



Controles de Selección



Controles de Selección

- Son básicamente listas de opciones, existen diferentes tipos con análoga funcionalidad, los más destacables son los Spinner o listas desplegadas, ListView o listas fijas y los GridViews o tablas. Además, existen otros componentes específicos de la plataforma, como las galerías de imágenes(Gallery).
- Todos estos controles tienen una peculiaridad y es que necesitan un conjunto de datos para mostrar las diferentes opciones, para que esta tarea sea lo más cómodo y uniforme posible se usan los llamados Adapters, que son interfaces para manejar los datos y sus vistas dentro de los controles de selección.



Controles de Selección (Adapters)

- Un Adaptador(Adapter) es un objeto que permite definir el modelo de datos que usan todos los componentes de selección de forma unificada. Es decir, todos los componentes de selección acceden a los datos que contienen a través de un adaptador.
- El Adaptador, además de suministrar datos a los componentes visuales, también es responsable de generar las vistas específicas que se muestran dentro del componente de selección. Por ejemplo, si cada opción de una lista estuviera formada por una imagen y varias etiquetas, el Adaptador se encarga de generar el contenido de todas estas “opciones” diseñadas.



Controles de Selección (Adapters)

- Android dispone de varios tipos de adaptadores sencillos, aunque es posible extender fácilmente mediante herencia su funcionalidad para mejorarlos según las necesidades de la aplicación. Los más comunes son los siguientes:
 - **ArrayAdapter:** Es el más sencillo de todos los adaptadores. Suministra datos a un componente de selección mediante una matriz de objetos de cualquier tipo. El más utilizado.
 - **SimpleAdapter:** Se usa para definir las opciones de un componente de selección con los datos de un fichero XML. Mapea datos a partir de un layout definido en XML. Menos utilizado.
 - **SimpleCursorAdapter:** Se utiliza para obtener las opciones de un componente de selección de la respuesta de una consulta a una base de datos. Mapea desde un cursor los elementos del control. Menos utilizado.

Controles de Selección (Adapters)

Ejemplo:

```
// Matriz con las opciones del Spinner
String[] data = new String[]{"Opción1", "Opción2", "Opción3"};

// Adaptador que usamos para indicar al Spinner dónde obtiene las opciones
ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, data);
```



Controles de Selección (Adapters)

- Se define un array con los elementos que deseamos, a continuación se crea un Adapter que contiene 3 parámetros:
 - El context se refiere normalmente a la actividad donde se crea el adaptador. Hace referencia al Activity donde tenemos definido el Adapter y por tanto, salvo alguna excepción, será “this”.
 - El ID de la Vista que va a mostrar la opción seleccionada por el usuario. En este caso, usamos un ID de diseño predefinido por Android (`android.R.layout.simple_spinner_item`) y formado por un componente `TextView`.
 - La matriz de los datos que definen las opciones de la lista. De esta forma, ya hemos creado el adaptador para mostrar las opciones seleccionables y sólo hay que asignarlo al componente de selección adecuado.



Spinner

- Es la típica lista desplegable que al escoger una opción queda seleccionada en el control. Solo cabe destacar la propiedad `setAdapter()` para asignarle los datos.
- Aunque tiene una peculiaridad y es que en este caso si hemos creado el Adapter como hemos explicado antes solo va a aplicar el estilo al elemento seleccionado y no a la lista desplegable, para personalizar dicha lista usaremos la propiedad `setDropDownViewResource(ID_layout)`, el ID usado en el ejemplo es uno ya definido en Android para estas listas.



Spinner

- Como evento importante tiene el `onItemSelected()` que se implementa y funciona de igual manera que los eventos vistos en otros elementos salvo que también definiremos el método `onNothingSelected` para que se ejecute cuando el Adapter no contenga datos.
- El evento de una Lista desplegable normalmente es `onItemSelected`, que se invoca cuando el usuario selecciona una opción de la lista. El manejo del evento es similar a otros componentes usando el método `setOnItemSelectedListener()`.
- En este evento definimos dos métodos:
 - El primero de ellos es `onItemSelected` y se invoca cada vez que cambia la selección de la lista desplegable;
 - El segundo es `onNothingSelected`, que se invoca cuando no hay ninguna opción seleccionada (esto ocurre si el adaptador no tiene datos).

