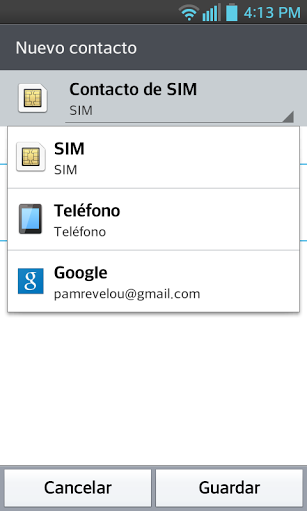
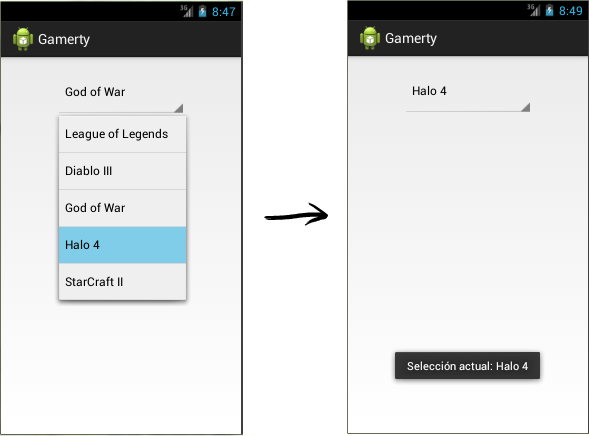
|  |  |  |
| --- | --- | --- |
| n android it is called a Spinner you can take a look at the tutorial here.  [Hello, Spinner](http://developer.android.com/guide/tutorials/views/hello-spinner.html)  And this is a very vague question, you should try to be more descriptive of your problem.   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/3024695)[improve this answer](http://stackoverflow.com/posts/3024695/edit) | answered Jun 11 '10 at 16:59  [gruntled](http://stackoverflow.com/users/360584/gruntled) **1,875**1114 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 12 |  | | -1: The spinner is NOT a combo box. I suggest you read the wikipedia article about it:[en.wikipedia.org/wiki/Combo\_box](http://en.wikipedia.org/wiki/Combo_box) – [vbence](http://stackoverflow.com/users/638471/vbence) [May 19 '11 at 10:36](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment7010595_3024695) | | |  |  | | --- | --- | | 12 |  | | I suggest you consider this in the context of android development. [designerandroid.com/?p=8](http://www.designerandroid.com/?p=8). In the Context of android dev it is referred to as a Spinner. Please do your research next time. – [gruntled](http://stackoverflow.com/users/360584/gruntled) [May 19 '11 at 16:56](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment7017789_3024695) | | |  |  | | --- | --- | | 3 |  | | Yes and by looking at the site you have provided yourself you can see that they do make mention to a ComboBox on that page but in the API there is only a reference to Spinner ([developer.android.com/resources/tutorials/views/…](http://developer.android.com/resources/tutorials/views/hello-spinner.html)) Here they clearly state that a "Spinner is a widget similar to a drop-down list for selecting items". I agree with you that this SHOULD be called a ComboBox as with other Java implementations but in this context it is not. – [gruntled](http://stackoverflow.com/users/360584/gruntled) [May 19 '11 at 20:12](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment7020973_3024695) | | |  |  | | --- | --- | | 5 |  | | Stop nitpicking and admit that everyone had problems finding a control that kind of acts like a combo box or list box the first time they wrote an Android app... – [Torp](http://stackoverflow.com/users/316748/torp) [Jul 26 '11 at 11:37](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment8114938_3024695) | | |  |  | | --- | --- | | 6 |  | | I am still looking for a combo box.. Ive seen Spinners. Used spinners. But frankly I have a scenario where I need to set the text to something other than a given option. Aka they can type in it. Spinners are not combo boxes, but they are usually a valid alternative. – [Doomsknight](http://stackoverflow.com/users/940834/doomsknight) [Jun 29 '12 at 15:01](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment14808280_3024695) |   [show **2** more comments](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote8down vote | **Here is an example of custom combobox in android:**  package myWidgets;  import android.content.Context;  import android.database.Cursor;  import android.text.InputType;  import android.util.AttributeSet;  import android.view.View;  import android.widget.AutoCompleteTextView;  import android.widget.ImageButton;  import android.widget.LinearLayout;  import android.widget.SimpleCursorAdapter;  public class ComboBox extends LinearLayout {  private AutoCompleteTextView \_text;  private ImageButton \_button;  public ComboBox(Context context) {  super(context);  this.createChildControls(context);  }  public ComboBox(Context context, AttributeSet attrs) {  super(context, attrs);  this.createChildControls(context);  }  private void createChildControls(Context context) {  this.setOrientation(HORIZONTAL);  this.setLayoutParams(new LayoutParams(LayoutParams.FILL\_PARENT,  LayoutParams.WRAP\_CONTENT));  \_text = new AutoCompleteTextView(context);  \_text.setSingleLine();  \_text.setInputType(InputType.TYPE\_CLASS\_TEXT  | InputType.TYPE\_TEXT\_VARIATION\_NORMAL  | InputType.TYPE\_TEXT\_FLAG\_CAP\_SENTENCES  | InputType.TYPE\_TEXT\_FLAG\_AUTO\_COMPLETE  | InputType.TYPE\_TEXT\_FLAG\_AUTO\_CORRECT);  \_text.setRawInputType(InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);  this.addView(\_text, new LayoutParams(LayoutParams.WRAP\_CONTENT,  LayoutParams.WRAP\_CONTENT, 1));  \_button = new ImageButton(context);  \_button.setImageResource(android.R.drawable.arrow\_down\_float);  \_button.