

Curso de programación Android

T-Formación

Alejandro Alcalde

elbaultdelprogramador.com

6 de abril de 2014

Contenidos

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

- Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

- RelativeLayout

Fragments

Persistencia de datos

Contenidos

Preparando el entorno

Instalar eclipse y el plugin ADT

Descargar el SDK de Android y eclipse

Instalar el plugin ADT

Conceptos básicos Android

Hola Mundo

Crear el proyecto

Componentes del proyecto

Carpeta Res

Interfaz gráfica

Conceptos básicos

Layouts

LinearLayout

RelativeLayout

Fragments

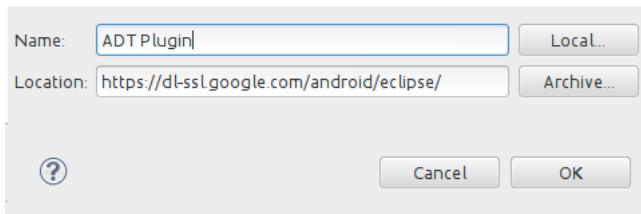
Persistencia de datos

Descargar el SDK de Android y eclipse

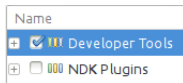
- ▶ Eclipse IDE for Java EE Developers - *enlace*
- ▶ Descargar el SDK - *enlace*
- ▶ Una vez descargado e instalado eclipse, instalar el plugin ADT.

Instalar el plugin ADT

- ▶ En eclipse ir a Help » Install New Software, Click en Add.
- ▶ En el cuadro de texto, poner un nombre (Ej. ADT Plugin) y en la url *<https://dl-ssl.google.com/android/eclipse/>*

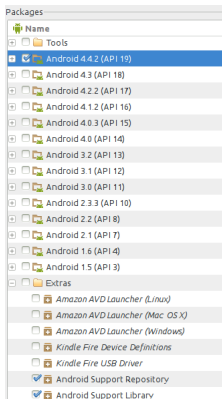


- ▶ A continuación, instalar las herramientas de desarrollador



Instalar el plugin ADT

- ▶ Una vez instalado el plugin y reiniciado eclipse, hay que decir dónde se encuentra el SDK.
- ▶ Window » Preferences » Android » SDK Location
- ▶ Instalar una imagen de Android y algunos paquetes extra. Window » Android SDK Manager



Contenidos

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

- RelativeLayout

Fragments

Persistencia de datos

Conceptos básicos Android

- ▶ **View:** Representa el componente básico en el que se apoyan todos los elementos que construyen una interfaz. Todos los elementos que generan interfaces heredan de la clase **View**

Conceptos básicos Android

- ▶ **View:** Representa el componente básico en el que se apoyan todos los elementos que construyen una interfaz. Todos los elementos que generan interfaces heredan de la clase **View**
- ▶ **Activity:** Encargada de mostrar la interfaz de usuario e interactuar con él. Responden a los eventos generados por el usuario (pulsar botones etc). Heredan de la clase **Activity**.

Conceptos básicos Android

- ▶ **View:** Representa el componente básico en el que se apoyan todos los elementos que construyen una interfaz. Todos los elementos que generan interfaces heredan de la clase **View**
- ▶ **Activity:** Encargada de mostrar la interfaz de usuario e interactuar con él. Responden a los eventos generados por el usuario (pulsar botones etc). Heredan de la clase **Activity**.
- ▶ **Services:** No tienen interfaz visual y se ejecutan en segundo plano, se encargan de realizar tareas que deben continuar ejecutandose cuando nuestra aplicación no está en primer plano. Todos los servicios extienden de la clase **Service**

Conceptos básicos Android

- ▶ **Content Provider:** Ponen un grupo de datos a disposición de distintas aplicaciones, extienden de la clase `ContentProvider` para implementar los métodos de la interfaz, pero para acceder a esta interfaz se ha de usar una clase llamada `ContentResolver`.

Conceptos básicos Android

- ▶ **Content Provider:** Ponen un grupo de datos a disposición de distintas aplicaciones, extienden de la clase `ContentProvider` para implementar los métodos de la interfaz, pero para acceder a esta interfaz se ha de usar una clase llamada `ContentResolver`.
- ▶ **BroadcastReceiver:** Simplemente reciben un mensaje y reaccionan ante él, extienden de la clase `BroadcastReceiver`, no tienen interfaz de usuario, pero pueden lanzar Actividades como respuesta a un evento o usar `NotificationManager` para informar al usuario.

