Curso de programación en Android

19/Junio/2012

Ramón Alcarria
Augusto Morales

Repaso conceptos básicos

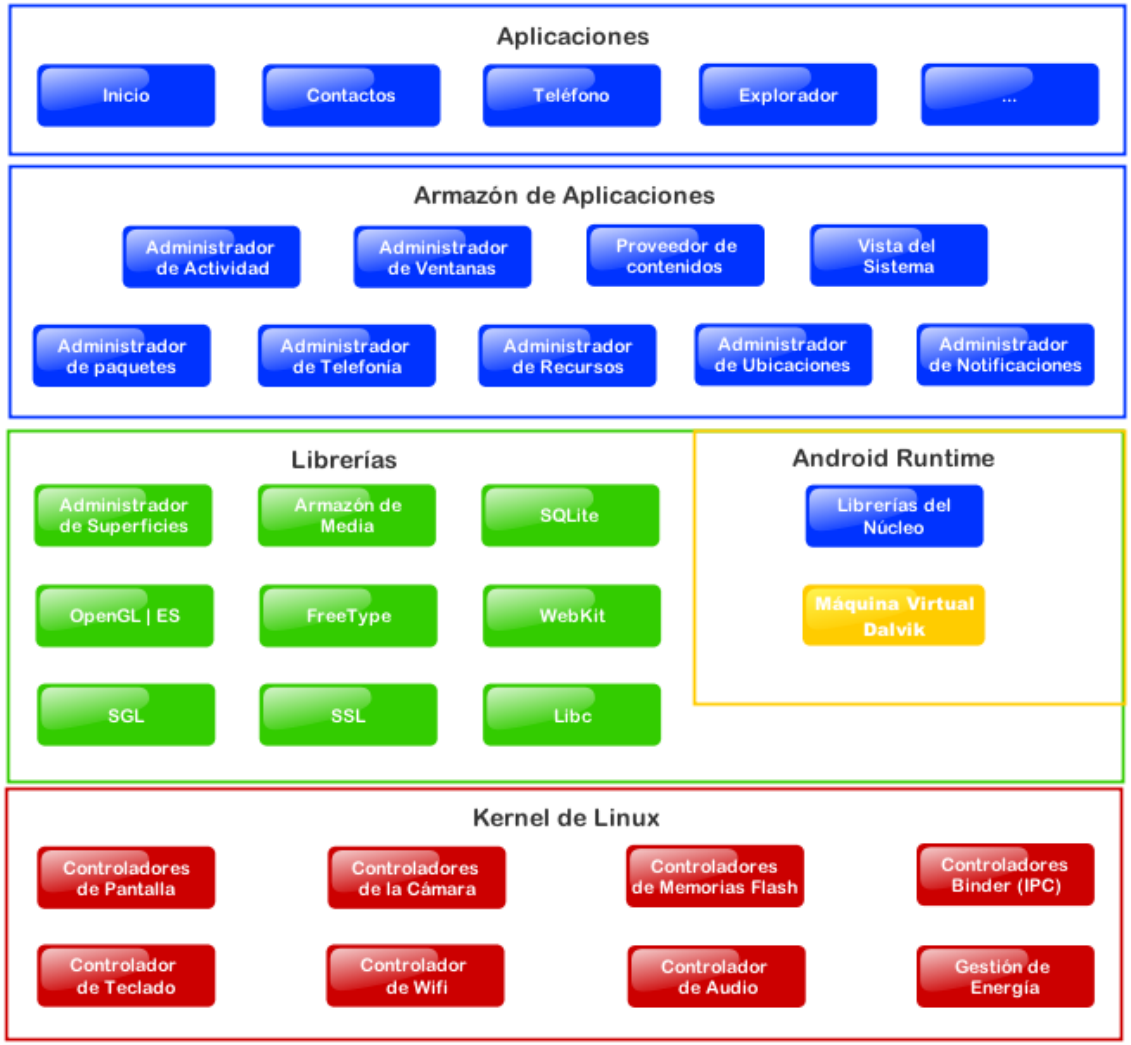
Arquitectura de Android

Elementos principales: Activity, Views, Intents, Services, Notifications, ContentProviders y Broadcast Receivers