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View v) {  \_text.showDropDown();  }  });  this.addView(\_button, new LayoutParams(LayoutParams.WRAP\_CONTENT,  LayoutParams.WRAP\_CONTENT));  }  /\*\*  \* Sets the source for DDLB suggestions.  \* Cursor MUST be managed by supplier!!  \* @param source Source of suggestions.  \* @param column Which column from source to show.  \*/  public void setSuggestionSource(Cursor source, String column) {  String[] from = new String[] { column };  int[] to = new int[] { android.R.id.text1 };  SimpleCursorAdapter cursorAdapter = new SimpleCursorAdapter(this.getContext(),  android.R.layout.simple\_dropdown\_item\_1line, source, from, to);  // this is to ensure that when suggestion is selected  // it provides the value to the textbox  cursorAdapter.setStringConversionColumn(source.getColumnIndex(column));  \_text.setAdapter(cursorAdapter);  }  /\*\*  \* Gets the text in the combo box.  \*  \* @return Text.  \*/  public String getText() {  return \_text.getText().toString();  }  /\*\*  \* Sets the text in combo box.  \*/  public void setText(String text) {  \_text.setText(text);  }  }  Hope it helps!!   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/11094410)[improve this answer](http://stackoverflow.com/posts/11094410/edit) | [edited Mar 11 '13 at 6:42](http://stackoverflow.com/posts/11094410/revisions) | answered Jun 19 '12 at 4:33  [Raneez Ahmed](http://stackoverflow.com/users/1219456/raneez-ahmed) **1,923**11642 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Thank you for your reply. I want to use this widget but I want to use an array of String as data source not a cursor. what should I do? – [Ali Behzadian Nejad](http://stackoverflow.com/users/1152549/ali-behzadian-nejad) [Sep 14 '13 at 15:13](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment27729706_11094410) |   add a comment |
| up vote5down vote | Not tested, but the closer you can get seems to be is with [AutoCompleteTextView](http://developer.android.com/reference/android/widget/AutoCompleteTextView.html). You can write an adapter wich ignores the filter functions. Something like:  class UnconditionalArrayAdapter<T> extends ArrayAdapter<T> {  public UnconditionalArrayAdapter(Context context, int textViewResourceId, T[] objects) {  super(context, textViewResourceId, objects);  }  public Filter getFilter() {  return new NullFilter();  }  class NullFilter extends Filter {  protected Filter.FilterResults performFiltering(CharSequence constraint) {  return getFilter().performFiltering("");  }  protected void publishResults(CharSequence constraint, Filter.FilterResults results) {  return getFilter().publishResults(constraint, results);  }  }  }  ... then in your onCreate:  String[] COUNTRIES = new String[] {"Belgium", "France", "Italy", "Germany"};  ArrayAdapter<String> adapter = new UnconditionalArrayAdapter<String>(this,  android.R.layout.simple\_dropdown\_item\_1line, COUNTRIES);  AutoCompleteTextView textView = (AutoCompleteTextView)  findViewById(R.id.countries\_list);  textView.setAdapter(adapter);  The code is not tested, there can be some features with the filtering method I did not consider, but there you have it, the basic principles to emulate a ComboBox with an AutoCompleteTextView.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/6057971)[improve this answer](http://stackoverflow.com/posts/6057971/edit) | [edited May 19 '11 at 12:09](http://stackoverflow.com/posts/6057971/revisions) | answered May 19 '11 at 11:32  [vbence](http://stackoverflow.com/users/638471/vbence) **12.6k**53983 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | This code doesn't come close to compiling. The calls to getFilter() look like attempts at an infinite loop, and publishResults is returning a value from a void method. The idea's good overall, but someone should fix this example. – [dhakim](http://stackoverflow.com/users/1151342/dhakim) [Mar 21 '14 at 19:59](http://stackoverflow.com/questions/3024656/how-can-i-show-a-combobox-in-android#comment34352160_6057971) |   add a comment |

