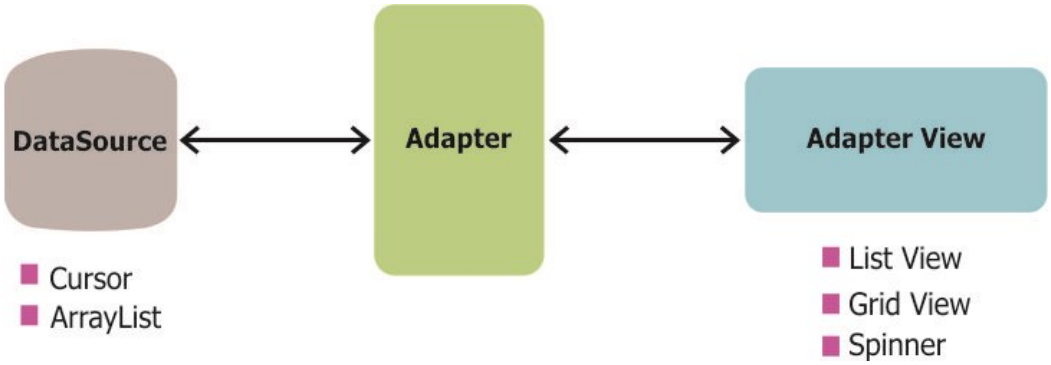
Interfaz de usuario en Android: Controles de selección (I)

Una vez repasados los controles básicos (I, II, III) que podemos utilizar en nuestras aplicaciones Android, vamos a dedicar los próximos artículos a describir los diferentes controles de selección disponibles en la plataforma. Al igual que en otros *frameworks* Android dispone de diversos controles que nos permiten seleccionar una opción dentro de una lista de posibilidades. Así, podremos utilizar por ejemplo listas desplegables (`Spinner`), listas fijas (`ListView`), o tablas (`GridView`).



En este primer artículo dedicado a los controles de selección vamos a describir un elemento importante y común a todos ellos, los ***adaptadores***, y lo vamos a aplicar al primer control de los indicados, las listas desplegables.

Adaptadores en Android (*adapters*)

Para los desarrolladores de java que hayan utilizado frameworks de interfaz gráfica como *Swing*, el concepto de *adaptador* les resultará familiar. Un adaptador representa algo así como una interfaz común al modelo de datos que existe por detrás de todos los controles de selección que hemos comentado. Dicho de otra forma, **todos los controles de selección accederán a los datos que contienen a través de un adaptador**.

Además de proveer de datos a los controles visuales, **el adaptador también será responsable de generar a partir de estos datos las vistas específicas que se mostrarán dentro del control de selección**. Por ejemplo, si cada elemento de una lista estuviera formado a su vez por una imagen y varias etiquetas, el responsable de generar y establecer el contenido de todos estos "*sub-elementos*" a partir de los datos será el propio adaptador.

Android proporciona de serie varios tipos de adaptadores sencillos, aunque podemos extender su funcionalidad fácilmente para adaptarlos a nuestras necesidades. Los más comunes son los siguientes:

- **`ArrayAdapter`**. Es el más sencillo de todos los adaptadores, y provee de datos a un control de selección a partir de un array de objetos de cualquier tipo.
- **`SimpleAdapter`**. Se utiliza para mapear datos sobre los diferentes controles definidos en un fichero XML de layout.
- **`SimpleCursorAdapter`**. Se utiliza para mapear las columnas de un cursor abierto sobre una base de datos sobre los diferentes elementos visuales contenidos en el control de selección.

Para no complicar excesivamente los tutoriales, por ahora nos vamos a conformar con describir la forma de utilizar un `ArrayAdapter` con los diferentes controles de selección disponibles.

Veamos cómo crear un adaptador de tipo `ArrayAdapter` para trabajar con un array genérico de java:

```
1
2
3
4
5
6
```

```
final String[] datos =
        new String[]{"Elem1","Elem2","Elem3","Elem4","Elem5"};

ArrayAdapter<String> adaptador =
        new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, datos);
```

Comentemos un poco el código. Sobre la primera línea no hay nada que decir, es tan sólo la definición del array java que contendrá los datos a mostrar en el control, en este caso un array sencillo con cinco cadenas de caracteres. En la segunda línea creamos el adaptador en sí, al que pasamos 3 parámetros:

1. El ***contexto***, que normalmente será simplemente una referencia a la actividad donde se crea el adaptador.
2. **El ID del *layout* sobre el que se mostrará cada uno de los datos del control** (cada elemento del array). En este caso le pasamos el ID de un layout predefinido en Android (`android.R.layout.simple_spinner_item`), formado únicamente por un control `TextView`, pero podríamos pasarle el ID de cualquier layout personalizado de nuestro proyecto con cualquier estructura y conjunto de controles, más adelante veremos cómo (en el apartado dedicado a las listas fijas).
3. El ***array*** que contiene los **datos a mostrar**.

Con esto ya tendríamos creado nuestro adaptador para los datos a mostrar y ya tan sólo nos quedaría asignar este adaptador a nuestro control de selección para que éste mostrase los datos en la aplicación.

