



Controles de selección

Adapter




















Controles de selección



- Android dispone de diversos controles que nos permiten **seleccionar una opción dentro de una lista** de posibilidades:
 - Spinner: listas desplegadas
 - ListView: listas fijas
 - GridView: tablas
 - ...
- Un elemento importante y común a todos ellos son los adaptadores (**Adapter**): todos los controles de selección accederán a los datos que contienen a través de un adaptador.

Listener para controles de selección

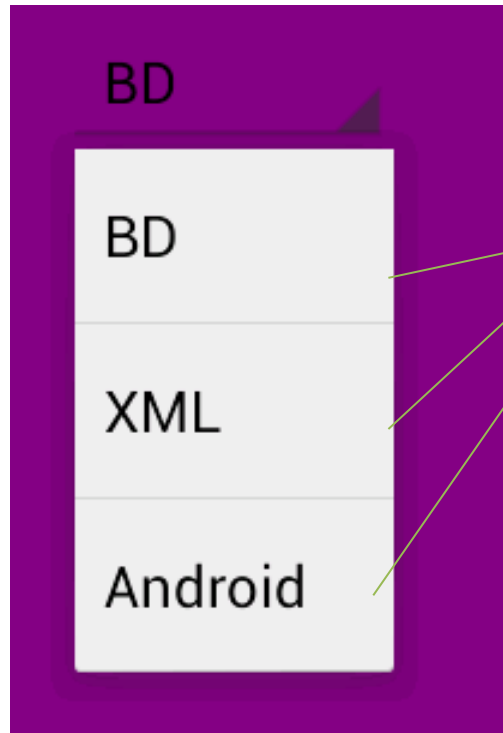


```
m  setOnClickListener (OnClickListener l)
m  setOnItemClickListener (OnItemClickListener l)
m  setOnApplyWindowInsetsListener (OnApplyWindowInsetsListe...
m  setOnContextClickListener (OnContextClickListener l)
m  setOnCreateContextMenuListener (OnCreateContextMenuListe...
m  setOnDragListener (OnDragListener l)
m  setOnFocusChangeListener (OnFocusChangeListener l)
m  setOnGenericMotionListener (OnGenericMotionListener l)
m  setOnHierarchyChangeListener (OnHierarchyChangeListener ...
m  setOnHoverListener (OnHoverListener l)
m  setOnItemLongClickListener (OnItemLongClickListener list...
m  setOnItemSelectedListener (OnItemSelectedListener listen...
m  setOnKeyListener (OnKeyListener l)
m  setOnLongClickListener (OnLongClickListener l)
m  setOnScrollChangeListener (OnScrollChangeListener l)
m  setOnSystemUiVisibilityChangeListener (OnSystemUiVisibil...
m  setOnTouchListener (OnTouchListener l)
```

Concepto de Adapter



- Un adapter (Adaptador) es un objeto que **enlaza los datos a mostrar con las vistas donde se muestran.**

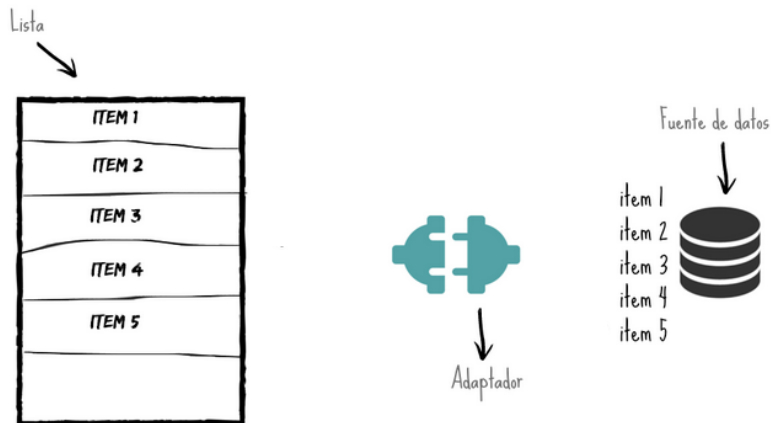


- Es el responsable de crear vistas hijas para cada uno de los ítems de la vista global (spinner, listview,...) y proporcionar acceso al dato que se mostrará.
- Los datos pueden ser tan sencillos como en el ejemplo (cadenas de texto) o tan complejos como necesitemos (imágenes con varias líneas de texto, etc.).

