Adapters

Algunos form widgets mas elaborados, que muestran una colección de ítems los cuales poseen información, tanto en forma de texto como de imagen, necesitan una forma standard y genérica de obtener la información que muestran en pantalla.
Por este motivo se utiliza el patrón de diseño "adapter" el cual "adapta" la información a mostrar, en un formato que es interpretado por todos los form widgets que lo implementan.

Algunos de los form widgets que utilizan adapters son:

- **ListView :** Muestra una lista de ítems.
- **GridView :** Muestra una grilla de ítems.
- **RecyclerView:** Reemplazo de ListView y GridView en Android 5.0
- **Spinner :** Muestra un combo de opciones.

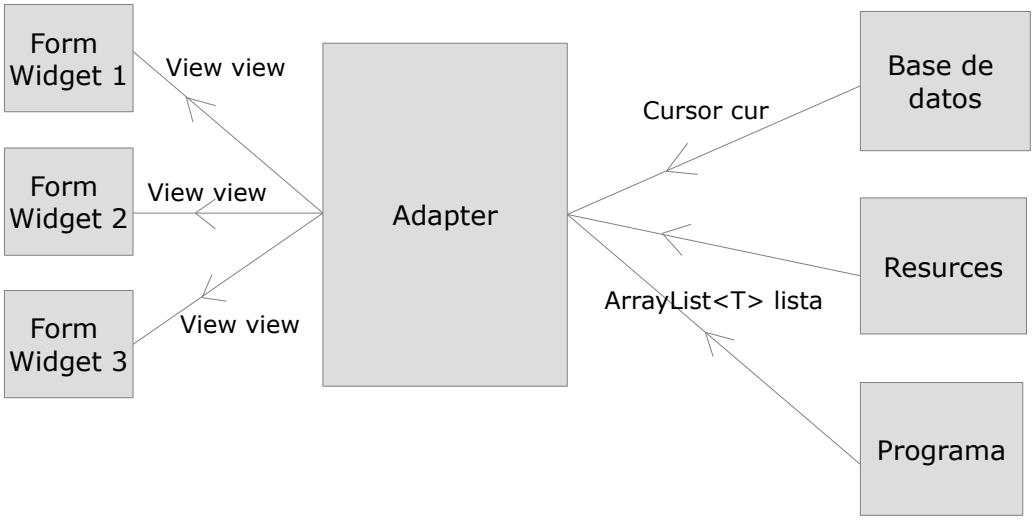
Algunos de los adapters que provee la API de Android son los siguientes:

- **ArrayAdapter<T>:** Adapter que almacena la información de los ítems en forma de array de objetos. Es muy utilizado para ListViews.
- **SimpleAdapter:** Utilizado para cargar información estática (proveniente de resources)
- **CursorAdapter:** Provee la información de los ítems a mostrar desde un objeto Cursor (El cual es devuelto por una consulta a una base de datos)
- **RecyclerView.Adapter:** Adapter para RecyclerView.

También pueden definirse adapters propios, extendiendo de la clase BaseAdapter o de cualquiera de las nombradas anteriormente.

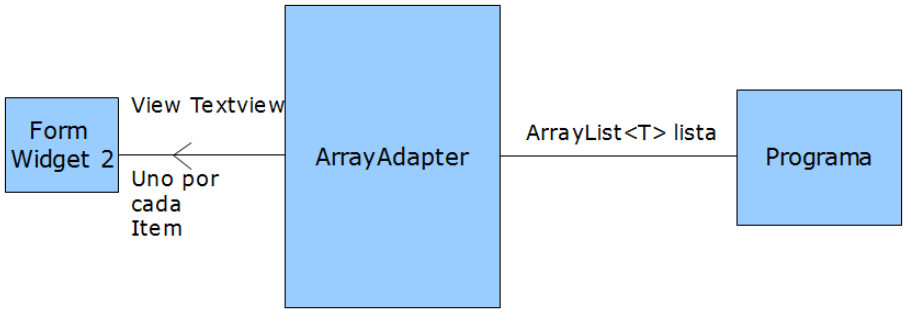
Funcionamiento del Adapter

¿Qué es lo que hace el adapter, para que todos los form widgets puedan entender la información que éste entrega? La respuesta a esta pregunta es la siguiente : El adapter genera un objeto del tipo View por cada ítem que el form widget debe mostrar. Este objeto View contiene la información que debe aparecer en pantalla y que el adapter toma de las estructuras de datos que se le proporcionan (Listas de objetos,imágenes, Cursores, etc.)



Clase ArrayAdapter – Ejemplo utilizando el form widget "Spinner"

Esta clase recibe una List de cualquier tipo de dato, y devuelve al form widget (En este ejemplo usaremos un spinner) un objeto View que contiene un TextView. El texto contenido en el TextView es obtenido de llamar al método "toString()" de los objetos de la List.



Si se pretende utilizar un tipo de dato propio, puede hacerse override del método "toString()" y devolver el texto que se pretende observar en el ítem del form widget que esta mostrando la información.

A continuación, mostraremos un ejemplo utilizando el tipo de dato String, cuyo método "toString()" devuelve el propio String.