Una **alternativa** a tener en cuenta si los datos a mostrar en el control son estáticos sería **definir** la lista de posibles valores como un recurso de tipo **`string-array`**. Para ello, primero crearíamos un nuevo fichero XML en la carpeta */res/values* llamado por ejemplo *valores_array.xml* e incluiríamos en él los valores seleccionables de la siguiente forma:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="valores_array">
        <item>Elem1</item>
        <item>Elem2</item>
        <item>Elem3</item>
        <item>Elem4</item>
        <item>Elem5</item>
    </string-array>
</resources>
```

```
9
10
```

Tras esto, a la hora de crear el adaptador, utilizaríamos el método **`createFromResource()`** para hacer referencia a este array XML que acabamos de crear:

```
1     ArrayAdapter<CharSequence> adapter =
2         ArrayAdapter.createFromResource(this,
3             R.array.valores_array,
4             android.R.layout.simple_spinner_item);
```

Control Spinner [API]

Las listas desplegables en Android se llaman Spinner. Funcionan de forma similar a cualquier control de este tipo, el usuario selecciona la lista, se muestra una especie de lista emergente al usuario con todas las opciones disponibles y al seleccionarse una de ellas ésta queda fijada en el control. Para añadir una lista de este tipo a nuestra aplicación podemos utilizar el código siguiente:

```
1     <Spinner android:id="@+id/CmbOpciones"
2         android:layout_width="match_parent"
3         android:layout_height="wrap_content" />
```

Poco vamos a comentar de aquí ya que lo que nos interesan realmente son los datos a mostrar. En cualquier caso, las opciones para personalizar el aspecto visual del control (fondo, color y tamaño de fuente, …) son las mismas ya comentadas para los controles básicos.

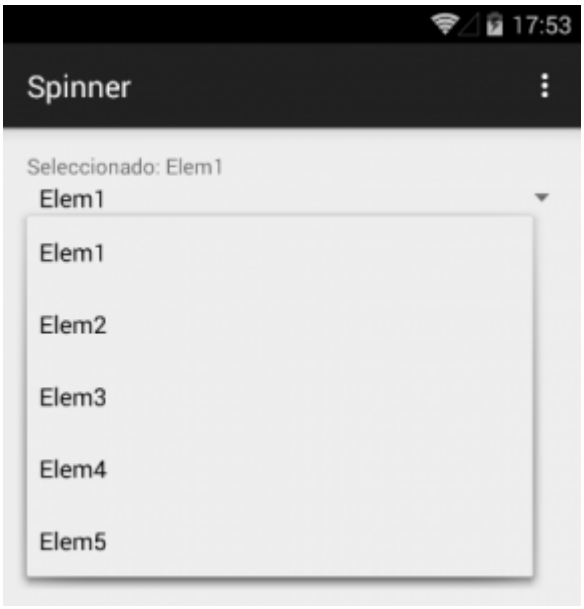
Para enlazar nuestro adaptador (y por tanto nuestros datos) a este control utilizaremos el siguiente código java:

```
1     private Spinner cmbOpciones;
2
3     //...
4
5     cmbOpciones = (Spinner)findViewById(R.id.CmbOpciones);
6
7     adaptador.setDropDownViewResource(
8             android.R.layout.simple_spinner_dropdown_item);
```

```
9       cmbOpciones.setAdapter(adaptador);
10
```

Comenzamos como siempre por obtener una referencia al control a través de su ID. Y en la última línea asignamos el adaptador al control mediante el método `setAdapter()`. ¿Y la segunda línea para qué es? Cuando indicamos en el apartado anterior cómo construir un adaptador vimos cómo uno de los parámetros que le pasábamos era el ID del layout que utilizaríamos para visualizar los elementos del control. Sin embargo, en el caso del control `Spinner`, este layout tan sólo se aplicará al elemento seleccionado en la lista, es decir, al que se muestra directamente sobre el propio control cuando no está desplegado. Sin embargo, antes indicamos que el funcionamiento normal del control `Spinner` incluye entre otras cosas mostrar una lista emergente con todas las opciones disponibles. Pues bien, **para personalizar también el aspecto de cada elemento en dicha lista emergente** tenemos el método **`setDropDownViewResource(`*`ID_layout`*`)`**, al que podemos pasar otro ID de layout distinto al primero sobre el que se mostrarán los elementos de la lista emergente. En este caso hemos utilizado otro layout predefinido Android para las listas desplegables (`android.R.layout.simple_spinner_dropdown_item`), formado por una etiqueta de texto con la descripción de la opción (en Android 2.x también se muestra un marcador circular a la derecha que indica si la opción está o no seleccionada).

Con estas simples lineas de código conseguiremos mostrar un control como el que vemos en la siguiente imagen:



En cuanto a los eventos lanzados por el control `Spinner`, el más comúnmente utilizado será el generado al seleccionarse una opción de la lista desplegable, **`onItemSelected`**. Para capturar este evento se procederá de forma similar a lo ya visto para otros controles anteriormente, asignándole su controlador mediante el método `setOnItemSelectedListener()`:

```
1
2       cmbOpciones.setOnItemSelectedListener(
3               new AdapterView.OnItemSelectedListener() {
4                   public void onItemSelected(AdapterView<?> parent,
5                                       android.view.View v, int position, long id) {
6                       lblMensaje.setText("Seleccionado: " +
7                               parent.getItemAtPosition(position));
8                   }
9
10                  public void onNothingSelected(AdapterView<?> parent) {
11                      lblMensaje.setText("");
12                  }
              });
```

A diferencia de ocasiones anteriores, para este evento definimos dos métodos, el primero de ellos (`onItemSelected`) que será llamado cada vez que se selecciones una opción en la lista desplegable, y el segundo (`onNothingSelected`) que se llamará cuando no haya ninguna opción seleccionada (esto puede ocurrir por ejemplo si el adaptador no tiene datos). Además, vemos cómo para recuperar el dato seleccionado utilizamos

el método `getItemAtPosition()` del parámetro `AdapterView` que recibimos en el evento.

Puedes consultar y/o descargar el código completo de los ejemplos desarrollados en este artículo accediendo a la página del curso en GitHub.

En el siguiente artículo describiremos el uso de controles de tipo lista (`ListView`).