Métodos



- Los Adapter tienen los siguientes métodos para comunicarse con controles de selección:



- **getCount():** Devuelve la cantidad de elementos que tiene un adaptador. Con este valor la lista ya puede establecer un límite para añadir filas.
- **getItemAtPosition(n):** Obtiene el n-ésimo elemento de la fuente de datos asignada al adaptador en una posición establecida. Normalmente la fuente de datos es un array o lista de objetos.

- **getView():** Devuelve la View elaborada e inflada de un elemento en una posición específica

Tipos de Adapter



- Existen diferentes tipos de adaptadores:
 - **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
 - **SimpleCursorAdapter**: enlaza el control con datos devueltos en un cursor (por ejemplo, obtenido de la consulta a una base de datos o a un ContentProvider)
 - **ListAdapter**
 - **SpinnerAdapter**
 - ...
- También se pueden definir adaptadores **propios**.

Clase ArrayAdapter



- El ArrayAdapter es una **clase genérica** que liga un array de objetos con clases que hereden de AdapterView (es decir, un elemento del array con vistas).
- Por ejemplo, ArrayAdapter<String> liga el valor toString de cada objeto de una lista a un control TextView definido en un diseño (layout).
- Se pueden utilizar otros constructores para usar layouts más complejos y se puede extender la clase ArrayAdapter para utilizar alternativas a TextView para cada elemento mediante la redefinición del método getView().

Constructor del ArrayAdapter



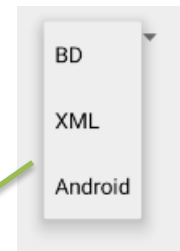
```
ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item, opcion);
```

Parámetro 1: el contexto de nuestra Activity (this)

Parámetro 3: el array con los datos mostrar.

Parámetro 2: indica el tipo de desplegable, pudiendo ser:

- layouts predefinidos de android:
 - android.R.layout.simple_spinner_item
 - android.R.layout.simple_spinner_dropdown_item
- cualquier layout de nuestro proyecto con la estructura y controles deseados



Vista Spinner (lista desplegable)



```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        final TextView textView= findViewById(R.id.textView2) ;
```

```
        final Spinner spinner=findViewById(R.id.spinner);
```

```
        String []opcion={"BD","XML","Android"};
```

```
        ArrayAdapter<String> arrayAdapter= new ArrayAdapter<>(this,android.R.layout.simple_spinner_dropdown_item, opcion);
```

```
        spinner.setAdapter(arrayAdapter);
```

Asignamos el adaptador a la vista Spinner con el método **setAdapter()**

```
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
            @Override
```

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

```
                String[] cursos = { "1º", "1º", "2º" };
```

```
                String resu = "El curso de " + parent.getItemAtPosition(position) + " es " + cursos[position];
```

```
                textView.setText(resu);
```

```
            }
```

```
            @Override
```

```
            public void onNothingSelected(AdapterView<?> parent) {
```

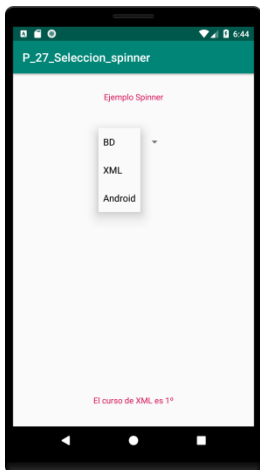
```
                textView.setText("Elige asignatura");
```

```
            }
```

```
        });
```

```
    }
```

```
}
```



OnItemSelectedListener



- Detecta la **selección de una opción** en la vista Spinner (que por defecto tiene la primera opción seleccionada)
- Para este evento definimos dos métodos:
 - **onItemSelected**: será llamado cada vez que se seleccione una opción en la lista desplegable
 - **onNothingSelected**: se llamará cuando no haya ninguna opción seleccionada (esto puede ocurrir por ejemplo si el adaptador no tiene datos porque debe leerlos de una BD todavía vacía).