|  |  |  |  |
| --- | --- | --- | --- |
| up vote1down vote | The questions is perfectly valid and clear since Spinner and ComboBox (read it: Spinner where you can provide a custom value as well) are two different things.  I was looking for the same thing myself and I wasn't satisfied with the given answers. So I created my own thing. Perhaps some will find the following hints useful. I am not providing the full source code as I am using some legacy calls in my own project. It should be pretty clear anyway.  Here is the screenshot of the final thing:  The first thing was to create a view that will look the same as the spinner that hasn't been expanded yet. In the screenshot, on the top of the screen (out of focus) you can see the spinner and the custom view right bellow it. For that purpose I used LinearLayout (actually, I inherited from Linear Layout) with style="?android:attr/spinnerStyle". LinearLayout contains TextView with style="?android:attr/spinnerItemStyle". Complete XML snippet would be:  <com.example.comboboxtest.ComboBox  style="?android:attr/spinnerStyle"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  >  <TextView  android:id="@+id/textView"  style="?android:attr/spinnerItemStyle"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ellipsize="marquee"  android:singleLine="true"  android:text="January"  android:textAlignment="inherit"  />  </com.example.comboboxtest.ComboBox>  As, I mentioned earlier ComboBox inherits from LinearLayout. It also implements OnClickListener which creates a dialog with a custom view inflated from the XML file. Here is the inflated view:  <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  >  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="horizontal"  >  <EditText  android:id="@+id/editText"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:ems="10"  android:hint="Enter custom value ..." >  <requestFocus />  </EditText>  <Button  android:id="@+id/button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:text="OK"  />  </LinearLayout>  <ListView  android:id="@+id/listView1"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  />  </LinearLayout>  There are two more listeners that you need to implement: onItemClick for the list and onClick for the button. Both of these set the selected value and dismiss the dialog.  For the list, you want it to look the same as expanded Spinner, you can do that providing the list adapter with the appropriate (Spinner) style like this:  ArrayAdapter<String> adapter =  new ArrayAdapter<String>(  activity,  android.R.layout.simple\_spinner\_dropdown\_item,  states  );  More or less, that should be it.   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/28441563)[improve this answer](http://stackoverflow.com/posts/28441563/edit) | answered Feb 10 at 20:54  [Viktor Brešan](http://stackoverflow.com/users/96313/viktor-bre%c5%a1an) **2,766**21826 | |
|  | add a comment |
| up vote0down vote | For a combobox (<http://en.wikipedia.org/wiki/Combo_box>) which allows free text input and has a dropdown listbox I used a AutoCompleteTextView as suggested by vbence.  I used the onClickListener to display the dropdown list box when the user selects the control.  I believe this resembles this kind of a combobox best.  private static final String[] STUFF = new String[] { "Thing 1", "Thing 2" };  public void onCreate(Bundle b) {  final AutoCompleteTextView view =  (AutoCompleteTextView) findViewById(R.id.myAutoCompleteTextView);  view.setOnClickListener(new View.OnClickListener()  {  @Override  public void onClick(View v)  {  view.showDropDown();  }  });  final ArrayAdapter<String> adapter = new ArrayAdapter<String>(  this,  android.R.layout.simple\_dropdown\_item\_1line,  STUFF  );  view.setAdapter(adapter);  }   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/25505337)[improve this answer](http://stackoverflow.com/posts/25505337/edit) | answered Aug 26 '14 at 12:00  [rve](http://stackoverflow.com/users/79455/rve) **2,677**11336 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote0down vote | Custom made :) you can use dropdown hori/vertical offset properties to position the list currently, also try android:spinnerMode="dialog" it is cooler.  Layout  <LinearLayout  android:layout\_marginBottom="20dp"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="horizontal">  <AutoCompleteTextView  android:layout\_weight="1"  android:id="@+id/edit\_ip"  android:text="default value"  android:layout\_width="0dp"  android:layout\_height= "wrap\_content"/>  <Spinner  android:layout\_marginRight="20dp"  android:layout\_width="30dp"  android:layout\_height="50dp"  android:id="@+id/spinner\_ip"  android:spinnerMode="dropdown"  android:entries="@array/myarray"/>  </LinearLayout>  Java  //set auto complete  final AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.edit\_ip);  ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_dropdown\_item\_1line, getResources().getStringArray(R.array.myarray));  textView.setAdapter(adapter);  //set spinner  final Spinner spinner = (Spinner) findViewById(R.id.spinner\_ip);  spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  @Override  public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  textView.setText(spinner.getSelectedItem().toString());  textView.dismissDropDown();  }  @Override  public void onNothingSelected(AdapterView<?> parent) {  textView.setText(spinner.getSelectedItem().toString());  textView.dismissDropDown();  }  });  res/values/string  <string-array name="myarray">  <item>value1</item>  <item>value2</item>  </string-array>  Was that useful??   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/33376089)[improve this answer](http://stackoverflow.com/posts/33376089/edit) | answered Oct 27 at 18:43  [ShAkKiR](http://stackoverflow.com/users/4440561/shakkir) **11**2 | | |
|  | add a comment | |
| up vote-1down vote | | Absolutely not, LinearLayouts and ArrayAdapters have absolutely nothing to do with combo boxes of any sort.  Just to clarify, the following are (roughly) the same:  A Combo Box (JComboBox) in Java, and a Spinner in Android.  A Spinner (JSpinner) in Java, and some sort of a *picker* in Android, like (NumberPicker).  Note that a spinner in Android is not the same as it is in Java. |

