Curso de programación Android

T-Formación

Alejandro Alcalde

elbauldelprogramador.com

17 de marzo de 2015

Contenidos

Contenidos I

Contenidos I

Conceptos básicos Android

Todos los componentes de la interfaz de usuario de Android descienden de la clase *View*. Dichos objetos están organizados en forma de árbol y pueden contener nuevos objetos View, permitiendo crear interfaces muy completas.
Los objetos *View* se pueden definir de dos maneras:

- Mediante un fichero XML colocado dentro del directorio *res/layout*, que es el que usaremos normalmente.

Conceptos básicos Android

Todos los componentes de la interfaz de usuario de Android descienden de la clase *View*. Dichos objetos están organizados en forma de árbol y pueden contener nuevos objetos View, permitiendo crear interfaces muy completas.
Los objetos *View* se pueden definir de dos maneras:

- Mediante un fichero XML colocado dentro del directorio *res/layout*, que es el que usaremos normalmente.
- En tiempo de ejecución, muy útil para crear nuestros propios componentes View.

Para dibujar la interfaz, el sistema necesita que le pasemos el objeto View raiz, para ir descendiendo por cada uno de sus nodos y presentar al usuario toda la interfaz. El método encargado de esto es *Activity.setContentView()*.

Conceptos básicos Android

Para que Android sepa dibujar correctamente los objetos, tenemos que pasarle algunos datos, como son la altura y anchura. Para eso nos servimos de la clase *LayoutParams*, que puede tomar los siguientes valores:

- Un número.

Conceptos básicos Android

Para que Android sepa dibujar correctamente los objetos, tenemos que pasarle algunos datos, como son la altura y anchura. Para eso nos servimos de la clase *LayoutParams*, que puede tomar los siguientes valores:

- Un número.
- La constante *MATCH_PARENT*, que indica que la vista debe intentar ser tan grande como su padre, quitando el padding.

Conceptos básicos Android

Para que Android sepa dibujar correctamente los objetos, tenemos que pasarle algunos datos, como son la altura y anchura. Para eso nos servimos de la clase *LayoutParams*, que puede tomar los siguientes valores:

- Un número.
- La constante *MATCH_PARENT*, que indica que la vista debe intentar ser tan grande como su padre, quitando el padding.
- La constante *WRAP_CONTENT*, para que intente ser lo suficientemente grande para mostrar su contenido, mas el padding.

Conceptos básicos Android

Un atributo imprescindible es el *id*(de tipo entero). Que sirve para identificar únicamente a un objeto View. Cuando lo declaramos mediante xml podemos referenciarlo a través de la clase de recursos R, usando una @.
Los objetos View pueden tener otros muchos atributos, como padding, colores, imágenes, fondos, márgentes etc.

- ***android:id="@+id/nombreID":*** Crea un nuevo atributo en la clase R llamado nombreID.

Conceptos básicos Android

Un atributo imprescindible es el *id*(de tipo entero). Que sirve para identificar únicamente a un objeto View. Cuando lo declaramos mediante xml podemos referenciarlo a través de la clase de recursos R, usando una @.
Los objetos View pueden tener otros muchos atributos, como padding, colores, imágenes, fondos, márgentes etc.

- ***android:id="@+id/nombreID":*** Crea un nuevo atributo en la clase R llamado nombreID.
- ***android:id="@id/nombreID":*** Hace referencia a un id ya existente asociado a la etiqueta 'nombreID'.

Conceptos básicos Android

Un atributo imprescindible es el *id*(de tipo entero). Que sirve para identificar únicamente a un objeto View. Cuando lo declaramos mediante xml podemos referenciarlo a través de la clase de recursos R, usando una @.
Los objetos View pueden tener otros muchos atributos, como padding, colores, imágenes, fondos, márgentes etc.

- ***android:id="@+id/nombreID":*** Crea un nuevo atributo en la clase R llamado nombreID.
- ***android:id="@id/nombreID":*** Hace referencia a un id ya existente asociado a la etiqueta 'nombreID'.
- ***android:id="@android:id/list":*** Referencia a un a etiqueta definida en la clase R del sistema llamada 'list'.