Funciones Básicas

Trazas y *logcat*

Arquitectura de Android



Elementos principales

Activity: Utilizada para interactuar con el usuario. Representa una pantalla en una app Android. Puede tener multiples vistas.

onCreate() método principal donde se inicializa la actividad

setContentView(*view*) / findViewById(*viewId*)

Views: Objeto representable en pantalla (Boton, Cuadro de Texto, Lista, etc)

Intents: Representa un mensaje intercambiado entre bloques. Sirven para arrancar actividades y servicios. Ejemplo: *Open website intent*

Services: Se ejecutan en segundo plano de forma indefinida. Por ejemplo, el reproductor multimedia, bluetooth, gps, se accede mediante servicios.

Notifications: Mensaje que aparece en la barra de estado. Los usuarios pueden interactuar con este icono para recibir información. Hay notificaciones predefinidas (SMS, Email) pero también definidas por nuestras aplicaciones.

Elementos principales

Content Providers: Utilizados para intercambiar datos entre aplicaciones. Utilizados para persistencia y grandes volumenes de datos frente a los Intents.

Broadcast Receivers: Publican eventos y mensajes de forma asincrona. Nos suscribimos a ellos para conocer cuando llega un SMS, una llamada, se reinicia el sistema, el nivel de batería es bajo, etc. y podemos ejecutar código en consecuecia.

Elementos principales

Activities

- Pantallas... independientes entre sí
- Concepto principal

Services

- Funcionalidad de larga duración sin UI
- Services vs Threads → Services funcionan con app cerrada

Content Providers

- Acceso a datos entre aplicaciones
- Independientes de su persistencia (XML, SQLite, remoto...)

Broadcast Receivers

- Escucha mensajes de difusión
- Ejemplos: pantalla apagada, archivo descargados, batería baja...

Intents

- Mensajes asíncronos entre componentes
- Enlazan componentes en tiempo de ejecución
- Contienen:
  - Action
  - URI
  - Extras
- ¿Qué acciones hay?
  - Definidas en SDK
  - OpenIntents:

Estructura:

src: donde se encuentran el código fuente.

res: recursos (imágenes, archivos de configuración, propiedades)

drawable: imágenes

layout: archivos xml para la interfaz

values: valores estáticos

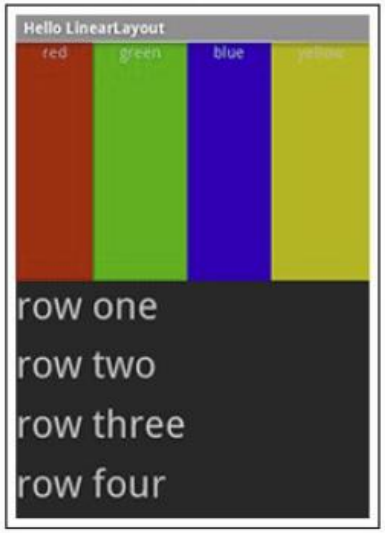
gen: Clase principal: R.java

Androidmanifest.xml

http://sudarmuthu.com/blog/the-structure-of-an-android-project

Layout

Linear Layout



Relative Layout

Table Layout

Grid View

Layout

Package Explorer -> **values** -> **strings.xml**

Layout editing with a mouse and can be checked in **GUI** Design view.

strings.xml Tab -> text editable XML file

modify the string. then the re-execution.