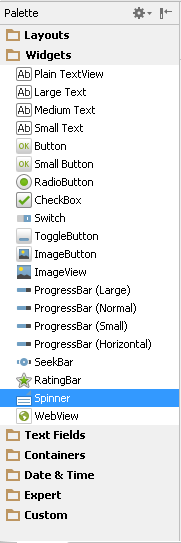
**¿Qué es un Spinner?**

Un Spinner es un View que despliega una lista de elementos que están expectantes a la selección del usuario. Dependiendo del elemento seleccionado nuestra aplicación tomará las decisiones pertinentes para seguir el flujo de ejecución.  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/android-spinner-contactos.png)  
El concepto es sencillo y he preparado un pequeño ejemplo para que puedas interiorizarlo. Se trata de una pequeña aplicación llamada *Gamerty*, que permite seleccionar la guía del videojuego que elija el usuario del Spinner. Veamos como se ve:  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/android-app-spinner.png)  
Puedes descargar el proyecto del siguiente enlace:

Ahora procedamos a recrear su elaboración:

**Ubicar un Spinner en Android Studio**

El primero paso es la creación de un nuevo proyecto en Android Studio con una actividad principal, a la cual le llamarás Main. Inmediatamente vas a su archivo de diseño (activity\_main.xml) y añades desde la ventana *Pallete* el Spinner.

Le asignarás el identificador “GameSpinner”, lo centras horizontalmente (centerHorizontal) y luego lo haces limitar con el borde superior del padre (alignParentTop).  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/android-studio-spinner.png)  
Si todo te ha salido excelente, tendrías el siguiente archivo de diseño:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:paddingBottom="@dimen/activity\_vertical\_margin"

tools:context=".Main">

<Spinner

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/GameSpinner"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true" />

</RelativeLayout>

**Crear un Origen de Datos**

Al igual que cuando vimos [listas en Android](http://www.hermosaprogramacion.com/2014/10/android-listas-adaptadores/), un Spinner también necesita algún origen de datos para definir sus elementos. Esto implica el uso de un adaptador que nos permita crear la comunicación.

Así que vas a crear una lista de strings que representen las opciones que desplegará nuestro Spinner. Como bien sabes, la aplicación Gamerty mostrará al usuario un menu con nombres de videojuegos a consultar. Supón que los videojuegos disponibles son: *League of Legends*, *Diablo III*, *God of War*,*Halo 4* y *StarCraft II*.

Podríamos crear una lista de Strings(ArrayList<String>) para almacenar en nuestra actividad estas cadenas, pero te quedaría mas cómodo y flexible aislarlas en tu archivo de recursos strings.xml. Esta preferencia te permitirá mayor flexibilidad en tu código.

Recuerda que la clase R.java almacena referencias constantes de nuestros recursos, así que no debes preocuparte por la alteración indebida de tus elementos <string> si vas a almacenar las opciones del Spinner.

Si te ha quedado claro, entonces comprenderás la siguiente estructura del strings.xml:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<!-- Strings generales -->

<string name="app\_name">Gamerty</string>

<!-- Conjunto de Strings para el View GameSpinner -->

<string-array name="Games">

<item>League of Legends</item>

<item>Diablo III</item>

<item>God of War</item>

<item>Halo 4</item>

<item>StarCraft II</item>

</string-array>

</resources>

Si te fijas en la definición, verás un elemento <string-array> con nombre “Games”. Si aún no lo sabes, este elemento representa un array de strings (como su nombre lo indica). Dentro de él debes incluir cada dato como un nodo <item>, cuyo contenido es la cadena.

**Poblar un Spinner con un Adaptador**

Una vez hayas definido tu origen de datos vas a crear un adaptador que se ajuste a tus necesidades. En el caso de Gamerty se usa un ArrayAdapter con elementos simples, ya que solo necesitas proyectar el nombre del juego.

Para ello se obtiene la instancia del Spinner, luego declaras el adaptador, lo inicializas en el método onCreate() de la actividad Main y finalmente los relacionas:

//ArrayAdapter para conectar el Spinner a nuestros recursos strings.xml

protected ArrayAdapter<CharSequence> adapter;

...

//Obtener instancia del GameSpinner

Spinner spinner = (Spinner) findViewById(R.id.GameSpinner);

//Asignas el origen de datos desde los recursos

adapter = ArrayAdapter.createFromResource(this, R.array.Games,

android.R.layout.simple\_spinner\_item);

//Asignas el layout a inflar para cada elemento

//al momento de desplegar la lista

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

//Seteas el adaptador

spinner.setAdapter(adapter);

El código anterior muestra el uso del método de clase createFromResource(), el cual nos permite crear un nuevo ArrayAdapter a partir de los recursos string que están en el proyecto.

El primer parámetro que recibe es el contexto, en este caso es la actividad actual, luego la referencia del <array-string> que declaraste y por último el layout para inflar cada item, donde se usa el recurso del sistema simple\_spinner\_item.xml, que representa un TextView sencillo.

Luego usamos el método del adaptador setDropDownViewResource(). Su objetivo es setear un layout para cada item del Spinner, para cuando se despliegue la lista.

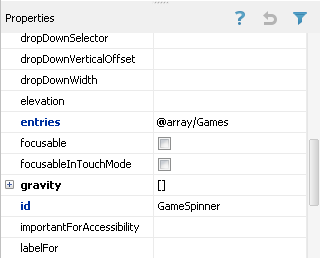
Tomaremos el recurso de la plataforma simple\_spinner\_dropdown\_item.xml, el cual representa un text view seleccionable o CheckedTextView.