Contenidos I

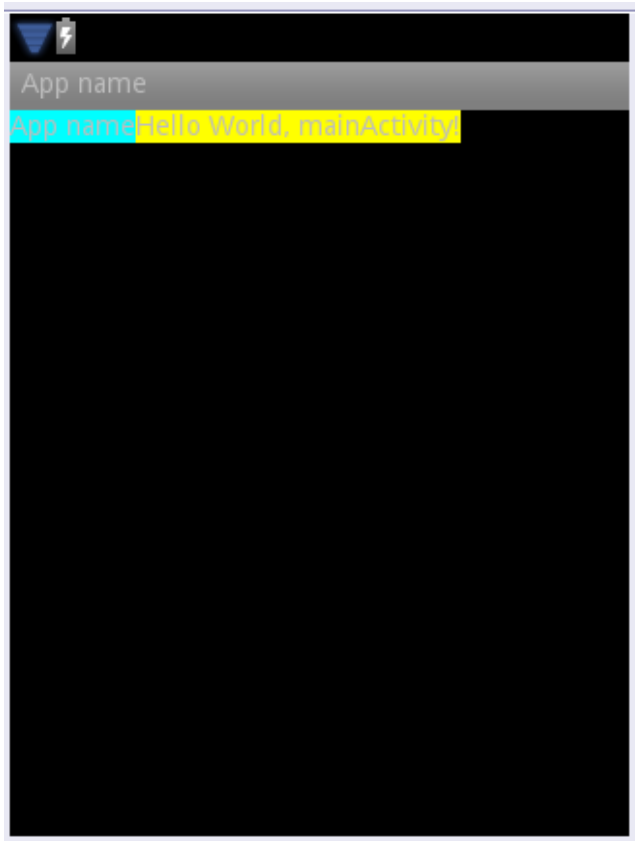
Qué es un layout

Los layout nos permiten posicionar cada objeto gráfico en el lugar que queramos de la pantalla, es decir, nos permite diseñar el aspecto gráfico que va a tener nuestra pantalla. Los layouts son de tipo ViewGroup, una subclase de View.
Existen varios tipos de Layouts para Android, vamos a ver los más comunes:

Contenidos I

LinearLayout

Este tipo de layout coloca sus hijos unos detrás de otros, comenzando por la esquina superior izquierda de la pantalla. Podemos colocarlos alineados horizontalmente o verticalmente mediante su propiedad *android:orientation="horizontal | vertical"*.



LinearLayout

```
<linearlayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <textview android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/app_name"
        android:background="#0ff"/>

    <textview android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:background="#ff0"/>
</linearlayout>
```

Contenidos I

RelativeLayout

Este Layout permite que coloquemos los elementos en un lugar con respecto a la posición de otro, es decir, colocar un botón a la derecha de un texto, o centrarlo en la pantalla, o por ejemplo, colocar un texto encima de tal elemento y a la derecha de este otro.
Para conseguir esto, *RelativeLayout* proporciona propiedades como *android:layout_toRightOf* o *android:layout_alignLeft*, que toman como valores los identificadores de los objetos, o valores booleanos.