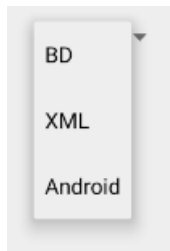
Método onItemSelected()



- Se invoca sólo cuando la posición recién seleccionada es diferente de la posición previamente seleccionada o si no había ningún elemento seleccionado.

```
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
    String[] cursos = { "1º", "1º", "2º" };  
    String resu = "El curso de " + parent.getItemAtPosition(position) + " es " + cursos[position];  
    textView.setText(resu);  
}
```

- Parámetros:



- parent El AdapterView donde ha sucedido la selección (el spinner)
- view La vista en la que se hizo click (el TextView proporcionado por android.R.layout.simple_spinner_dropdown_item)
- position La posición de esa vista en el adaptador
- id El ID de la fila del elemento que está seleccionado

- En la implementación del código del método se puede llamar a getItemAtPosition(position) si se necesita acceder a los datos asociados con el elemento seleccionado y, en este ejemplo, nos da lo mismo aplicarlo sobre parent que sobre spinner (son lo mismo!).

Soporte para idiomas



- En nuestros ejemplos, la construcción del adapter ha sido a partir de un array de String declarado y definido en el código:

```
String []opcion={"BD","XML","Android"};
```
- Pero qué ocurriría si quisieramos dotar a la aplicación de soporte para idiomas? No podríamos "internacionalizar" las String definidas en el código.
- Debemos separar diseño y lógica.

Separar diseño y lógica



strings.xml o fichero arrays.xml

```
<resources>
...
<string-array name="opciones">
    <item>BD</item>
    <item>XML</item>
    <item>Android</item>
</string-array>
<array name="cursos">
    <item>1º</item>
    <item>1º</item>
    <item>2º</item>
</array>
</resources>
```

Da lo mismo la etiqueta
array o string-array

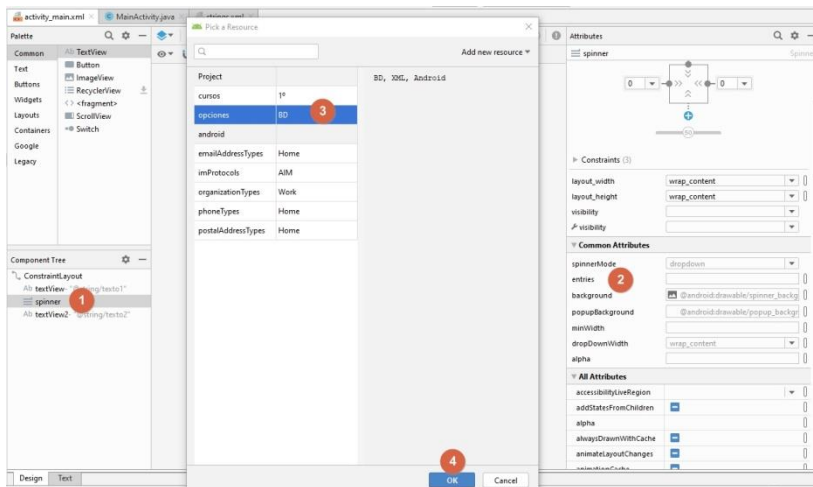
Código Java

- Para array cursos
**String[] cursos =
getResources().getStringArray(R.array.cursos);**
 - Para la construcción del Adapter hay 2 soluciones:
 - Solución 1:
**String []opcion=
getResources().getStringArray(R.array.opciones);**
 - Solución 2: No se declara ni define el array opcion y se usa el método createFromResource de la clase ArrayAdapter
- ```
ArrayAdapter<CharSequence> arrayAdapter =
ArrayAdapter.createFromResource(this, R.array.opciones,
android.R.layout.simple_spinner_dropdown_item);
```

# Más usos de string-array



- Si el Adapter es "estático" (no depende por ejemplo de los datos leídos de una BD), podríamos obviar su construcción por código y hacerlo directamente desde layout:



No hace falta Adapter en el código.  
Pero volvemos a mezclar diseño y lógica!

```
public class MainActivity extends AppCompatActivity {
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
 super.onCreate(savedInstanceState);
```

```
 setContentView(R.layout.activity_main);
```

```
 final TextView textView= findViewById(R.id.textView2);
```

```
 final Spinner spinner=findViewById(R.id.spinner);
```

```
 spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
 @Override
```

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

```
 String[] cursos = getResources().getStringArray(R.array.cursos);
```

```
 String resu = "El curso de " + parent.getItemAtPosition(position) + " es " + cursos[position];
 textView.setText(resu);
```

```
 }
```

```
 @Override
```

```
 public void onNothingSelected(AdapterView<?> parent) {
```

```
 textView.setText("Escoja asignatura");
```

```
 }
```