Este tipo de text view hereda los comportamientos de uno normal, solo que tiene algunas características adicionales en su forma de iluminarse.

Si deseas establecer la posición inicial del Spinner usa el método setSelection() con un indice entero como parámetro. Si pones atención, en Gamerty creamos una constante llamadaDEFAULT\_POSITION a la cual se le asigno el entero 3 que corresponde a "Halo 4".

**Poblar un Spinner con el Atributo entries**

Debido a que los elementos de nuestro Spinner son estáticos, podemos utilizar un atributo muy útil del elemento <Spinner> llamado android:entries. En él puedes asignar el arreglo de strings que se ha predefinido en los recursos y automáticamente se inflarán implícitamente los elementos el Spinner sin necesidad de un adaptador.

Solo basta con que te dirijas a la pestaña de diseño y modifiques el atributo con el siguiente valor “@array/Games”:  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/android-spinner-entries.png)  
O ir a la definición XML y realizar la respectiva asignación:

<Spinner

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/GameSpinner"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"

<strong>android:entries="@array/Games" </strong>/>

**Manejo de eventos de un Spinner**

En esta sección verás como definir el flujo de tu aplicación de acuerdo a la opción del Spinner que sea seleccionada por el usuario. Para cumplir ese objetivo se empleará la escucha OnSelectedItemListener.

El primer paso que debes realizar es implementar sobre tu actividad Main la escucha. Luego en el método onCreate() le asignas la referencia de la escucha a tu Spinner:

public class Main extends Activity implements OnItemSelectedListener {

...

@Override

protected void onCreate(Bundle savedInstanceState) {

...

//Asignado la escucha

spinner.setOnItemSelectedListener(this);

...

}

...

}

A continuación se deben implementar los métodos onItemSelected() y onNothingSelected() relacionados con OnItemSelectedListener. El primero es llamado cuando has seleccionado un elemento del Spinner y el segundo cuando no tiene items seleccionados.

¿Cual es la idea?, vas a sobrescribir onItemSelected() y añadirás las instrucciones necesarias para que visualice un Toast. Este debe mostrar el nombre del elemento que se ha seleccionado.

Veamos:

@Override

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

//Salvar la posición y valor del item actual

this.position = position;

selection = parent.getItemAtPosition(position).toString();

//Mostramos la selección actual del Spinner

Toast.makeText(this,"Selección actual: "+selection,Toast.LENGTH\_SHORT).show();

}

La primera tarea que se ha realizado es guardar la posición y el valor del texto en dos variables privadas llamadas position y selection (Si revisas el proyecto, han sido declaradas en el inicio de Main).

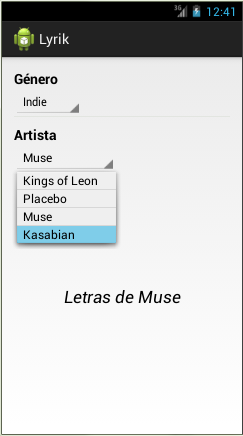
El parámetro position del método onItemSelected() se refiere al indice del elemento seleccionado, por lo que te queda fácil obtener una instancia con getItemAtPosition(). Luego puedes consultar el valor de su atributo text con toString().

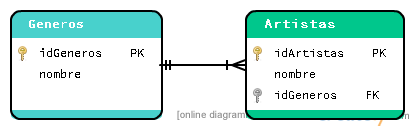
Finalmente creas un Toast prefabricado y muestras el contenido de la selección.

**Poblar un Spinner desde una Base de Datos SQLite**

En esta sección aprenderás a crear los items de un Spinner desde un cursor que contenga el resultado de una consulta en SQLite ([Aprender mas sobre Android y SQLite](http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/)).

El proceso es el mismo que con las listas. Como ya sabes, solo debes usar la clase SimpleCursorAdapter y todo estará resuelto.

Suena fácil pero es mejor aclararlo con un ejemplo. La siguiente aplicación que estudiarás a continuación se llama Lyrik. Esta aplicación muestra las letras de las canciones de un artista disponible, clasificados por género musical.  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/android-dos-spinners.png)  
Puedes descargar el proyecto desde aquí:

Su diseño se basa en una actividad principal (Main) con dos Spinners(GenreSpinner y ArtistSpinner). El primero para desplegar los géneros disponibles y el segundo para proyectar los artistas que pertenecen a ese género. Por lo que el Spinner de los artistas depende del género, representando una cardinalidad uno a muchos (*1:N*) en la base de datos.  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/generos-artistas-mer.png)  
Teniendo en cuenta lo anterior, la base de datos tendría dos tablas: Géneros(Genres) y Artistas(Artists). Donde Artistas recibiría una llave foránea relacionada con la llave principal de Generos.

Una vez definido todas las características de la base de datos se crea un script que automatice su creación en cualquier momento. El nombre de clase es DataBaseScript (Obsérvala en el proyecto).