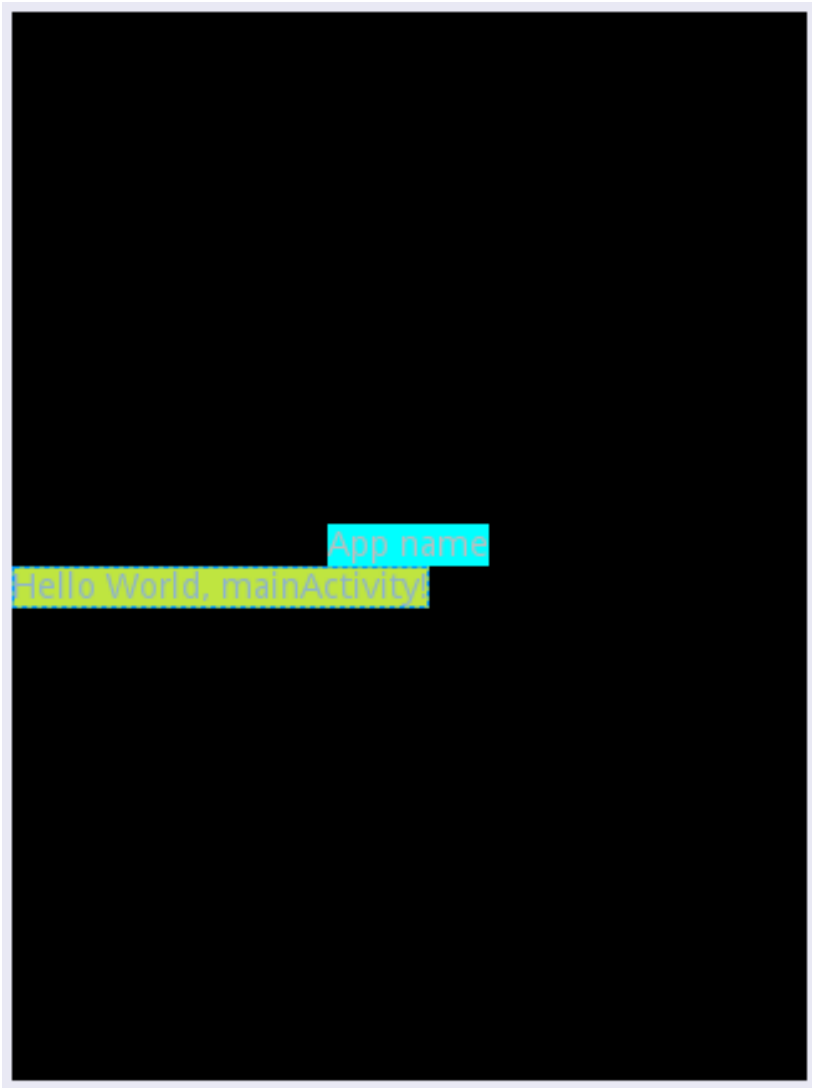


Figura: RelativeLayout

RelativeLayout

```
<relativelayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <textview android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/app_name"
        android:background="#0ff"
        android:layout_centerInParent="true"
        android:id="@+id/text1"/>
    <textview android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:background="#ff0"
        android:layout_below="@id/text1"/>
</relativelayout>
```

Contenidos I

Notificaciones y Diálogos

En ocasiones hay que mostrar mensajes al usuario para informarle del estado de la aplicación, o del estado de las operaciones que se estén llevando a cabo. Hay 3 tipos:

- Notificaciones Toast.

Notificaciones y Diálogos

En ocasiones hay que mostrar mensajes al usuario para informarle del estado de la aplicación, o del estado de las operaciones que se estén llevando a cabo. Hay 3 tipos:

- Notificaciones Toast.
- Notificaciones en la barra de estado.

Para dibujar la interfaz, el sistema necesita que le pasemos el objeto View raiz, para ir descendiendo por cada uno de sus nodos y presentar al usuario toda la interfaz. El método encargado de esto es *Activity.setContentView()*.

Notificaciones y Diálogos

En ocasiones hay que mostrar mensajes al usuario para informarle del estado de la aplicación, o del estado de las operaciones que se estén llevando a cabo. Hay 3 tipos:

- Notificaciones Toast.
- Notificaciones en la barra de estado.
- Diálogos.

Para dibujar la interfaz, el sistema necesita que le pasemos el objeto View raiz, para ir descendiendo por cada uno de sus nodos y presentar al usuario toda la interfaz. El método encargado de esto es *Activity.setContentView()*.

Contenidos I

Fragments

- Permiten encapsular componentes de la interfaz para reutilizarlos.

Fragments

- Permiten encapsular componentes de la interfaz para reutilizarlos.
- Clase Fragment, puede definir su propio layout y maneja su ciclo de vida.

Fragments

- Permiten encapsular componentes de la interfaz para reutilizarlos.
- Clase Fragment, puede definir su propio layout y maneja su ciclo de vida.
- Se puede configurar junto con otros fragments dentro de una actividad para adaptarlo al tamaño de la pantalla.

Contenidos I

Persistencia de datos

La mayoría de apps necesitan guardar datos, ya sea para no perder la información cuando se ejecuta el método onPause(), información de preferencias o cantidades mayores de información en bases de datos. Los distintos métodos de almacenamiento disponibles son:

Persistencia de datos

La mayoría de apps necesitan guardar datos, ya sea para no perder la información cuando se ejecuta el método onPause(), información de preferencias o cantidades mayores de información en bases de datos. Los distintos métodos de almacenamiento disponibles son:

- Pares clave-valor de tipos de datos simples en un fichero Shared Preference.

Persistencia de datos

La mayoría de apps necesitan guardar datos, ya sea para no perder la información cuando se ejecuta el método onPause(), información de preferencias o cantidades mayores de información en bases de datos. Los distintos métodos de almacenamiento disponibles son:

- Pares clave-valor de tipos de datos simples en un fichero Shared Preference.
- Ficheros de cualquier tipo en el sistema de archivos.

Persistencia de datos

La mayoría de apps necesitan guardar datos, ya sea para no perder la información cuando se ejecuta el método onPause(), información de preferencias o cantidades mayores de información en bases de datos. Los distintos métodos de almacenamiento disponibles son:

- Pares clave-valor de tipos de datos simples en un fichero Shared Preference.
- Ficheros de cualquier tipo en el sistema de archivos.
- Bases de datos en SQLite.

