CONTROLES DE SELECCIÓN

Programación multimedia
y dispositivos móviles
2º DAM

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

- Se consideran ViewGroup a los menús y a los listados.
- Los listados son los controles de selección, entre ellos están los Spinner, los GridView y los ListView.
- Los controles de selección son un conjunto de elementos ordenados por los que es posible navegar utilizando una barra de Scroll.
- Al seleccionar o pulsar uno de los elementos de la lista, se desencadenará un evento, que es posible controlar y al que se puede responder.
- Para dar un aspecto personalizado a los listados en cualquiera de sus variantes (Spinner, GridView, ListView) se podrá hacer creando unos adaptadores personalizados, y que veremos en el próximo tema.

2. SPINNER

El widget Spinner de Android muestra una lista desplegable para seleccionar un único elemento, y es equivalente a ComboBox de java.

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Una vez declarado el Spinner en xml, tendremos que indicar desde java que valores contiene. A estos valores, se les llama adaptador.

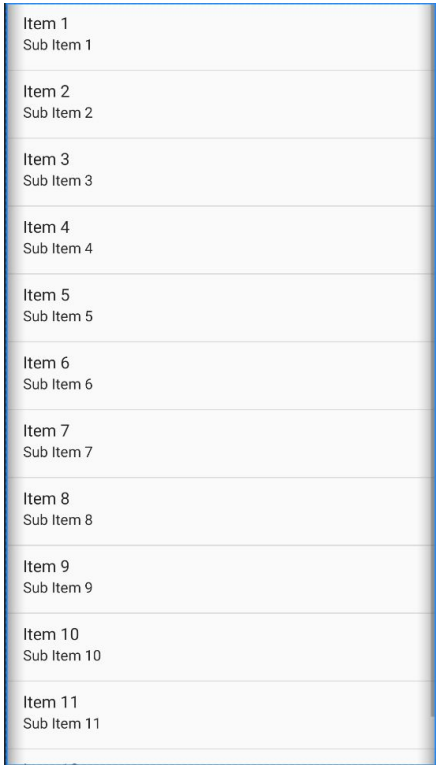
3. LISTVIEW

Este objeto visualiza una lista deslizable verticalmente de varios elementos. Al igual que spinner, cada uno de los elementos de la lista puede ser seleccionado sobre el propio control. En caso de disponer de más elementos de lo que es posible mostrar en pantalla, aparecerá automáticamente un Scroll que permitirá acceder a los elementos no visibles.

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Una vez declarado el ListView en xml, tendremos que indicar desde java que valores contiene. A estos valores, se les llama adaptador.

3. LISTVIEW



Al indicar el ListView en el xml, la vista previa se verá así.

3. LISTVIEW

Algunos atributos para modificar la estética del ListView:

- **android:divider** → permite configurar el color de la línea divisoria.
- **android:dividerHeight** → configura el grosor de la línea divisoria.
- **android:footerDividersEnabled** → habilita el divisor que va delante del “footer” (último)
- **android:headerDividersEnabled** → habilita el divisor que va delante “header” (primero)