Spinner

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

ControlesBasicos

Mercury

Venus

Earth

Mars

Jupiter

Saturn

Uranus

Neptune

Spinner

```
List<String> list = new ArrayList<String>();
list.add("Mercury");
list.add("Venus");
list.add("Earth");
list.add("Mars");
list.add("Jupiter");
list.add("Saturn");
list.add("Uranus");
list.add("Neptune");

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, list);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

```
spinner.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {
        String item = (String) adapterView.getItemAtPosition(position);
        Toast.makeText(getApplicationContext(), "Selected: " + item, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {
        Toast.makeText(getApplicationContext(), "Nothing Selected", Toast.LENGTH_LONG).show();
    }
});
```

ListView

- La lista de selección(ListView) permite al usuario hacer clic sobre una lista de opciones seleccionables. Estas opciones se muestran directamente sobre el propio componente; por lo tanto no se trata de una lista emergente como el Spinner.
- Si no se visualizan todas las opciones disponibles en la pantalla del dispositivo por que no caben se puede desplazar el listado con el dedo o los botones de navegación.
- Su evento fundamenta es el onItemClick.

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




ListView

```
listView = (ListView) findViewById(R.id.listview);

List<String> list = new ArrayList<String>();
list.add("Mercury");
list.add("Venus");
list.add("Earth");
list.add("Mars");
list.add("Jupiter");
list.add("Saturn");
list.add("Uranus");
list.add("Neptune");

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, list);
listView.setAdapter(adapter);
```



ListView Personalizado

- En muchos casos necesitamos que cada elemento de la lista esté formado a su vez por varios elementos (imágenes, textos, checkbox).
- Para ello necesitamos crear una clase java para contener nuestros datos de prueba. Animal.

ListView Personalizado

```
public class Animal {  
    private String nombre;  
    private int drawableImageID;  
  
    public Animal(String nombre, int drawableImageID) {  
        this.nombre = nombre;  
        this.drawableImageID = drawableImageID;  
    }  
  
    public String getNombre() { return nombre; }  
  
    public void setNombre(String nombre) { this.nombre = nombre; }  
  
    public int getDrawableImageID() { return drawableImageID; }  
  
    public void setDrawableImageID(int drawableImageID) { this.drawableImageID = drawableImageID; }  
}
```


ListView Personalizado

- En cada elemento de la lista queremos mostrar el nombre, posición e imagen, por lo que el siguiente paso será crear un layout XML con la estructura que deseemos. En nuestro caso vamos a mostrarlo en dos etiquetas de texto (TextView) y una imagen (ImageView).


```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" >

    <ImageView
        android:id="@+id/imgAnimal"
        android:layout_width="75dp"
        android:layout_height="75dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:src="@android:drawable/ic_dialog_info" />

    <TextView
        android:id="@+id/tvField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/imgAnimal"
        android:layout_marginLeft="10dp"
        android:text="TextView" />

    <TextView
        android:id="@+id/tvContent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/imgAnimal"
        android:layout_toRightOf="@+id/imgAnimal"
        android:layout_marginLeft="10dp"
        android:text="TextView" />

</RelativeLayout>
```



ListView Personalizado

- Ahora que ya tenemos creados tanto el soporte para nuestros datos como el layout que necesitamos para visualizarlos, lo siguiente que debemos hacer será indicarle al adaptador cómo debe utilizar ambas cosas para generar nuestra interfaz de usuario final. Para ello vamos a crear un fichero que extienda de ArrayAdapter.

ListView Personalizado

```
public class ListAdapter extends ArrayAdapter<Animal> {
    private final Activity context;
    private final List<Animal> items;

    public ListAdapter(Activity context, List<Animal> items) {
        super(context, R.layout.item_listview, items);
        this.context = context;
        this.items = items;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View view = inflater.inflate(R.layout.item_listview, null);

        Animal item = items.get(position);

        ImageView imagen = (ImageView) view.findViewById(R.id.imgAnimal);
        imagen.setImageResource(item.getDrawableImageID());

        TextView nombre = (TextView) view.findViewById(R.id.tvContent);
        nombre.setText(item.getNombre());