Contenidos I

Interactuar con otras aplicaciones

Ya hemos visto cómo usar Intents para comunicar actividades de la misma app.
Cuando un intent se lanza al sistema en lugar de a otra activiy, se usa para iniciar el componente apropiado de una aplicación.
Hay dos tipos de Intents:

Interactuar con otras aplicaciones

Ya hemos visto cómo usar Intents para comunicar actividades de la misma app.
Cuando un intent se lanza al sistema en lugar de a otra activiy, se usa para iniciar el componente apropiado de una aplicación.
Hay dos tipos de Intents:

- *Explícito:* Para iniciar un componente específico (Un activity concreto).

Interactuar con otras aplicaciones

Ya hemos visto cómo usar Intents para comunicar actividades de la misma app.
Cuando un intent se lanza al sistema en lugar de a otra activiy, se usa para iniciar el componente apropiado de una aplicación.
Hay dos tipos de Intents:

- *Explícito:* Para iniciar un componente específico (Un activity concreto).
- *Implícito:* Para iniciar cualquier componente que pueda controlar la acción requerida. (Echar una foto).

Interactuar con otras aplicaciones

Con los intents se puede:

- Mover de una pantalla a otra dentro de nuestra app.

Interactuar con otras aplicaciones

Con los intents se puede:

- Mover de una pantalla a otra dentro de nuestra app.
- Mandar al usuario a la pantalla de otra aplicación para realizar una acción.

Interactuar con otras aplicaciones

Con los intents se puede:

- Mover de una pantalla a otra dentro de nuestra app.
- Mandar al usuario a la pantalla de otra aplicación para realizar una acción.
- Cualquier pantalla a la que invoquemos puede devolver un resultado.

Interactuar con otras aplicaciones

Con los intents se puede:

- Mover de una pantalla a otra dentro de nuestra app.
- Mandar al usuario a la pantalla de otra aplicación para realizar una acción.
- Cualquier pantalla a la que invoquemos puede devolver un resultado.
- Podemos permitir que otras aplicaciones lancen una actividad de la nuestra.

Contenidos I

Componentes Gráficos y eventos

Para que una aplicación sea funcional debe responder a los eventos del usuario. Los más típicos son:

- *onClickListener:* Para botones, texto, en general, para todos los Views.

Componentes Gráficos y eventos

Para que una aplicación sea funcional debe responder a los eventos del usuario. Los más típicos son:

- *onClickListener:* Para botones, texto, en general, para todos los Views.
- *onKeyListener:* Para cajas de texto.

Componentes Gráficos y eventos

Para que una aplicación sea funcional debe responder a los eventos del usuario. Los más típicos son:

- *onClickListener:* Para botones, texto, en general, para todos los Views.
- *onKeyListener:* Para cajas de texto.
- *onCheckedChangeListener:* Para checkboxes.

Componentes Gráficos y eventos

Para que una aplicación sea funcional debe responder a los eventos del usuario. Los más típicos son:

- *onClickListener:* Para botones, texto, en general, para todos los Views.
- *onKeyListener:* Para cajas de texto.
- *onCheckedChangeListener:* Para checkboxes.
- *onItemClickListener:* Elementos de un ListView

Contenidos I

Menús

Veremos cómo crear menús con Action Bar, ya que los menús estándar están anticuados. Los tres tipos fundamentales de menús son:

- **Options menu y action bar:** Elementos principales del menú. Aquí deben ir acciones que tienen un impacto global en la app. (buscar, ajustes, crear mensaje etc)

Menús

Veremos cómo crear menús con Action Bar, ya que los menús estándar están anticuados. Los tres tipos fundamentales de menús son:

- **Options menu y action bar:** Elementos principales del menú. Aquí deben ir acciones que tienen un impacto global en la app. (buscar, ajustes, crear mensaje etc)
- **Context menu y contextual action mode:** Menús basados en el contexto. Son menús flotantes que aparecen al realizar un "long-click" en algún elemento. Las acciones se realizan sobre el contenido seleccionado. (Ej. Al seleccionar texto, copiar, pegar...)

Contenidos I

Adaptadores

- Puete entre un AdapterView y los datos de una Vista (View)

Adaptadores

- Puete entre un AdapterView y los datos de una Vista (View)
- Colecciones de datos que asignamos a una vista para que ésta los muestre. (Un ListView p.e)

Bibliografía recomendada I

Satya Komatineni.

Pro Android 4 - Libro en Amazon.

Apress 2012.

Developer.android.com

Documentación oficial de Android.

developer.android.com