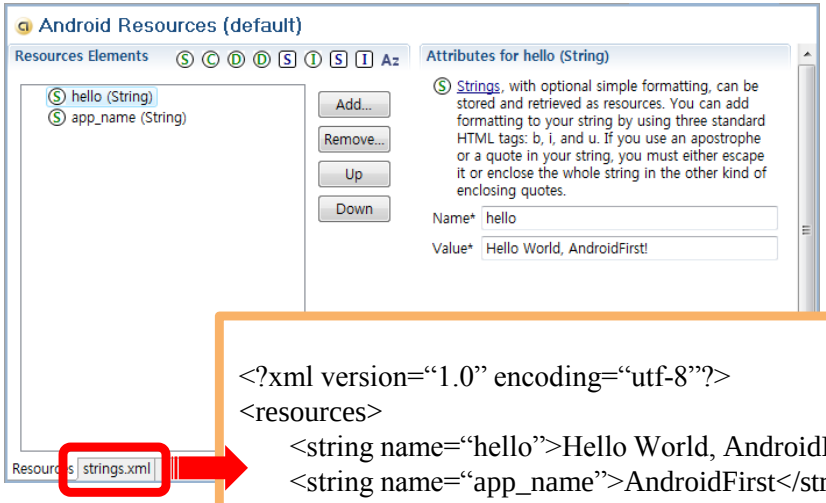
(Ctrl + S , Ctrl + F11)

R.java

Codes and definitions in the XML file that references the resource ID

Files that are automatically managed

MS Visual C++, resource.h




```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, AndroidFirst!</string>
    <string name="app_name">AndroidFirst</string>
</resources>
```

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the aapt tool from the resource data it found.
 * It should not be modified by hand.
 */

package exam.AndroidFirst;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x07f040001;
        public static final int hello=0x7f040000;
    }
}
```

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-
filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-
filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-
filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
        </provider>

        <uses-library />

    </application>

</manifest>
```

Working sets

Muy util cuando tenemos muchos proyectos en el "Package Explorer del Eclipse"

Generalmente tenemos dos opciones:

Cambiar el workspace a otra carpeta

Definir Working sets

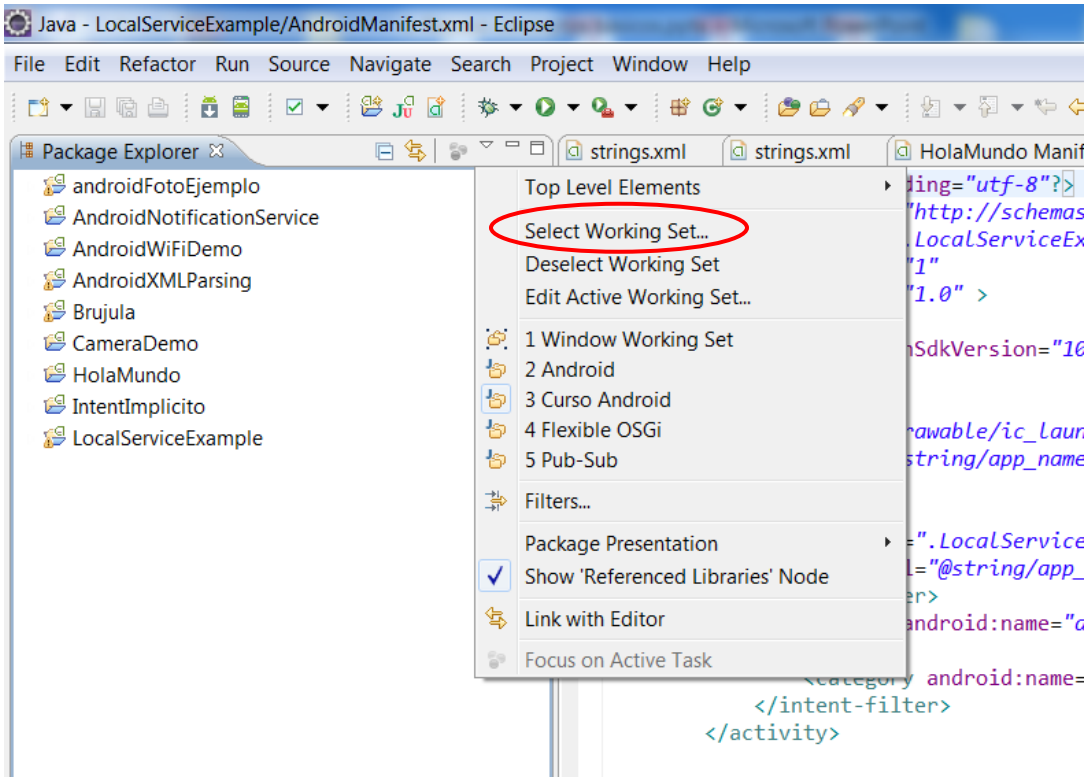
Que son?