Conceptos básicos Android

- ▶ **Content Provider:** Ponen un grupo de datos a disposición de distintas aplicaciones, extienden de la clase `ContentProvider` para implementar los métodos de la interfaz, pero para acceder a esta interfaz se ha de usar una clase llamada `ContentResolver`.
- ▶ **BroadcastReceiver:** Simplemente reciben un mensaje y reaccionan ante él, extienden de la clase `BroadcastReceiver`, no tienen interfaz de usuario, pero pueden lanzar Actividades como respuesta a un evento o usar `NotificationManager` para informar al usuario.
- ▶ **Intent:** Permite realizar la comunicación y transferencia de datos entre objetos de la clase `Activity` o `Service`. También permite iniciar otras `Activities` o lanzar otras aplicaciones.

Contenidos

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

- Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

- RelativeLayout

Fragments


Persistencia de datos

Crear el proyecto

Pasos a realizar

En eclipse, File » New » Android Application Project. Rellenamos la ventana con los siguientes datos:

New Android Application
Creates a new Android Application



Application Name:

Project Name:

Package Name:

Minimum Required SDK:

Target SDK:

Compile With:

Theme:

Todo siguiente hasta crear el proyecto.

Contenidos

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

- Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto**

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

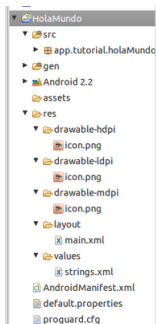
- RelativeLayout

Fragments

Persistencia de datos

Componentes del proyecto

Los proyectos de Android siguen una estructura fija de carpetas que debemos respetar. Podemos ver esta estructura con la vista Package Explorer que proporciona eclipse. (En Android Studio cambia). Gen » Generados por el compilador. Assets » Recursos externos que podamos necesitar, mp3, xml etc. Res » Recursos de la aplicación (Compilados)



Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con `src`. Se compilar y se generan referencias en la clase `R`, para acceder a ellos desde código. Están escritos en `XML`.

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores
- ▶ **drawable**: Ficheros bitmap(.png, .9.png, .jpg, .gif) o XML con contenidos que se dibujarán (fondos, botones etc).

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores
- ▶ **drawable**: Ficheros bitmap(.png, .9.png, .jpg, .gif) o XML con contenidos que se dibujarán (fondos, botones etc).
- ▶ **layout**: Definen la capa de interfaz de usuario

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores
- ▶ **drawable**: Ficheros bitmap(.png, .9.png, .jpg, .gif) o XML con contenidos que se dibujarán (fondos, botones etc).
- ▶ **layout**: Definen la capa de interfaz de usuario
- ▶ **menu**: Definición de los menús de la aplicación

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores
- ▶ **drawable**: Ficheros bitmap(.png, .9.png, .jpg, .gif) o XML con contenidos que se dibujarán (fondos, botones etc).
- ▶ **layout**: Definen la capa de interfaz de usuario
- ▶ **menu**: Definición de los menús de la aplicación
- ▶ **raw**: Binarios que no se pueden colocar en las otras carpetas.

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores
- ▶ **drawable**: Ficheros bitmap(.png, .9.png, .jpg, .gif) o XML con contenidos que se dibujarán (fondos, botones etc).
- ▶ **layout**: Definen la capa de interfaz de usuario
- ▶ **menu**: Definición de los menús de la aplicación
- ▶ **raw**: Binarios que no se pueden colocar en las otras carpetas.
- ▶ **values**: Definición de estilos, cadenas de texto para Localización etc.

Carpeta Res

Ésta es una de la carpeta que más se va a usar junto con **src**. Se compilar y se generan referencias en la clase **R**, para acceder a ellos desde código. Están escritos en **XML**.