        TextView numCelda = (TextView) view.findViewById(R.id.tvField);
        numCelda.setText(String.valueOf(position));

        return view;
    }
}
```



ListView Personalizado

- Lo primero que encontramos es el constructor para nuestro adaptador, al que sólo pasaremos el contexto (que será la actividad desde la que se crea el adaptador). En este constructor tan sólo guardaremos el contexto para nuestro uso posterior y llamaremos al constructor padre, pasándole el ID del layout que queremos utilizar y el array que contiene los datos a mostrar.
- Posteriormente, redefinimos el método encargado de generar y rellenar con nuestros datos todos los controles necesarios de la interfaz gráfica de cada elemento de la lista. El método es **getView()**.



ListView Personalizado

- El método `getView()` se llamará cada vez que haya que mostrar un elemento de la lista. Lo primero que debe hacer es “inflar” el layout XML que hemos creado. Esto consiste en consultar el XML de nuestro layout y crear e inicializar la estructura de objetos java equivalente. Para ello, crearemos un nuevo objeto `LayoutInflater` y generaremos la estructura de objetos mediante su método `inflate(id_layout)`.
- Una vez tenemos definido el comportamiento de nuestro adaptador la forma de proceder en la actividad principal será análoga a lo ya comentado, definiremos el array de datos de prueba, crearemos el adaptador y lo asignaremos al control mediante `setAdapter()`.

ListView Personalizado

```
public class CustomListActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_list);

        ListView listView = (ListView) findViewById(R.id.listview);

        List<Animal> items = createListItems();

        ListAdapter adapter = new ListAdapter(this, items);
        listView.setAdapter(adapter);
    }

    private List<Animal> createListItems() {
        List<Animal> items = new ArrayList<>();
        items.add(new Animal("aguiła", R.drawable.aguiła));
        items.add(new Animal("ballena", R.drawable.ballena));
        items.add(new Animal("caballo", R.drawable.caballo));
        items.add(new Animal("camaleón", R.drawable.camaleón));
        items.add(new Animal("canario", R.drawable.canario));
        items.add(new Animal("cerdo", R.drawable.cerdo));
        items.add(new Animal("delfín", R.drawable.delfín));
        items.add(new Animal("gato", R.drawable.gato));
        items.add(new Animal("iguana", R.drawable.iguana));
        items.add(new Animal("lince", R.drawable.lince));
        items.add(new Animal("lobo", R.drawable.lobo_9));
        items.add(new Animal("morena", R.drawable.morena));
        items.add(new Animal("orca", R.drawable.orca));
        items.add(new Animal("perro", R.drawable.perro));
        items.add(new Animal("yaca", R.drawable.yaca));
        return items;
    }
}
```


ListView Personalizado

- El evento onItemClick permite realizar una acción cuando pulsamos sobre uno de los elementos de la lista.

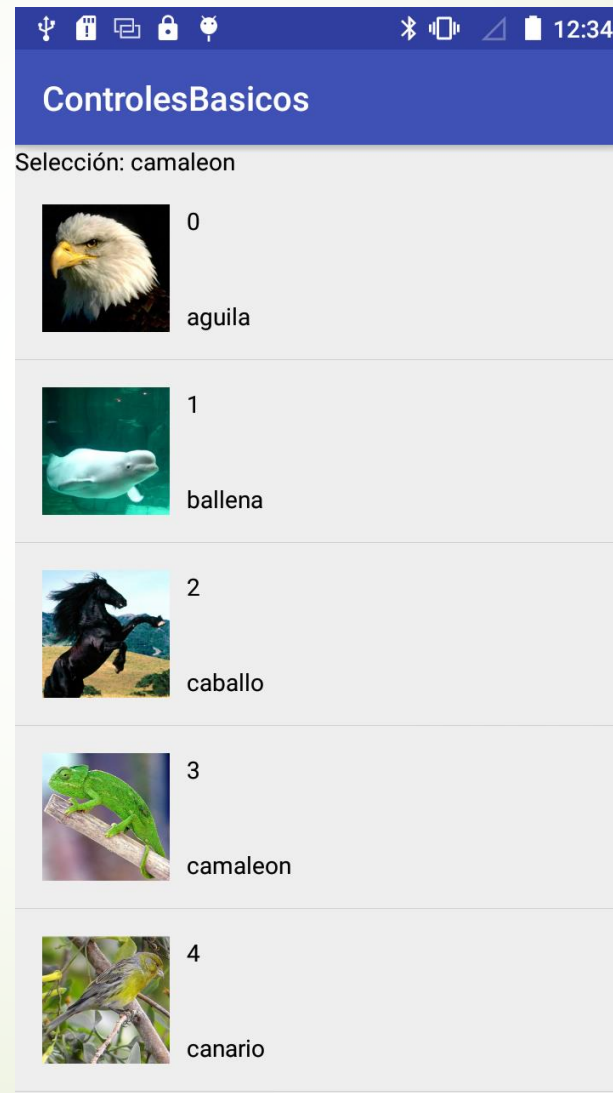
```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {  
        String selection = ((Animal)adapterView.getAdapter().getItem(position)).getNombre();  
  