Forman subconjuntos de archivos, clases, carpetas o proyectos. Permiten al desarrollador agrupar proyectos en un único grupo.

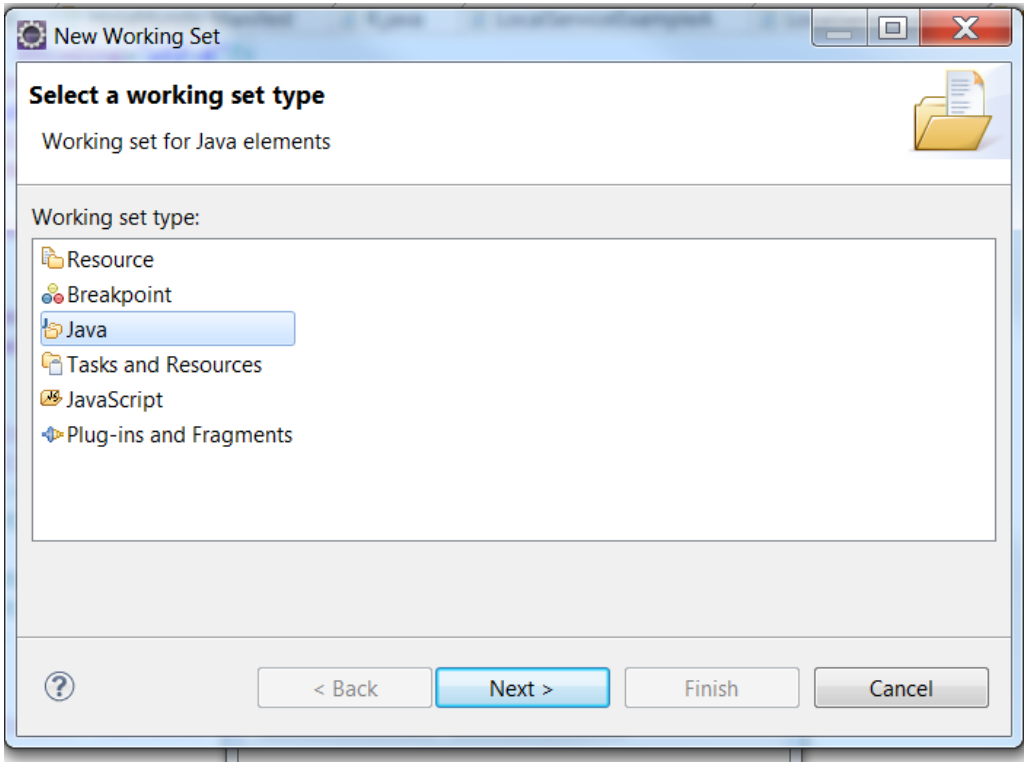
Para crear working sets:

Estando en el Package explorer

Working sets



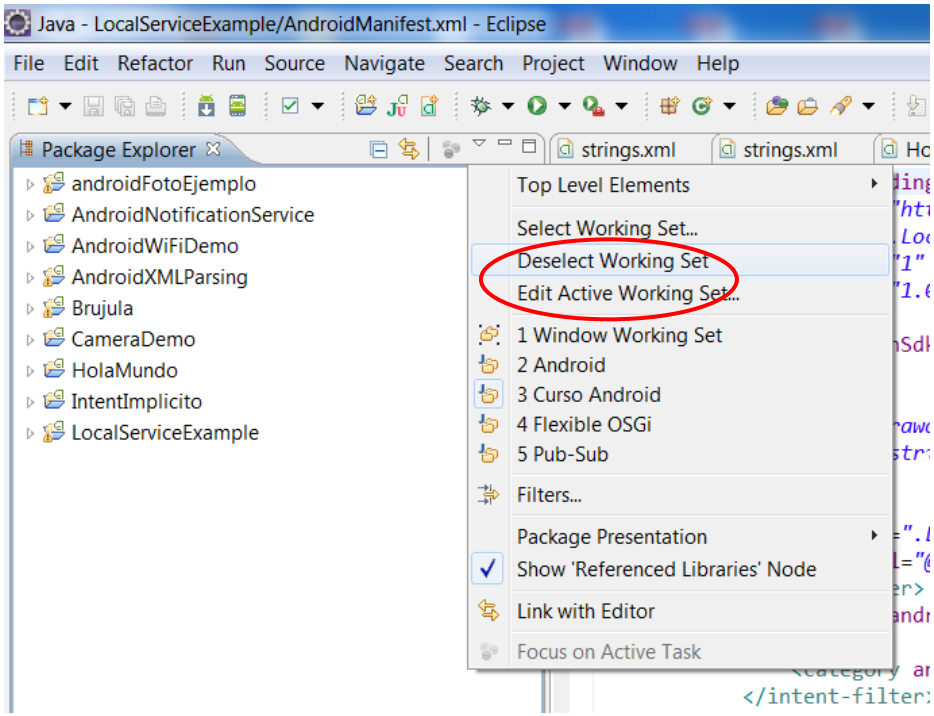
Working sets



Le ponemos nombre y seleccionamos los proyectos que queremos meter dentro

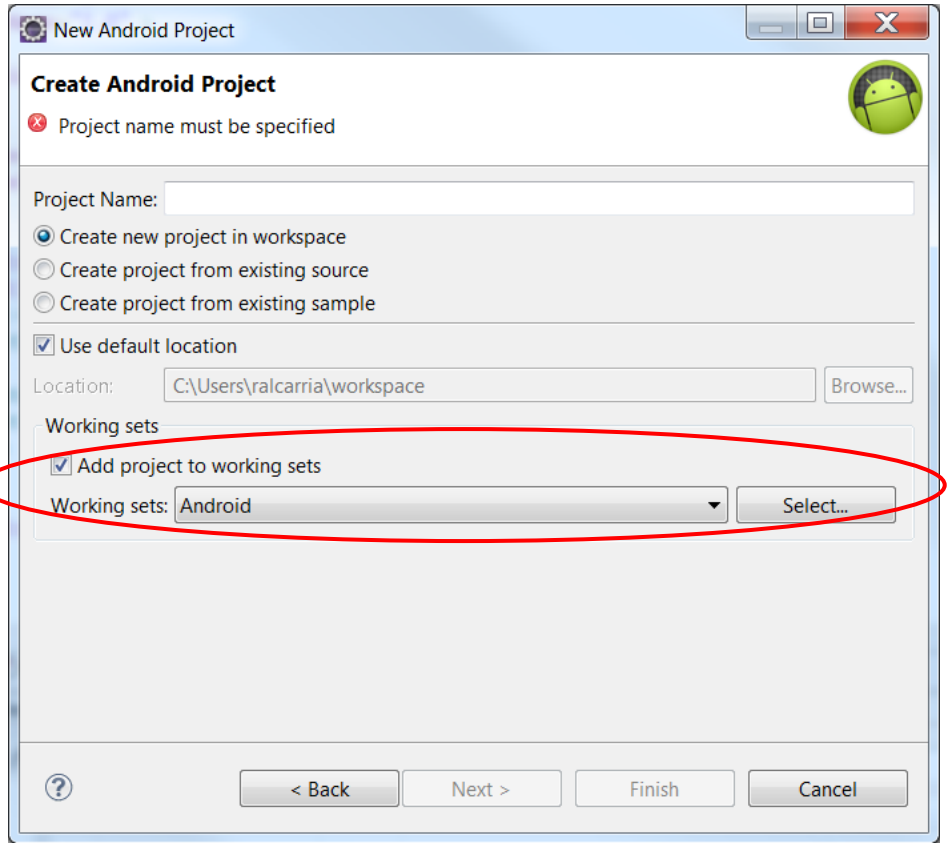
