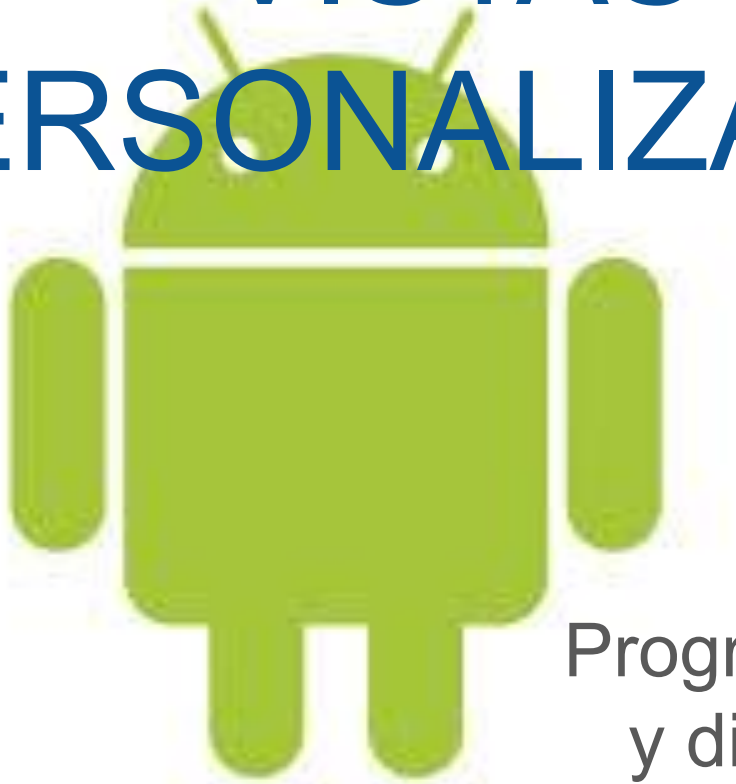
VISTAS PERSONALIZADAS



Programación multimedia
y dispositivos móviles
2º DAM

ÍNDICE DE CONTENIDOS

1. Introducción
2. Layout XML
3. Clase Modelo Datos
4. Adaptador con ArrayAdapter

1. INTRODUCCIÓN

- En el tema anterior ya vimos cómo crear controles de selección (Spinner, listas y tablas) en los que podíamos mostrar un texto.

- En ocasiones, queremos crear elementos que, a parte de un sencillo texto, tengan otros elementos como imágenes o bien contengan otros tipos de vistas.

- Para resolver este problema, Android nos permite crear nuestros propios elementos personalizados que después podremos mostrar dentro de cualquier control de selección.

1. INTRODUCCIÓN

ArrayList

Spinner
ListView
GridView

1. INTRODUCCIÓN

Para crear nuestros controles personalizados debemos trabajar con 3 archivos:

- **Layout del adaptador** → Tenemos que indicar cómo se distribuyen los elementos de la vista para un elemento.
- **Clase Java** → Esta clase será la que forme la estructura de los datos a mostrar. Los datos que vamos a poner en cada uno de los ítem tienen que tener los getter para poder acceder a ellos.
    - El ArrayList de los datos, debe ser de este tipo de objeto.
- **Clase Adaptador** → Esta clase será la encargada de transformar los elementos que queremos visualizar dentro del control en elementos de nuestro tipo personalizado.

2. LAYOUT XML

El primer paso será crear el xml para indicar qué componentes se deben visualizar por cada ítem que queremos mostrar.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/nombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/telefono"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/provincia"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

En el ejemplo que vamos a ver, vamos a mostrar información de varias personas.

De cada persona queremos ver su nombre, su teléfono y su provincia.

Por lo que en el xml crea un textView para cada uno de estos datos.

3. CLASE MODELO DATOS

Una vez establecido el xml, debemos crear una clase en Java con los datos que vamos a necesitar.

```java
public class Persona {

    private String nombre;
    private String telefono;
    private String provincia;
```

En esta clase, tendremos que establecer constructores, getters, setters y toString.

4. ADAPTADOR CON ARRAYADAPTER

Como vimos en el tema anterior, para poder llenar el control de elementos debemos usar un intermediario, **el adaptador**.

Para los casos en los que usamos el tipo de elementos por defecto, podemos usar los adaptadores que nos proporciona el propio Android, sin embargo, puesto que hemos creado un elemento customizado, necesitaremos crear un adaptador que se amolde a este nuevo paradigma.