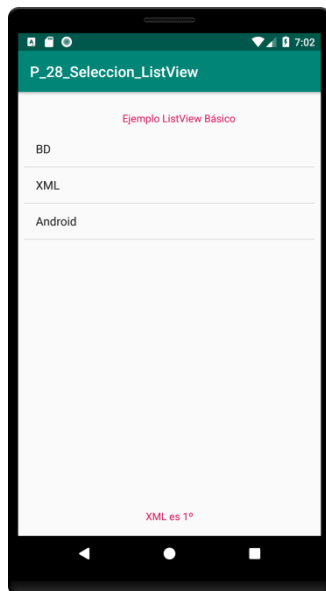
```
 };
```

```
}
```

# Vista ListView



- Una vista ListView muestra al usuario **una lista de opciones seleccionables** directamente sobre el propio control, sin listas emergentes como en el caso del control Spinner.
- En caso de existir más opciones de las que se pueden mostrar sobre la vista se podrá por supuesto hacer **scroll** sobre la lista para acceder al resto de elementos.
- (Ojo con el id que asigna el asistente!)
- AS lo marca como "legacy"



# OnItemClickListener – setOnClickListener - onItemClick



```
public class MainActivity extends AppCompatActivity {
```

```
 @Override
```

```
 protected void onCreate(Bundle savedInstanceState) {
```

```
 super.onCreate(savedInstanceState);
```

```
 setContentView(R.layout.activity_main);
```

```
 String[] opcion = { "BD", "XML", "Android" };
```

```
 final TextView textView = findViewById(R.id.textView2);
```

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

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

```
 listView.setAdapter(adapter);
```

```
 listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
```

```
 @Override
```

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

```
 String[] cursos = { "1º", "1º", "2º" };
```

```
 textView.setText(String.format("%s es %s", listView.getItemAtPosition(position), cursos[position]));
```

```
 }
```

```
 });
```

```
 }
```

```
}
```

Otro layout genérico de Android para los controles de tipo ListView formado únicamente por un TextView con unas dimensiones determinadas.





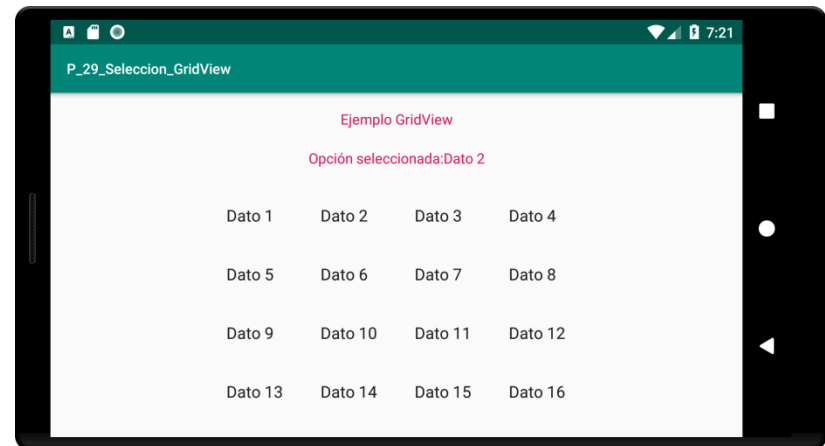
# Vista GridView



- Presenta al usuario un conjunto de opciones seleccionables distribuidas de forma tabular.
- Propiedades más importantes:
  - **android:numColumns**, indica el número de columnas de la tabla o “auto\_fit” si queremos que sea calculado por el propio sistema operativo a partir de las siguientes propiedades.
  - **android:columnWidth**, indica el ancho de las columnas de la tabla.
  - **android:horizontalSpacing**, indica el espacio horizontal entre celdas.
  - **android:verticalSpacing**, indica el espacio vertical entre celdas.
  - **android:stretchMode**, indica qué hacer con el espacio horizontal sobrante. Si se establece al valor “columnWidth” este espacio será absorbido a partes iguales por las columnas de la tabla. Si por el contrario se establece a “spacingWidth” será absorbido a partes iguales por los espacios entre celdas.
- AS lo marca como "legacy"

<GridView

```
android:id="@+id/gridView"
android:layout_width="368dp"
android:layout_height="0dp"
android:layout_marginBottom="8dp"
android:columnWidth="80dp"
android:horizontalSpacing="5dp"
android:numColumns="auto_fit"
android:stretchMode="columnWidth"
android:verticalSpacing="10dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/textView2" />
```