**Usar la clase SimpleCursorAdapter con un Spinner**

Si detallas la clase Main del proyecto, verás que el primer paso fue inicializar un adaptador del tipo SimpleCursorAdapter de la siguiente forma:

//Creando Adaptador para GenreSpinner

genreSpinnerAdapter = new SimpleCursorAdapter(this,

android.R.layout.simple\_spinner\_item,

dataSource.getAllGenres(),

new String[]{DataBaseScript.GenreColumns.NAME\_GENRE},

new int[]{android.R.id.text1},

SimpleCursorAdapter.FLAG\_REGISTER\_CONTENT\_OBSERVER);

El constructor ha recibido el layout prefabricado simple\_spinner\_item.xml y como fuente de datos se ha establecido el cursor que retorna el método getAllGenres() de la clase LyrikDataSource.

Como su nombre lo indica, este método obtiene todas las filas de la tabla Genres de la base de datos. Su definición no es mas que una sentencia SELECT que selecciona todas las columnas y registros:

public Cursor getAllGenres(){

//Seleccionamos todas las filas de la tabla Genres

return database.rawQuery(

"select \* from " + DataBaseScript.GENRES\_TABLE\_NAME, null);

}

Hasta aquí todo muy bien. Ahora veamos la estrategia para poblar el Spinner de artistas.

**Crear una dependencia entre dos Spinners**

*¿Como hacer para que el segundo Spinner sea poblado al elegirse un item del primer Spinner?*

La respuesta está en los eventos de selección. Bien sabemos que la interfaz OnItemSelectedListener es quién implementa el método onItemSelected() para la ejecución de actividades de selección, así que es allí donde debemos aplicar nuestra lógica.

Ahora pregúntate: *¿Como obtengo los nombres de los artistas que están relacionados con un género?*

En esencia, esta es una duda que debemos resolver primero desde el modelo de datos, es decir, en la base de datos. Y la solución es sencilla, simplemente debes consultar aquellos artistas cuyo valor de su campo idGenre sea igual al asignado en tiempo real por el flujo del programa.

*Por ejemplo…*

La siguiente consulta obtiene el id y nombre de los artistas del genero con código 1001:

SELECT \_id,name

FROM Artists

WHERE idGenre = 1001;

Así que generalizaremos esta sentencia con el siguiente método llamado getArtistsByGenre() –*“Obtener artistas por genero”*:

public Cursor getArtistsByGenre(String genreSelection) {

//Argumentos del WHERE

String selectionArgs[] = new String[]{genreSelection};

String query =

"select "+DataBaseScript.ArtistColumns.ID\_ARTIST+","+DataBaseScript.ArtistColumns.NAME\_ARTIST+

" from "+DataBaseScript.ARTISTS\_TABLE\_NAME +

" where "+DataBaseScript.ArtistColumns.ID\_GENRE+

"= ?";

return database.rawQuery(query,selectionArgs);

}

Si detallas bien, usamos el placeholder ‘?’ para reemplazarlo por genreSelection, que es el parámetro de entrada que representa el id del género requerido. Esto facilita tu trabajo para crear el adaptador del Spinner de artistas.

Finalmente implementamos el método onItemSelected(), donde obtienes el id del género seleccionado y luego inicias el adaptador:

@Override

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

//Obteniendo el id del Spinner que recibió el evento

int idSpinner = parent.getId();

switch(idSpinner) {

case R.id.GenreSpinner:

//Obteniendo el id del género seleccionado

Cursor c1 = (Cursor) parent.getItemAtPosition(position);

String genreSelection = c1.getString(

c1.getColumnIndex(DataBaseScript.GenreColumns.ID\_GENRE));

//Poblar el ArtistSpinner

activeArtistSpinner(genreSelection);

break;

...

Recuerda que si el adaptador maneja cursores el resultados obtenido por getItemAtPosition() debe ser casteado a Cursor. Seguidamente puedes obtener el valor de la fila deseado.

Para activar el Spinner de artistas de forma personalizada se creó el método activeArtistSpinner(). Su trabajo es crear el adaptador con el id de entrada a través del método getArtistsByGenre(). Luego lo asignas al Spinner y finalmente relacionas la escucha:

private void activeArtistSpinner(String genreSelection) {

//Creando Adaptador para ArtistSpinner con el id del género

artistSpinnerAdapter = new SimpleCursorAdapter(this,

android.R.layout.simple\_spinner\_item,

dataSource.getArtistsByGenre(genreSelection),

new String[]{DataBaseScript.ArtistColumns.NAME\_ARTIST},

new int[]{android.R.id.text1},

SimpleCursorAdapter.FLAG\_REGISTER\_CONTENT\_OBSERVER);

//Seteando el adaptador creado

artistSpinner.setAdapter(artistSpinnerAdapter);

//Relacionado la escucha

artistSpinner.setOnItemSelectedListener(this);

}

Ahora solo queda agregar al método onItemSelected() las acciones que se realizaran cuando sea seleccionado un elemento del ArtistSpinner:

case R.id.ArtistSpinner:

//Obteniendo el nombre del artista seleccionado

Cursor c2 = (Cursor) parent.getItemAtPosition(position);

String artistSelection = c2.getString(

c2.getColumnIndex(DataBaseScript.ArtistColumns.NAME\_ARTIST));

//Cambiando el texto de LyricList según el Artista seleccionado

lyricList.setText(getResources().getString(R.string.LyricList)+" "+artistSelection);

break;

La idea fue asignarle al TextView un mensaje del tipo “Letras de <Artista>”, donde <Artista> será reemplazado por la selección actual del usuario.