Para crear este nuevo Adaptador, simplemente debemos crear una nueva clase Java que extienda de la clase ArrayAdapter:

```
public class AdaptadorPersonalizado extends ArrayAdapter {
```

4. ADAPTADOR CON ARRAYADAPTER

Esta clase debe tener, como atributos:

- El contexto de la aplicación
- Un ArrayList<Clase> de la clase que creamos para estructurar los datos.

```
public class AdaptadorPersonalizado extends ArrayAdapter {
    ArrayList<Persona> personas;
    Context context;
```

4. ADAPTADOR CON ARRAYADAPTER

Podemos utilizar alguno de los constructores que vienen por defecto o podemos crear uno nuevo.

- El constructor siempre debe recibir el contexto de la actividad en la que se va a mostrar.
- Normalmente, recibirá también la lista de los elementos a mostrar.

En este constructor, tendremos que invocar al constructor de la clase ArrayAdapter (super), para enviarle estos datos, junto con el xml que debe utilizar.

También se guardarán los valores de los atributos.

```
public AdaptadorPersonalizado(Context context, ArrayList<Persona> lista) {
    super (context, R.layout.item, lista);
    this.personas = new ArrayList<Persona>();
    this.personas = lista;
    this.context = context;
}
```

4. ADAPTADOR CON ARRAYADAPTER

getView → Este método es realmente el encargado de convertir los elementos Java provenientes del ArrayList en elementos de tipo View que podemos insertar en nuestro control de selección. Este método se ejecuta por cada elemento de la lista, por lo que el atributo posicion, será el índice que tiene.

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
```

4. ADAPTADOR CON ARRAYADAPTER

LayoutInflater → Es una clase de la API de Android de la que nos vamos a servir para poder insertar los elementos personalizados dentro del control.

- Se define a partir del contexto de la aplicación
- Es la encargada de indicar a la View que formato debe adoptar, “inflando” el xml que creamos al principio.

```
LayoutInflater inflater = LayoutInflater.from(context);
```

```
LayoutInflater inflater = LayoutInflater.from(getContext());
```

4. ADAPTADOR CON ARRAYADAPTER

LayoutInflater → Es una clase de la API de Android de la que nos vamos a servir para poder insertar los elementos personalizados dentro del control.

- Se define a partir del contexto de la aplicación
- Es la encargada de indicar a la View que formato debe adoptar, “inflando” el xml que creamos al principio.

```
LayoutInflater inflater = LayoutInflater.from(context);
View vista = inflater.inflate(R.layout.item,  root: null);
```

4. ADAPTADOR CON ARRAYADAPTER

Creamos los objetos de tipo vista y los relacionamos con las vistas del XML.

```
TextView textViewNombre = vista.findViewById(R.id.nombre);
TextView textViewTelefono = vista.findViewById(R.id.telefono);
TextView textViewProvincia = vista.findViewById(R.id.provincia);
```

Asignamos los valores a estas vistas obteniéndose a partir del ArrayList que creamos dentro de nuestra clase Adaptador.

```
textViewNombre.setText(this.personas.get(position).getNombre());
textViewTelefono.setText(this.personas.get(position).getTelefono());
textViewProvincia.setText(this.personas.get(position).getProvincia());
```

4. ADAPTADOR CON ARRAYADAPTER

El método getView devuelve un objeto de tipo View, por lo que una vez que se ha creado cada uno de los componentes, devolveremos la vista sobre la que hemos trabajado.

```
return vista;
```

4. ADAPTADOR CON ARRAYADAPTER

Para hacer uso de nuestro nuevo y flamante control personalizado simplemente debemos hacerlo como si de un control habitual se tratase:

- Creamos un ArrayList con los elementos que queremos mostrar en el control
- Lo usamos para definir un nuevo Adaptador
- Asignamos este nuevo adaptador a nuestro control

```
ArrayList<Persona> personas = new ArrayList<Persona>();
personas.add(new Persona( nombre: "Ana",  telefono: "6666666666",  provincia: "Salamanca"));
personas.add(new Persona( nombre: "María",  telefono: "677777777",  provincia: "Zaragoza"));
personas.add(new Persona( nombre: "Marta",  telefono: "699999999",  provincia: "Madrid"));

AdaptadorPersonalizado miAdaptador = new AdaptadorPersonalizado( context: this, personas);
listView.setAdapter(miAdaptador);
```

5. DUDAS Y PREGUNTAS