# Código



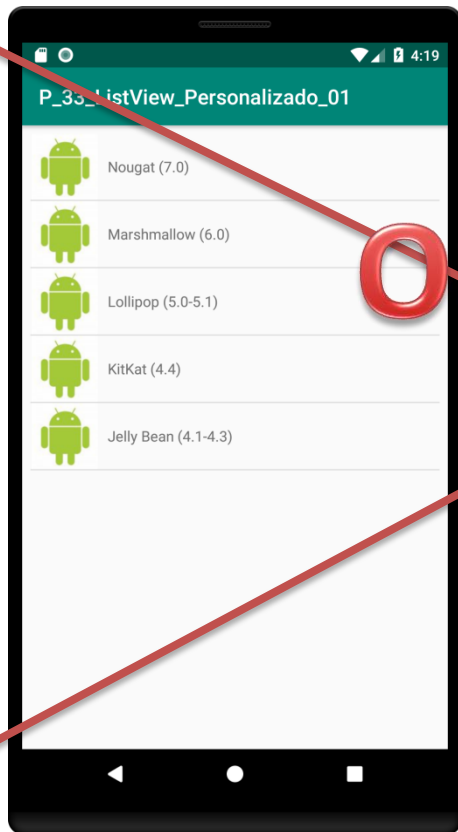
```
public class MainActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 final TextView textView=findViewById(R.id.textView2);
 String[] datos = new String[50];
 for(int i=1; i<=50; i++)
 datos[i-1] = "Dato "+i;
 ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, datos);
 GridView gridView = findViewById(R.id.gridView);
 gridView.setAdapter(arrayAdapter);
 gridView.setOnItemClickListener(
 new AdapterView.OnItemClickListener() {
 public void onItemClick(AdapterView<?> parent, android.view.View v, int position, long id) {
 textView.setText("Opción seleccionada: " + parent.getItemAtPosition(position));
 }
 }
);
 }
}
```