Working sets

Podemos deseleccionar el working set o editarlo apra quitar o añadir proyectos

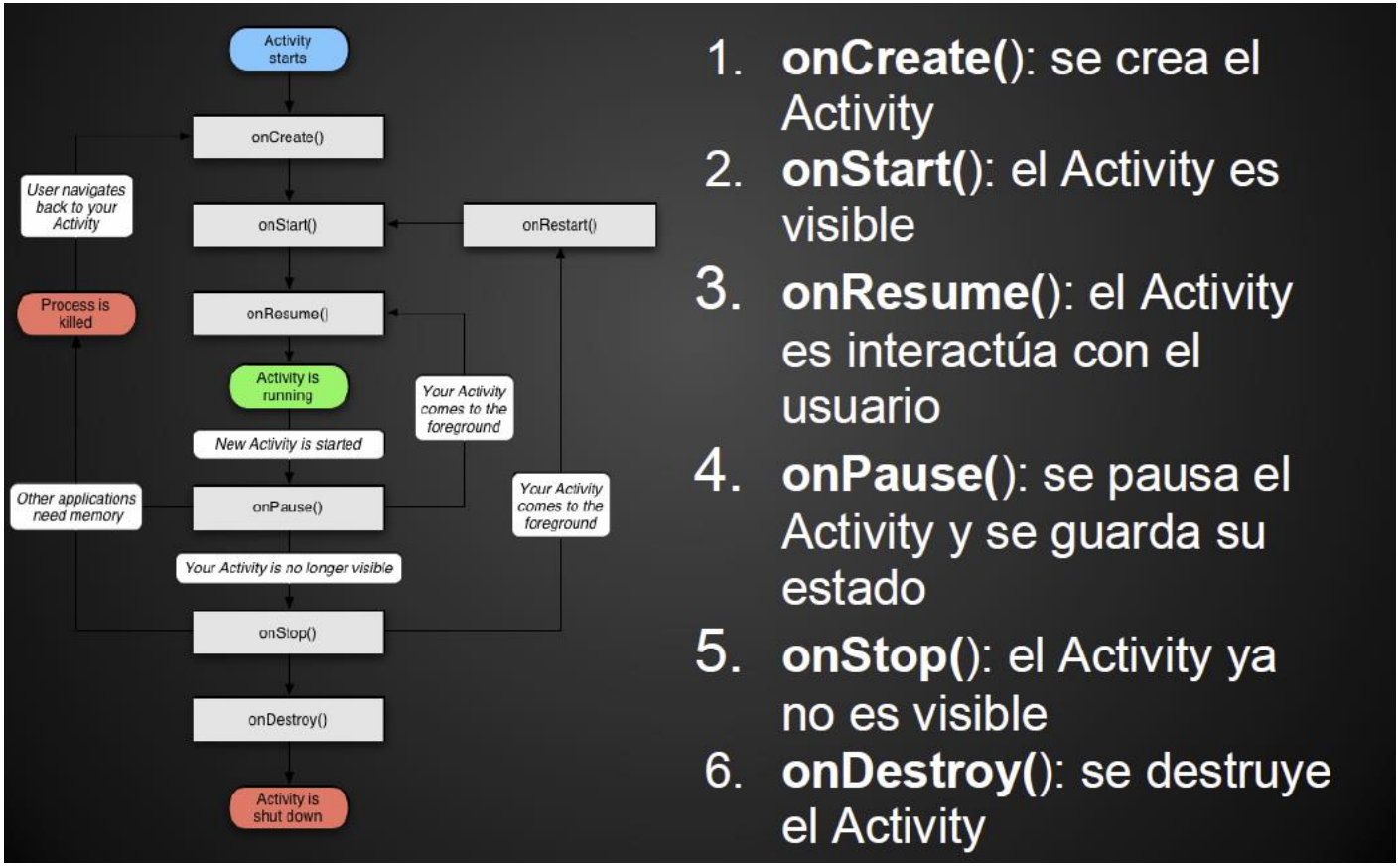


Working sets

Para añadir proyectos a los working set cuando los estás creando:



Ciclo de vida de un activity



Intents

Acciones estándar:

- ACTION_MAIN
- ACTION_VIEW
- ACTION_ATTACH_DATA
- ACTION_EDIT
- ACTION_PICK
- ACTION_CHOOSER
- ACTION_GET_CONTENT
- ACTION_DIAL
- ACTION_CALL
- ACTION_SEND
- ACTION_SENDTO
- ACTION_ANSWER
- ACTION_INSERT
- ACTION_DELETE
- ACTION_RUN
- ACTION_SYNC
- ACTION_PICK_ACTIVITY
- ACTION_SEARCH
- ACTION_WEB_SEARCH
- ACTION_FACTORY_TEST

Ejecución de Intents

```
Intent i = new Intent(this, IntentA.class);
startActivity(i);
```

Intents Implícitos

Estos Intents no especifican la clase Java a ser llamada, sino que especifican la acción a realizar y opcionalmente una URI a utilizar para esta acción.

Existen otras clases que serán los consumidores del Intent. El SO Android busca todos los componentes consumidores de un Intent específico.

- Si sólo se encuentra un componente Android arranca ese componente directamente

- Si encuentra más de uno nos aparecerá un menú de selección para que elijamos la aplicación que lo va a gestionar.

Por ejemplo el navegador Web se registra al Intent.ACTION_VIEW

Intents Implícitos

Estos Intents no especifican la clase Java a ser llamada, sino que especifican la acción a realizar y opcionalmente una URI a utilizar para esta acción.

Existen otras clases que serán los consumidores del Intent. El SO Android busca todos los componentes consumidores de un Intent específico.

- Si sólo se encuentra un componente Android arranca ese componente directamente

- Si encuentra más de uno nos aparecerá un menú de selección para que elijamos la aplicación que lo va a gestionar.

Por ejemplo el navegador Web se registra al Intent.ACTION_VIEW

```
Uri uri = Uri.parse( "http://iparty.aditel.org" );
startActivity( new Intent( Intent.ACTION_VIEW, uri ) );
```

Práctica con Intents implícitos

1. Creamos nuevo proyecto Android
2. Creamos el archivo intents.xml en “res/values”
3. Creamos un string array y le llamamos “intents”
4. Creamos los siguientes items dentro del array:

Abrir Navegador
Llamar a alguien
Marcar
Mostrar mapa
Buscar en Mapa
Tomar foto
Mostrar contactos

Práctica con Intents implícitos

intents.xml debería quedar así:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="intents">
        <item >Abrir Navegador</item>
        <item >Llamar a alguien</item>
        <item >Marcar</item>
        <item >Mostrar mapa</item>
        <item >Buscar en Mapa</item>
        <item >Tomar foto</item>
        <item >Mostrar contactos</item>
    </string-array>
</resources>
```

Práctica con Intents implícitos

En el main.xml creamos un Spinner y un Button de esta forma (Dentro del LinearLayout)

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="fill_horizontal"
    android:drawSelectorOnTop="true" >
  </Spinner>

<Button
    android:id="@+id/trigger"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Ir al Intent" >
</Button>
```

Práctica con Intents implícitos

Clase principal

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    spinner = (Spinner) findViewById(R.id.spinner);
    ArrayAdapter adapter = ArrayAdapter.createFromResource(this,
            R.array.intents, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner.setAdapter(adapter);
}
```

Práctica

Clase pri