4. GRIDVIEW

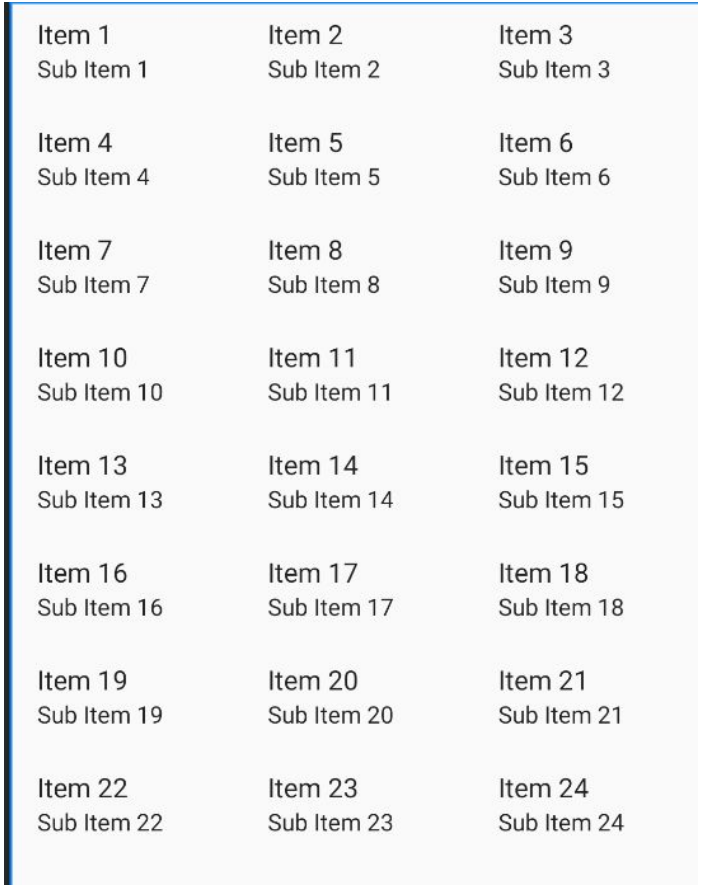
Muestra los datos a modo de rejilla bidimensional e incluye automáticamente un Scroll para cuando los datos ocupen más tamaño que las posibilidades de la pantalla.

Si no se indica el número de columnas, será lo mismo que un ListView.

```
<GridView
    android:id="@+id/gridView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numColumns="3" />
```

Una vez declarado el GridView en xml, tendremos que indicar desde java que valores contiene. A estos valores, se les llama adaptador.

4. GRIDVIEW



Como hemos indicado 3 columnas, se verán los ítems distribuidos en forma de tabla en la pantalla.

Sólo hay que indicar las columnas, puesto que las filas se calcularán automáticamente con el número de elementos que tenga.

4. GRIDVIEW

Atributos importantes de GridView:

- **android:numColumns** → número de columnas que deseamos establecer en el GridView..
- **android:columnWidth** → define un ancho de cada columna en la cuadrícula.
- **android:verticalSpacing** → separación vertical entre las filas del GridView.
- **android:horizontalSpacing** → separación horizontal entre las columnas del GridView.
- **android:stretchMode** → define el modo en que se extenderán las columnas.
    - **None** → ninguna extensión.
    - **spacingWidth** → se extiende al espacio entre cada columna.
    - **columnWidth** → se hace un reparto equitativo del espacio.
    - **spacingWidthUniform** → se hace un reparto uniforme.
- **android:gravity** → define el posicionamiento del contenido en cada celda.

5. ADAPTADORES

Los adaptadores son comunes a todos los controles de selección.

Existen diferentes subclases para trabajar los adaptadores:

- **ArrayAdapter**
- **BaseAdapter**
- **CursorAdapter** → este último se utiliza para la gestión de las consultas de bases de datos.

De momento, en los adaptadores sencillos vamos a trabajar únicamente con la clase ArrayAdapter.

Los adaptadores sencillos son aquellos que tienen como ítem un único texto.

5. ADAPTADORES

Pasos para crear un adaptador:

- Definimos los datos
- Definimos el control de selección.
- Definimos el adaptador.
- Añadimos el adaptador al control de selección.
- Establecemos los listeners necesarios al control de selección.

5.1. DEFINICIÓN DE LOS DATOS

Los datos, puesto que en los adaptadores sencillos son un único texto, se crearán con un array de String.