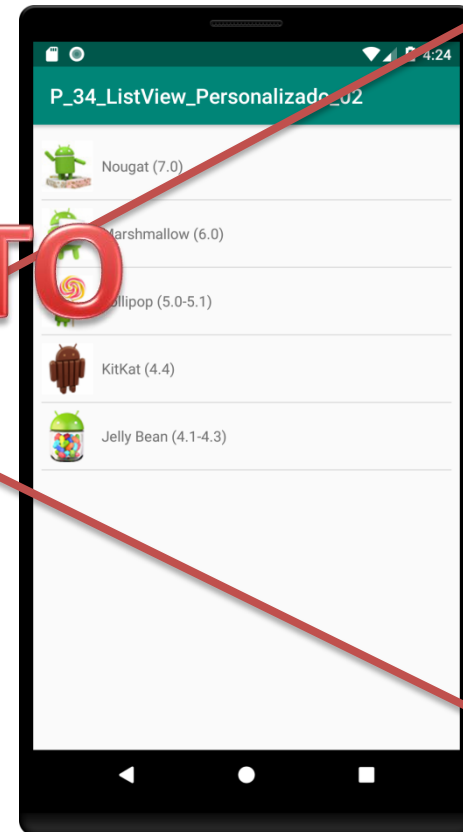
# ListView personalizados



Con layout propio para el item



Con adaptador propio

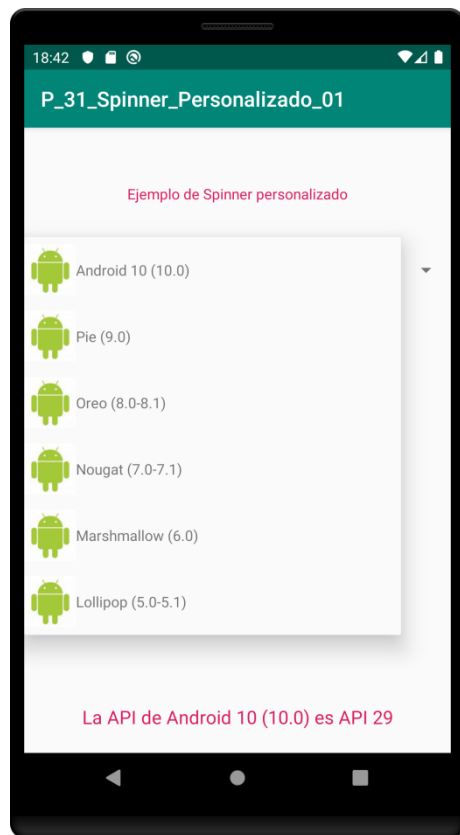


**OBSOLETO**

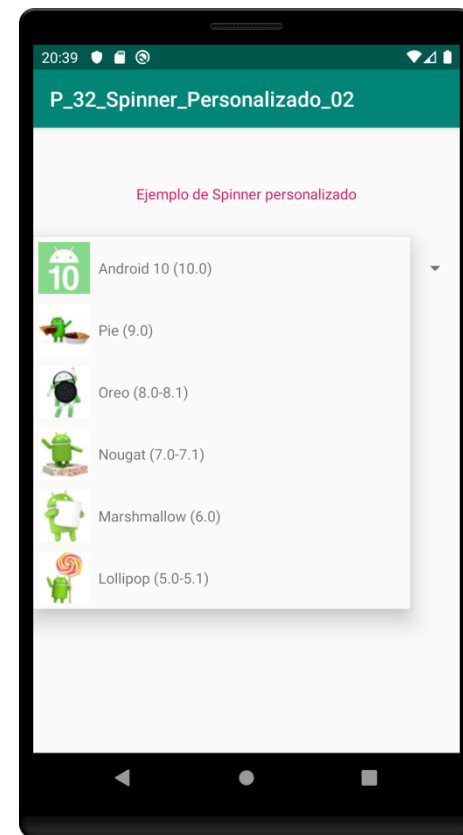
# Spinner personalizados



## Con layout propio para el item



## Con adaptador propio

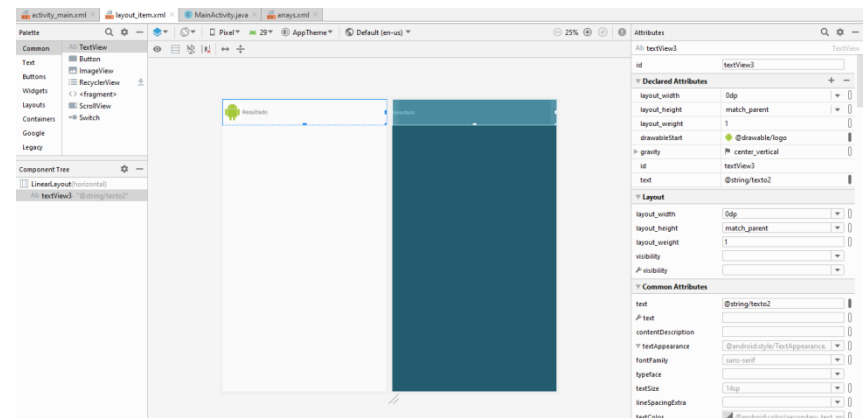
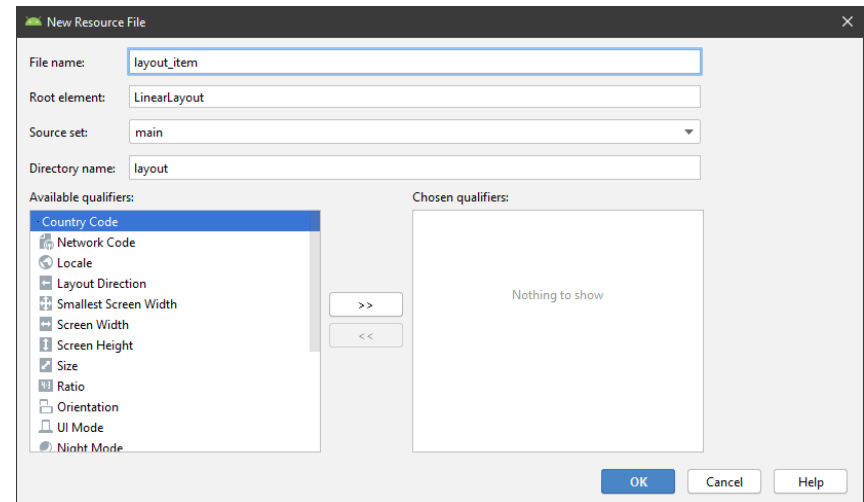