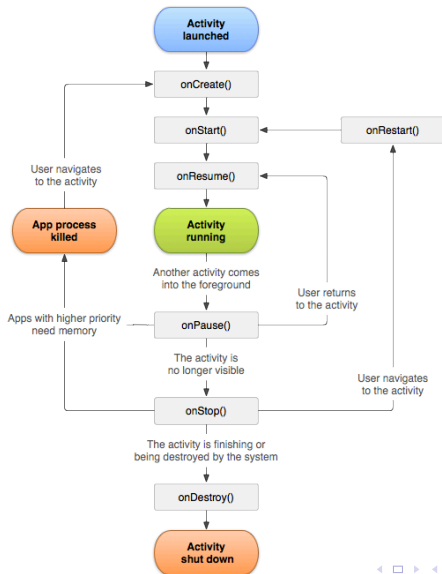
- ▶ **anim**: Definición de Animaciones.
- ▶ **color**: Definición de colores
- ▶ **drawable**: Ficheros bitmap(.png, .9.png, .jpg, .gif) o XML con contenidos que se dibujarán (fondos, botones etc).
- ▶ **layout**: Definen la capa de interfaz de usuario
- ▶ **menu**: Definición de los menús de la aplicación
- ▶ **raw**: Binarios que no se pueden colocar en las otras carpetas.
- ▶ **values**: Definición de estilos, cadenas de texto para Localización etc.
- ▶ **xml**: Ficheros XML que pueden ser accedidos en tiempo de ejecución

Hola Mundo

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    /**
     * Método encargado de "inflar" la actividad.
     * Inicializar cada componente de la actividad
     * con su correspondiente View.
     */
    setContentView(R.layout.activity_main);
}
```

Ciclo de vida de una Activity



Hola Mundo

`./res/layout/activity_main.xml`

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

`./res/values/strings.xml`

```
<resources>
    <string name="hello_world">Hello world!</string>
</resources>
```

Qué hemos visto

- ▶ Cómo preparar el entorno para desarrollar aplicaciones Android.
- ▶ Conceptos básicos Android.
- ▶ Creación de un proyecto Hola Mundo.

¿Y ahora qué?

- ▶ A partir de ahora, trabajaremos sobre ejemplos funcionales, deteniéndonos en las partes de código importantes para explicarlas.

Contenidos

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

- Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

- RelativeLayout

Fragments

Persistencia de datos

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:

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* raíz, 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, márgenes 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, márgenes 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, márgenes 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 una etiqueta definida en la clase R del sistema llamada 'list'.

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

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

- Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

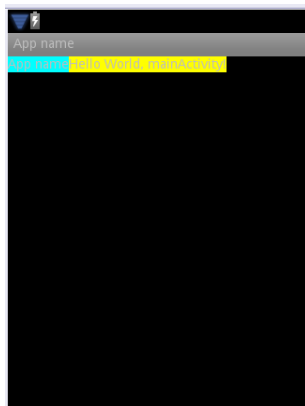
- RelativeLayout

Fragments

Persistencia de datos

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

Preparando el entorno

- Instalar eclipse y el plugin ADT

 - Descargar el SDK de Android y eclipse

 - Instalar el plugin ADT

- Conceptos básicos Android

Hola Mundo

- Crear el proyecto

- Componentes del proyecto

 - Carpeta Res

Interfaz gráfica

- Conceptos básicos

Layouts

- LinearLayout

- RelativeLayout

Fragments

Persistencia de datos

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.

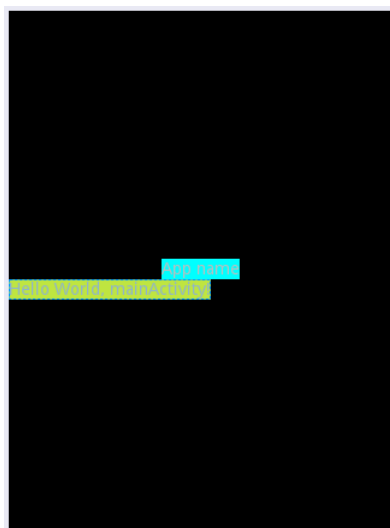


Figura: RelativeLayout

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>
```


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.

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.

Bibliografía recomendada I



Satya Komatineni.

Pro Android 4 - Libro en Amazon.

Apress 2012.



[Developer.android.com](http://developer.android.com)

Documentación oficial de Android.

developer.android.com