        //String selection = ((TextView)view.findViewById(R.id.tvContent)).getText().toString();  
  
        textSelection.setText("Selección: "+selection);  
    }  
});
```



ListView Personalizado

- El evento onItemClick recibe 4 parámetros:
 - Referencia al control lista que a recibido el click (AdapterView).
 - Referencia al objeto View correspondiente al ítem pulsado de la lista (View).
 - Posición del elemento pulsado dentro del adaptador de la lista (int).
 - Id del elemento pulsado (int).

ListView Personalizado





GridView

- Representa datos en forma de tablas, es decir, filas y columnas. El evento característico es el `onItemSelected`. Al igual que en los demás controles de selección, se hará automáticamente scroll en caso necesario. Tiene unas propiedades específicas como son:
 - **android:numColumns:** Define el numero de columnas que tendrá el control, también puede tomar el valor de “auto_fit” para que se adapte automáticamente.
 - **android:stretchMode:** Establece qué hacer con el espacio horizontal sobrante.
 - **spacingWidth:** El espacio horizontal sobrante se reparte entre los espacios de las celdas
 - **columnWidth:** El espacio se repare entre las columnas.
 - **android:columnWidth:** Ancho de las columnas.
 - **android:horizontalSpacing:** Representa el espacio horizontal entre las celdas.
 - **android:verticalSpacing:** Representa el espacio vertical entre las celdas.

GridView

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_grid_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    tools:context="com.android.curso.controlesbasicos.GridViewActivity">

    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:numColumns="auto_fit"
        android:horizontalSpacing="5dp"
        android:verticalSpacing="5dp"></GridView>

</RelativeLayout>
```




GridView

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_grid_view);

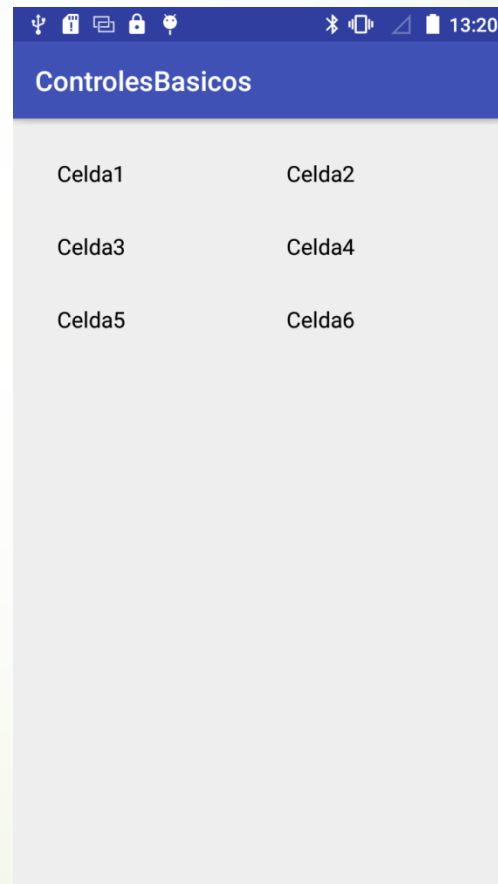
    String [] data = new String[]{"Celda1", "Celda2", "Celda3", "Celda4", "Celda5", "Celda6"};

    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, data);

    GridView gridTest = (GridView) findViewById(R.id.gridView);

    gridTest.setAdapter(adapter);
}
```


GridView





➤ <https://github.com/rononunsys/ControlesBasicos.git>