# Spinner con layout propio para el item



- Hay que crear un layout para el item (llamado, por ejemplo layout\_item) de tipo LinearLayout en horizontal.
- Es aconsejable establecer el alto del layout a partir de un parámetro de configuración del sistema para alturas de items en listas:  
**?android:attr/listPreferredItemHeight**
- Contiene un ImageView y a su derecha un TextView. Si el layout consta solo de estos dos elementos, Lint muestra el siguiente mensaje : *This tag and its children can be replaced by one and a compound drawable*. Es decir, es posible simplificar el layout eliminando la imagen y usarla dentro del elemento TextView como Compound Drawable.
- Por tanto, el layout tiene 1 TextView en el que fijaremos que el valor del atributo android:drawableStart sea la imagen deseada.
- Observación: Si en vez de LinearLayout hubiésemos escogido ConstraintLayout necesitaríamos las dos vistas.
- En el código Java de la actividad debe crearse el adaptador indicando el layout que debe utilizarse y el elemento de ese layout donde se visualizará el valor de cada item:

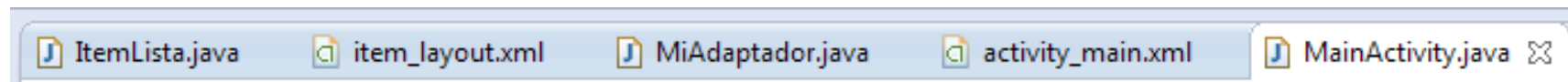
```
ArrayAdapter<String> arrayAdapter= new
ArrayAdapter<>(this, R.layout.layout_item,
R.id.textView3, opcion);
```



# Spinner con adaptador propio



- Si queremos algo más adaptable, como cambiar las imágenes y/o cambiar varios textos, tendremos que escribir nuestro propio adaptador extendiendo la clase BaseAdapter.
- Pasos: (Usados y explicados en ejercicio 2)
  1. Diseñar el layout propio del item
  2. Si el item de la lista contiene objetos diversos (imágenes, textos,...) para trabajar mejor nos crearemos una clase que lo defina.
  3. Personalizar el Adapter usando el layout propio y la clase anterior
  4. En la actividad que contiene el Spinner asignarle el Adapter personalizado



# Personalizar el Adapter



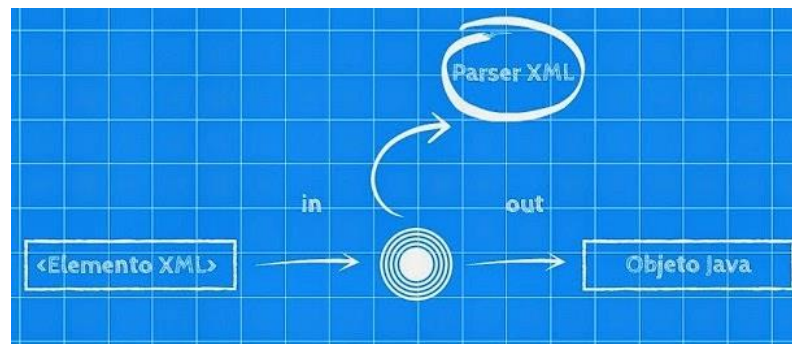
- Tendremos que crear una nueva clase (de nombre, por ejemplo, MiAdaptador) extendiendola de la clase BaseAdapter, para tener nuestro propio adaptador de ListView .
- En esta clase habrá que sobrescribir los siguientes cuatro métodos:
  - View **getView**(int position, View convertView, ViewGroup parent)
    - Este método ha de construir un nuevo objeto View con el Layout correspondiente a la posición position y devolverlo.
    - El último parámetro corresponde al padre al que la vista va a ser añadida.
  - int **getCount**()
    - Devuelve el número de elementos de la lista.
  - Object **getItem**(int position)
    - Devuelve el elemento en la posición position de la lista.
  - long **getItemId**(int position)
    - Devuelve el identificador de fila en la posición position de la lista.

# Clase LayoutInflater

## Método inflate()



- El método más importante para personalizar el adapter es getView() el cual tiene que construir los diferentes layouts que serán añadidos en la lista a partir del código xml definido en el diseño del layout propio para el item.
- Este trabajo se realiza por medio del método inflate() de la clase LayoutInflater.
- El LayoutInflater actúa como puente entre el XML del layout y los objetos de tipo View, inflando ("parseando") los objetos XML del layout para que podamos usarlos, convirtiéndolos en un objeto java.



Ver ejercicio  
P\_32 con  
explicación en  
código!



# Prácticas propuestas



- Realiza la hoja de ejercicios  
Ejercicios\_08\_Controles\_de\_selección