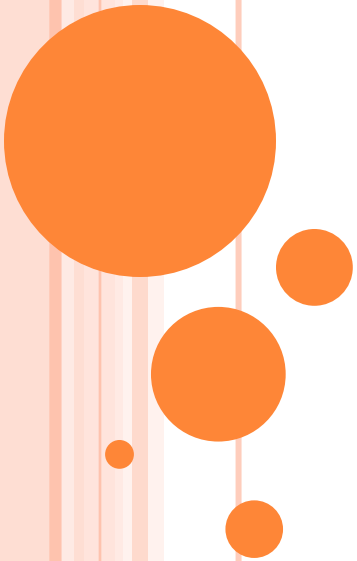


ListView

Acceso a datos

Uso de ListView

- Insertamos el control de tipo ListView en el layout en el que vamos a mostrar la lista de valores (layout1.xml).
- Creamos un archivo layout en el que diseñamos detalladamente la apariencia de cada uno de los elementos visuales del ListView (layout2.xml).
- Implementamos una clase Adapter que es la que se encarga de mostrar los elementos de la lista de valores en cada uno de los elementos visuales del ListView (ClaseAdapter.java).

Clase Adapter I

- La clase Adapter se encarga de visualizar la lista de valores en el ListView.
- public class Adaptador extends **ArrayAdapter<Tipo>** {
  private Context contexto;
  private ArrayList<Tipo> lista;
  public Adaptador(**Context c, ArrayList<Tipo> lista**) {
  }
  @Override
  public View **getView(int pos, View v, ViewGroup vg)** {
  }
  }

Clase Adapter II

- La clase extiende ArrayAdapter<Tipo>, donde Tipo es el tipo básico de los elementos de la lista.
- Al constructor se le debe pasar el contexto así como la lista de valores completa. Lo habitual será guardar estos valores en variables de instancia.
- El contexto será la actividad en la que esté insertado el control de tipo ListView.
- Se sobrescribirá el método getView(int, View, ViewGroup). En este método se indicará cómo se debe representar el elemento de la lista de valores, cuya posición se pasa como parámetro.

Método getView()

- public View getView(int pos, View v, ViewGroup g) {
  LayoutInflater i = (LayoutInflater)
    contexto.getSystemService(
                Context.LAYOUT_INFLATER_SERVICE);
  v = i.inflate(R.layout.layout2, null);
  //obtenemos el elemento pos de la lista
  //asignamos los valores a los elementos del layout2
  //devolvemos la vista generada
  return v;
  }
- Este método se puede mejorar reutilizando los elementos ya usados.

Click sobre un elemento del ListView

```
ListView lv = (ListView) findViewById(R.id.lvLista);

lv.setOnItemClickListener(new OnItemClickListener() {
   public void onItemClick(AdapterView<?> av, View v,
                                                int pos, long id) {
      // la variable pos indica el elemento seleccionado
      Tipo t = lista.getItemAtPosition(pos);
      …
   }
});
```

LongClick sobre el ListView

```
lv.setOnItemLongClickListener(new
   OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?>
                                   av, View v, int pos, long id) {
        Tipo t = lista.getItemAtPosition(pos);
        …
        return true;
    }
});
```

ContextMenu sobre un ListView I

- Para mostrar un menú contextual al seleccionar un elemento del ListView no se puede implementar el OnItemLongClickListener, puesto que son incompatibles.
- Se debe registrar el control ListView como elemento que va a mostrar un menú contextual:

  registerForContextMenu(lv);
- Se deben implementar los métodos

  onCreateContextMenu()

  y

  onContextItemSelected()

ContextMenu sobre un ListView II

```
@Override
public void onCreateContextMenu(ContextMenu
                menu, View v, ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.menucontextual, menu);
}
```

Este método es el que muestra el menú contextual al realizar una pulsación larga sobre el elemento.

ContextMenu sobre un ListView III

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="…" >
    <item android:id="@+id/mneditar"
        android:title="@string/str_editar"/>
    <item android:id="@+id/mnborrar"
        android:title="@string/str_borrar" />
</menu>
```

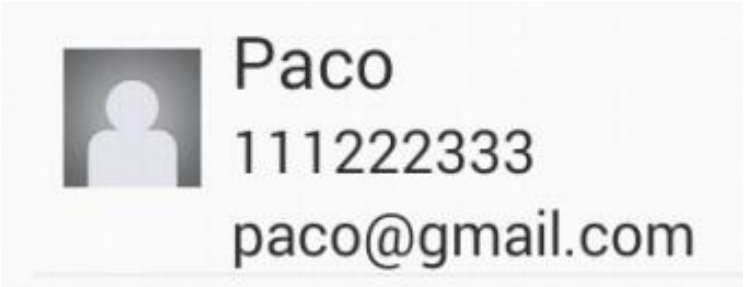
En res/menu/ creamos el archivo menucontextual.xml que contiene las opciones del menú.

ContextMenu sobre un ListView IIII

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo)
                                                    item.getMenuInfo();
    int id = item.getItemId();
    switch (item.getItemId()) {
        case R.id.mneditar:
            …; // info.position indica el nº elemento a editar
            return true;
        case R.id.mnborrar:
            … // info.position indica el nº elemento a borrar
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

Click sobre un View del ListView I

- Si los elementos del ListView contienen, por ejemplo, una imagen, se puede implementar el evento onClick sobre la imagen.
- El problema es que en el gestor de evento no será posible saber qué elemento exacto del ListView se ha pulsado.
- Una posible solución puede consistir en almacenar información en el propio control que permita identificar de forma única el elemento que haya sido pulsado.



Click sobre un View del ListView II

- Asociación del evento.

```
imagen.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        …
    }
}); //en el método getView de la clase adaptador
```

o

```
<ImageView android:onClick="clicImagenUsuario" />
```

y

```
public void clicImagenUsuario(View v){…} //en la clase activity
```

Click sobre un View del ListView III

- En el método getView() del adaptador, obtenemos la imagen a la que queremos asociar el gestor de eventos y guardamos información sobre el dato actual en el atributo Tag del ImageView: setTag().
- En el gestor de evento, extraemos del parámetro View el atributo Tag: getTag().

Actualización del ListView

- Si desde la actividad principal, implementamos la edición y el borrado de elementos de la lista de valores del control ListView, hemos de informar al objeto adaptador de que se ha producido un cambio en los datos, con el fin de que éstos sean mostrados correctamente:

```
adaptador.notifyDataSetChanged();
```