Primero, creamos una List de objetos String:

```
List<String> listaPalabras = new ArrayList<String>();

listaPalabras.add("Uno");
listaPalabras.add("Dos");
listaPalabras.add("Tres");
```

Luego creamos el objeto ArrayAdapter, el constructor de esta clase posee 3 parámetros, el primero es el Context (en este caso this, se refiere a la Activity donde se encuentra este código), el segundo, es el id de un archivo xml de layout, el cual debe contener un TextView (utilizamos un layout de la clase R de android que existe para este propósito), y por último la List con los datos.

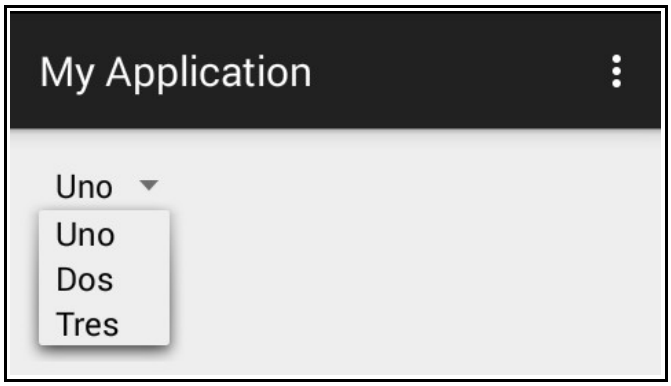
```
ArrayAdapter<String> adapter = new ArrayAdapter<String>
                                        (this,
                                         android.R.layout.simple_spinner_item,
                                         listaPalabras);
```

Por último, asignamos el adapter creado, a un objeto del tipo "Spinner", el cual definimos en el layout que utiliza la Activity y obtenemos mediante *findViewById*.

```
Spinner combo = (Spinner) findViewById(R.id.spinner1);
combo.setAdapter(adapter);
```

Una vez seteado el adapter, el objeto Spinner que se muestra en pantalla, utilizará el TextView que se le pasó al adapter en el constructor para mostrar las opciones.

Resultado de la ejecución del programa:



¿Qué hizo el adapter? El adapter, devolvió al Spinner, objetos del tipo View, creados desde el xml que se le pasó en el constructor (el que provenía de la clase R de android), seteando el texto del objeto TextView, con el valor que contenía la List de Strings que también se le pasó en el constructor.

Obtener ítem seleccionado del Spinner

Mediante el método "getSelectedItem()" podemos obtener el objeto de la List que fue seleccionado:

```
String selec = (String)combo.getSelectedItem();
```

Spinner con Material Design

A partir de Android 5.0, el spinner puede tener una línea debajo, la cual se pinta del color "acento" cuando se hace click sobre el mismo, para obtener esta funcionalidad en todas las versiones de Android, deberemos asegurarnos que nuestra Activity herede de *AppCompatActivity* y se colocar la siguiente propiedad en el archivo XML de Layout en el tag del Spinner:

```
style="@style/Widget.AppCompat.Spinner.Underlined"
```

AutoCompleteTextView – Otro ejemplo de ArrayAdapter

Este objeto es un campo de texto al igual que el EditView, con la diferencia de que sugiere al usuario opciones según lo que éste escriba. Las opciones sugeridas se deben pasar al objeto AutoCompleteTextView a través de un ArrayAdapter.

Definición en XML:

```
<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="AutoCompleteTextView" >
```

Para utilizarlo, primero, definimos una List, la cual contendrá las palabras que serán sugeridas durante la escritura en el campo de texto.

```
List<String> listaSugerencias = new ArrayList<String>();
```

Cargamos la lista:

```
listaSugerencias.add("Argel");
listaSugerencias.add("Argelia");
listaSugerencias.add("Argentina");
listaSugerencias.add("Armenia");
```

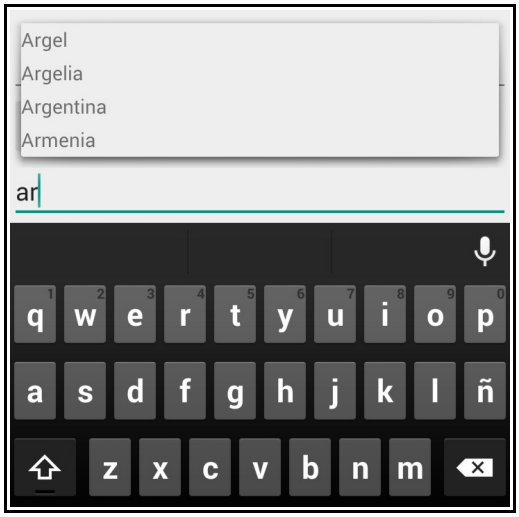
Definimos un ArrayAdapter, como en el ejemplo anterior, en este caso, utilizaremos el layout que provee la API llamado "test_list_item":

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

Seteamos el adapter al objeto mediante el metodo "setAdapter()":

```
AutoCompleteTextView text = (AutoCompleteTextView)
                                        findViewById(R.id.autoCompleteTextView1);
text.setAdapter(adapter);
```

Al escribir las primeras dos letras en el campo de texto, aparecerán las sugerencias:



Para obtener el texto escrito, utilizamos el mismo código que para un EditBox común:

```
AutoCompleteTextView text = (AutoCompleteTextView)
                                        findViewById(R.id.autoCompleteTextView1);

String txtIngresado = text.getText().toString();
```