Este array se puede crear en Java:

```
String[] miArray = new String[] {"Lengua", "Matemáticas", "Inglés"};
```

Pero también se puede crear como un recurso xml:

- Crear un archivo xml en res/values y en este fichero es donde creamos el array de String.

```
<resources>
    <string-array name="asignaturas">
        <item>Lengua</item>
        <item>Matemáticas</item>
        <item>Inglés</item>
    </string-array>
</resources>
```

5.2. DEFINICIÓN DEL CONTROL DE SELECCIÓN

Al igual que cualquier otro componente estudiado, tenemos que crear un objeto de la misma clase del control de selección, y establecer en el método **onCreate** a qué id del xml se corresponde.

```
Spinner spinnerArrayJava, spinnerArrayRecursos;
ListView listViewArrayJava, listViewArrayRecursos;
GridView gridViewArrayJava, gridViewArrayRecursos;
```

```
spinnerArrayJava = findViewById(R.id.spinner1);
spinnerArrayRecursos = findViewById(R.id.spinner2);
listViewArrayJava = findViewById(R.id.listView1);
listViewArrayRecursos = findViewById(R.id.listView2);
gridViewArrayJava = findViewById(R.id.gridView1);
gridViewArrayRecursos = findViewById(R.id.gridView2);
```

5.3. DEFINICIÓN DEL ADAPTADOR

Para este tipo de adaptadores, utilizaremos la clase ArrayAdapter.

A la hora de crear el adaptador, tenemos que indicar 3 parámetros:

- Los datos con los que vamos a trabajar.
- El tipo de layout que queremos utilizar para cada ítem. Para este tipo de adaptadores (sólo un texto por ítem), podremos utilizar los layouts disponibles de android.
- El contexto.

Dependiendo del tipo de datos que vamos a utilizar (array Java o string-array de recursos), se realizará el adaptador de forma diferente.

5.3. DEFINICIÓN DEL ADAPTADOR

Para crear el adaptador con datos de un array en Java, lo haremos con el constructor de ArrayAdapter.

Orden de los parámetros:

- Contexto
- Layout
- Datos

```
ArrayAdapter<String> adaptadorJavaSpinner = new ArrayAdapter<String>( context: this, android.R.layout.simple_spinner_item, miArray);
ArrayAdapter<String> adaptadorJavaListView = new ArrayAdapter<String>( context: this, android.R.layout.simple_list_item_1, miArray);
ArrayAdapter<String> adaptadorJavaGridView = new ArrayAdapter<String>( context: this, android.R.layout.simple_list_item_checked, miArray);
```

5.3. DEFINICIÓN DEL ADAPTADOR

Para crear el adaptador usando los recursos, lo haremos con el método **createFromResources** de ArrayAdapter.

Orden de los parámetros:

- Contexto
- Datos
- Layout

```
ArrayAdapter<CharSequence> adaptadorRecursosSpinner= ArrayAdapter.createFromResource( context: this, R.array.asignaturas, android.R.layout.simple_dropdown_item_1line);
ArrayAdapter<CharSequence> adaptadorRecursosListView = ArrayAdapter.createFromResource( context: this, R.array.asignaturas,  android.R.layout.simple_list_item_1);
ArrayAdapter<CharSequence> adaptadorRecursosGridView = ArrayAdapter.createFromResource( context: this, R.array.asignaturas, android.R.layout.simple_list_item_checked);
```

5.4. AÑADIR EL ADAPTADOR AL CONTROL DE SELECCIÓN

Una vez definido tanto el adaptador como el control de selección, tenemos que indicar que el control de selección tendrá asociado un adaptador, mediante el método **setAdapter**.

```
spinnerArrayJava.setAdapter(adaptadorJavaSpinner);
spinnerArrayRecursos.setAdapter(adaptadorRecursosSpinner);
listViewArrayJava.setAdapter(adaptadorJavaListView);
listViewArrayRecursos.setAdapter(adaptadorRecursosListView);
gridViewArrayJava.setAdapter(adaptadorJavaGridView);
gridViewArrayRecursos.setAdapter(adaptadorRecursosGridView);
```

5.5. AÑADIR LISTENER AL CONTROL DE SELECCIÓN

Como cualquier componente, podemos establecer una serie de acciones cuando el usuario interacciona con el control de selección.