**Añadir un Spinner a la Action Bar**

Si logras recordar un poco sobre [la anatomía de la Action Bar](http://www.hermosaprogramacion.com/2014/09/android-action-bar/#anatomia) tendrás presente que existe una sección llamada *View Control*. Se había dicho que en este espacio era posible inflar Views que interactuan con el contenido actual de la actividad o fragmento. Muy bien, en este apartado veremos como inflar un Spinner justo en ese lugar para implementar un filtro de datos.

Dicho filtro será incluido en una aplicación llamada *Fazhion*. Esta aplicación fue pedida por un cliente imaginario que desea proyectar en una actividad todos los estilos de zapatos que vende su negocio. Quiere que se muestre el *nombre del zapato*, su *precio*, la *descripción* y una *miniatura* de su aspecto. Ademáss desea que el cliente pueda filtrar los productos por precio y nombre.

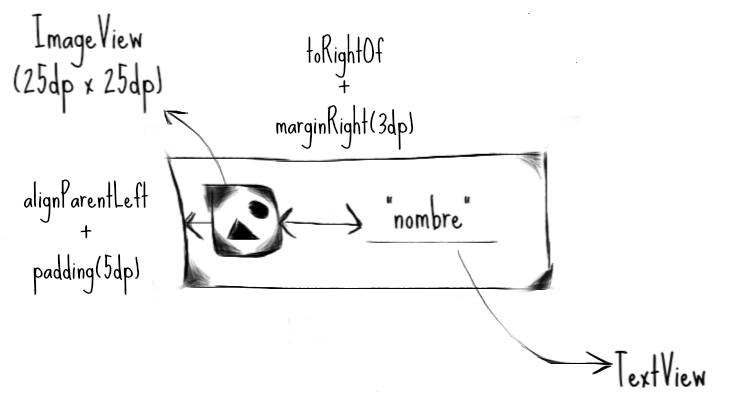
Así que lo que harás es añadir un Spinner a la Action Bar para solucionar este problema. Donde el resultado final será algo como esto:  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/spinner-personalizado-actionbar.png)  
Puedes descargar el código completo presionando el siguiente botón:

La aplicación Fazhion se compone de una ListActivity principal llamada Main. Los elementos de la lista son inflados a través de un CursorAdapter personalizado que accede a la base de datos Fazhion previamente creada.

Ahora veamos como se hizo para añadir el Spinner a la Action Bar:

**Paso 1: Crear el Adaptador del Spinner**

Puedes usar un ArrayAdapter personalizado o un SimpleCursorAdapter como vimos con anterioridad. También puedes personalizar los items al igual que en la captura de la aplicación Fazhion, cuyo Spinner tiene un ImageView en cada item.

Si deseas añadir un ImageView al Spinner debes crear un layout personalizado. Fazhion utiliza el layout spinner\_item.xml, cuyo diseño se proyecta en el siguiente bosquejo:  
[](http://www.hermosaprogramacion.com/wp-content/uploads/2014/10/spinner-item-boceto.png)  
Y su descripción xml es la siguiente:

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="5dp">

<ImageView

android:layout\_width="25dp"

android:layout\_height="25dp"

android:id="@+id/ItemIcon"

android:src="@drawable/ic\_launcher"

android:layout\_alignParentLeft="true"

android:layout\_centerInParent="true"

android:layout\_marginRight="3dp" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceMedium"

android:text="Texto"

android:id="@+id/TextItem"

android:layout\_centerVertical="true"

android:layout\_toRightOf="@+id/ItemIcon" />

</RelativeLayout>

Luego debes crear un adaptador personalizado que soporte el anterior layout. Esto implica crear un origen de datos basado en una clase que represente cada elemento del Spinner.

Si abstraes las características de cada elemento del Spinner, deducirás que poseen un icono que proyecta su significado y un nombre. Así que crea una nueva clase con estas características y llámala SpinnerItem:

public class SpinnerItem {

private String Name;

private int iconId;

public SpinnerItem(String name, int iconId) {

Name = name;

this.iconId = iconId;

}

public String getName() {

return Name;

}

public void setName(String name) {

Name = name;

}

public int getIconId() {

return iconId;

}

public void setIconId(int iconId) {

this.iconId = iconId;

}

}

Por ultimo debes crear un adaptador que infle cada view a través de una lista de tipo SpinnerItem. A este le denominarás FilterAdapter:

public class FilterAdapter extends ArrayAdapter {

public FilterAdapter(Context context, List<SpinnerItem> objects) {

super(context, 0, objects);

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

/\*

Obtener el objeto procesado actualmente

\*/

SpinnerItem currentItem = getItem(position);

