de renderización se pueden re usar permitiendo un ahorro de memoria.

- InLine Item Render

La renderización ya no es desacoplada como un componente separado, sino que es codificada como un hijo del contenedor padre usando la MXML tag `<itemRenderer/>`. Esto nos da buenos resultados en la prototipación rápida de interfaces de prueba de conceptos, pero no es lo mejor para aplicaciones que pasan a la etapa de Producción.

- DropIn Item Renderer

Cuando presentamos controles con elementos con las propiedades renderizables y editables (`itemrenderer` y `itemeditor`) se dice que estamos usando Drop-In ítem renderer o Drop-In Itemeditor. Para poder utilizarlos es necesario implementar sus interfaces. (`IDropInItemRenderer`).

Algunos componentes que soportan la renderización en el DropIn:

`Button`; `CheckBox`; `DataField`; `Image`; `label`; `Text`; `TextArea`; `TextInput`.

ITEM EDITORS

La renderización entra en estado editable cuando la edición es habilitada (la propiedad `editable=true`) y el usuario activa el evento de edición (al hacer doble click o click sobre el elemento por ejemplo).

-Habilitar la edición de un Item:

Por defecto la característica de edición está deshabilitada para los elementos renderizados.

Procedimiento de ejecución: Cuando el usuario se focaliza sobre el elemento que desea editar con click, tab o enter, la renderización pasa a ser edición, el elemento editable interactúa con el usuario para obtener la nueva información. Luego el usuario termina la edición con tab, enter, etc. El valor desplegado es actualizado. A su vez la variable que contiene los datos del `DataProvider` (proveedor de datos de la colección), es también actualizada con el nuevo valor disparando el correspondiente EVENTO (`OnChange` (cuando se detecta algún cambio)) para que sea escuchado por otros.

Como crear un elemento editable:

-Se puede crear un archivo MXML externo con el fin de ser invocado por cualquiera que lo desee.

-Se puede crear una variable para que sea usada por componentes List-Based, para saber donde será guardado el nuevo valor. (los tipos de datos entre variable y datos del componente deben ser los mismo número-número, carácter-carácter, etc).

-Cuando inicia la edición del elemento, se debería llenar el campo editable con los valores deseados, cargando a su vez la variable.

-Eventos de edición de un elemento:

Flex dispara muchos eventos durante la edición que pueden ser interceptados para agregar lógica de negocio, parámetros, limpiar rutinas o agregar puntos de control (checkpoints).

- `itemEditBeginning`: Se dispara cuando el proceso de edición comienza. Por ejemplo click en una celda, apretar enter sobre una celda, tab cuando paso de una celda a otra. Se utiliza principalmente para agregar lógica de negocio para determinar si se va a editar la celda.
- `itemEditBegin`: Ocurre inmediatamente después de del evento anterior, cuando los datos son pasados al elemento editable. Se utiliza para manipular dichos datos si es necesario.
- `itemEditEnd`: Ocurre cuando el usuario valida la terminación de la edición en el elemento. El estado de edición continúa pero está a punto de terminar. Se utiliza para comparar y validar el valor actual con el valor nuevo a insertar.

Los eventos manejados son relativos a los componentes que los utilizan por ejemplo: ListEvents son usados por Spark o componentes MX List-Based. DataGridEvents son usados por los DataGrids junto con los ListEvents. Los AdvancedDataGridEvents son usados por los AdvancedDataGrid junto con los ListEvents.

-Propiedades de los eventos de un objeto:

columnIndex;currentTarget;itemRenderer;reason;rowIndex;type;dataField;PreventDefault

-Edicion con Renderizacion(rendererIsEditor)

Cuando un componente es editable y renderizable a la vez, se habla de un componente **hibrido**.

Se utiliza cuando deseo que un componente pueda cambiar de solo lectura a lectura-escritura o al revés. Generalmente las aplicaciones que utilizan esto necesitan que se mantenga el estado de edición porque los datos están cambiando constantemente y quiero que ese cambio sea rápido. Los componentes más apropiados para lo antes mencionado son los AdvancedDataGrid.