Las interfaces con las que se trabaja son:

- Spinner → OnItemSelectedListener
- ListView → OnItemClickListener / OnItemLongClickListener
- GridView → OnItemClickListener / OnItemLongClickListener

5.5. AÑADIR LISTENER AL CONTROL DE SELECCIÓN

La interfaz **OnItemSelectedListener** tiene dos métodos obligatorios:

- **onItemSelected** → método que se activa cuando se selecciona un ítem
- **onNothingSelected** → método que se activa cuando la selección desaparece. Nunca se desencadena.

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

}

@Override
public void onNothingSelected(AdapterView<?> parent) {

}
```

AdapterView<?> parent → adaptador sobre el que se ha seleccionado.

View view → componente sobre el que se ha seleccionado.

int position → índice del elemento pulsado.

long id → id de la fila seleccionada.

5.5. AÑADIR LISTENER AL CONTROL DE SELECCIÓN

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    String componente = "";

    if (parent.getId() == R.id.spinner1)
        componente = "Spinner Java";
    else if (parent.getId() == R.id.spinner2)
        componente = "Spinner Recursos";

    Toast.makeText( context: this,  text: "Componente: " + componente + "\nContenido: " + parent.getAdapter().getItem(position), Toast.LENGTH_SHORT).show();
}
```

5.5. AÑADIR LISTENER AL CONTROL DE SELECCIÓN

La interfaz **OnItemClickListener** sólo tiene un método obligatorio:

- **onItemClick** → método que se activa cuando se selecciona un ítem

Al igual que la interfaz **OnItemLongClickListener** con el método **onItemLongClick.**

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

}
```

AdapterView<?> parent → adaptador sobre el que se ha seleccionado.

View view → componente sobre el que se ha seleccionado.

int position → índice del elemento pulsado.

long id → id de la fila seleccionada.

5.5. AÑADIR LISTENER AL CONTROL DE SELECCIÓN

```java
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String componente = "";

    if (parent.getId() == R.id.listView1)
        componente = "ListView Java";
    else if (parent.getId() == R.id.listView2)
        componente = "ListView Recursos";
    else if (parent.getId() == R.id.gridView1)
        componente = "GridView Java";
    else if (parent.getId() == R.id.gridView2)
        componente = "GridView Recursos";

    Toast.makeText( context: this, text: "Componente: " + componente + "\nContenido: " + parent.getAdapter().getItem(position), Toast.LENGTH_SHORT).show();
}
```

5.5. AÑADIR LISTENER AL CONTROL DE SELECCIÓN

En cualquiera de los componentes anteriores, hay que asociar la interfaz al componente de la misma forma que aprendimos con el resto de componentes.

```
spinnerArrayJava.setOnItemSelectedListener(this);
spinnerArrayRecursos.setOnItemSelectedListener(this);
listViewArrayJava.setOnItemClickListener(this);
listViewArrayRecursos.setOnItemClickListener(this);
gridViewArrayJava.setOnItemClickListener(this);
gridViewArrayRecursos.setOnItemClickListener(this);
```

6. LAYOUTS ANDROID

Como hemos podido ver, android contiene diferentes vistas para visualizar los datos.

Listado completo:

https://developer.android.com/reference/android/R.layout

7. LAYOUTS PROPIOS

Los layouts de android son bastante normales, por lo que si queremos que los ítems tengan una vista diferente (color de fondo, color de texto, tamaño de texto, estilo…) podemos crear nosotros el layout.

Para ello, tenemos que crear un nuevo layout en el que el elemento padre sea un TextView.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="40sp"
    android:textColor="@color/colorPrimaryDark"
    android:backgroundTint="@color/colorPrimary">

</TextView>
```

```
ArrayAdapter<CharSequence> adaptadorRecursosListView = ArrayAdapter.createFromResource( context: this, R.array.asignaturas,  R.layout.layout_listview);
```

8. DUDAS Y PREGUNTAS