/\*

Obtener LayoutInflater de la actividad

\*/

LayoutInflater inflater = (LayoutInflater) parent.getContext().

getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

/\*

Evitar inflar de nuevo un elemento previamente inflado

\*/

if(convertView==null){

convertView = inflater.inflate(R.layout.spinner\_item, parent, false);

}

/\*

Instancias del Texto y el Icono

\*/

TextView name = (TextView)convertView.findViewById(R.id.TextItem);

ImageView icon = (ImageView)convertView.findViewById(R.id.ItemIcon);

/\*

Asignar valores

\*/

name.setText(currentItem.getName());

icon.setImageResource(currentItem.getIconId());

return convertView;

}

@Override

public View getDropDownView(int position, View convertView, ViewGroup parent) {

/\*

Debido a que deseamos usar spinner\_item.xml para inflar los

items del Spinner en ambos casos, entonces llamamos a getView()

\*/

return getView(position,convertView,parent);

}

}

Si eres buen observador verás algo inusual que nunca antes habíamos hecho al definir un adaptador. Hemos sobrescrito un método llamado getDropDownnView().

Este método tiene las mismas funciones de getView(), solo que se ejecuta cuando el Spinner es desplegado. Como ya habías visto, los items del Spinner pueden utilizar otro diseño en ese momento.

Debido a que no deseamos que tomen otra forma, simplemente retornamos en el mismo view que genera el método getView().

Ahora solo crea en tu actividad principal una lista de items para el Spinner y luego asignalos en el constructor del nuevo FilterAdapter:

/\*

Inicializar los items estáticamente

\*/

List items = new ArrayList();

items.add(new SpinnerItem("Mayor Precio",R.drawable.ic\_higher\_prices));

items.add(new SpinnerItem("Menor Precio",R.drawable.ic\_lower\_prices));

items.add(new SpinnerItem("Alfabéticamente",R.drawable.ic\_alphabetical\_order));

//Creando el nuevo adaptador para el Spinner

spinnerAdapter = new FilterAdapter(this,items);

**Paso 2: Implementar la interfaz OnNavigationListener**

Para los eventos usaremos la clases OnNavigationListener. Esta interfaz te permite decidir que sucederá cuando el usuario selecciona uno de los elementos del Spinner en la Action Bar. En el caso de Fazhion se ejecutaron las consultas necesarias en la base de datos para obtener los registros de forma ordenada:

OnNavigationListener navigationListener =new OnNavigationListener() {

/\*

0 = Mayor Precio

1 = Menor Precio

2 = Alfabeticamente

\*/

@Override

public boolean onNavigationItemSelected(int position, long itemId) {

switch(position){

case 0:

/\*

Obtener precios en orden descendente

\*/

fazhionAdapter.changeCursor(dataSource.getShoesOrderPrice(false));

break;

case 1:

/\*

Obtener precios en orden ascendente

\*/

fazhionAdapter.changeCursor(dataSource.getShoesOrderPrice(true));

break;

case 2:

/\*

Obtener nombres en orden alfabético

\*/

fazhionAdapter.changeCursor(dataSource.getShoesOrderAZ(true));

break;

}

return true;

}

};

La escucha OnNavigationLisner te entrega a disposición el método onNavigationItemSelected() para definir las acciones de cada elemento del Spinner. Donde el parámetro entero position será nuestro referente para identificar cada elemento.

Al seleccionar el item en la posición 0, el adaptador de la lista actualiza el cursor a través de un método llamado getShoesOrderPrice(). Este ordena por precio la lista de los zapatos.

Será en orden descendente si recibe false y en orden ascendente si recibe true. En este caso consulta los registros de forma descendente y para la posición 1 se hace en forma ascendente.

El elemento en la posición 2 llama a getShoesOrderAZ(). Este método ordena alfabéticamente los nombres de los zapatos. Si recibe true lo hace de *A-Z*, si es false ordena de *Z-A*.

Revisa la clase FazhionDataSource del proyecto. Allí encontrarás la definición de ambos métodos.

**Paso 3: Habilitar el Modo de Navegación de Lista**

La navegación de lista le indica a la action bar que inflaremos un menú con una lista de items, para gestionar la navegación entre fragmentos o actividades. Pero en el caso de Fazhion no habrá navegación, si no que filtraremos los elementos de la ListActivity.

Si deseas activar este modo y relacionar la escucha haz lo siguiente:

//Obteniendo una instancia de la Action bar

ActionBar actionBar = getActionBar();

if( actionBar != null) {

//Habilitiando el modo de navegación con lista

actionBar.setNavigationMode(ActionBar.NAVIGATION\_MODE\_LIST);

//Seteando la escucha y el spinner relacionado

actionBar.setListNavigationCallbacks(spinnerAdapter, navigationListener);

}

El código anterior muestra como se activa el modo de navegación por lista con el método setNavigationMode(). Este recibe como parámetro una constante llamada NAVIGATION\_MODE\_LIST para que el Spinner sea funcional sobre la Action Bar.

Luego debes asignar la escucha y el adaptador del Spinner con el método setListNavigationCallbacks(). Con ello tendrás habilitado el modo de navegación y la funcionalidad*DropDown* del Spinner.