```
public void onClick(View view) {
    int position = spinner.getSelectedItemPosition();
    Intent intent = null;
    switch (position) {
    case 0:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("http://www.marca.com"));
        break;
    case 1:
        intent = new Intent(Intent.ACTION_CALL,
                Uri.parse("tel:(+34)616479034"));
        break;
    case 2:
        intent = new Intent(Intent.ACTION_DIAL,
                Uri.parse("tel:(+34)616479034"));
        startActivity(intent);
        break;
    case 3:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("geo:50.123,7.1434?z=19"));
        break;
    case 4:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("geo:0,0?q=query"));
        break;
    case 5:
        intent = new Intent("android.media.action.IMAGE_CAPTURE");
        break;
    case 6:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("content://contacts/people/"));
        break;
    }
    if (intent != null) {
        startActivity(intent);
    }
}
```

GUI en Java o XML? Paso de uno a otro

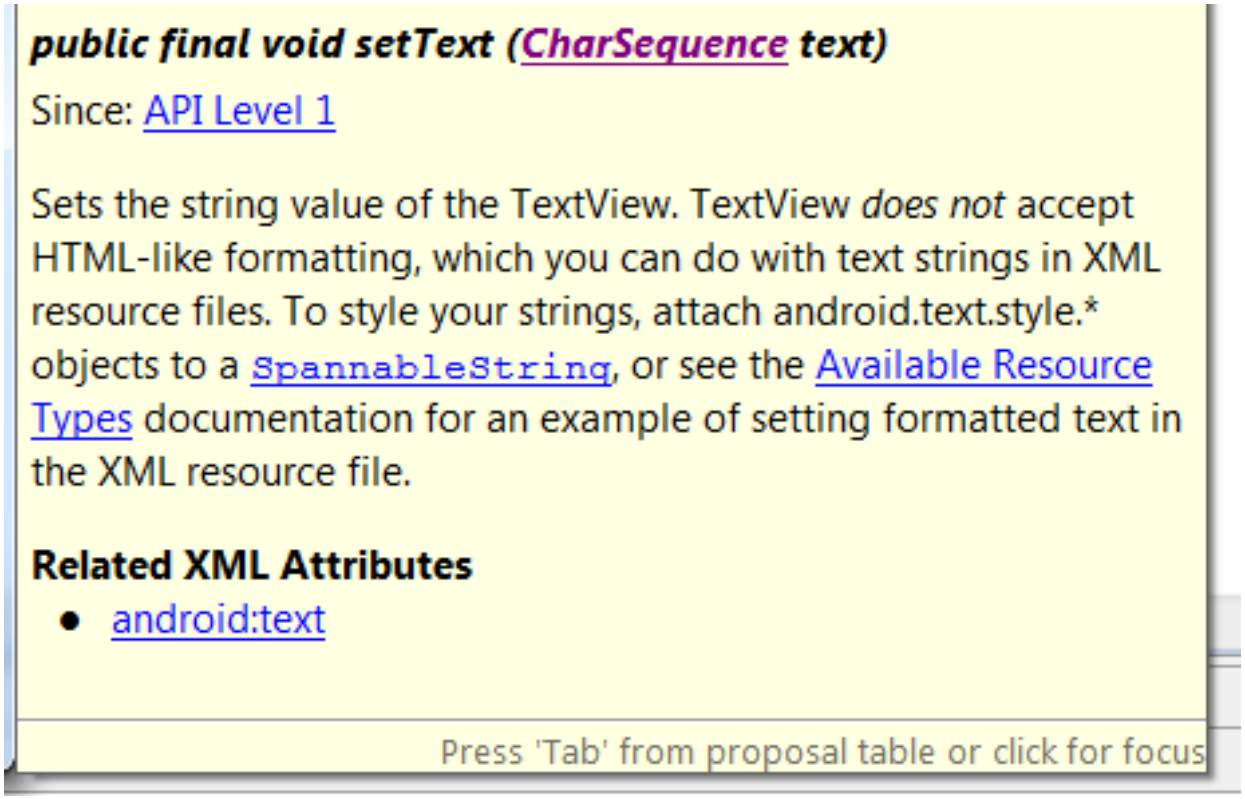
1. Partimos del proyecto Hola Mundo
2. setContentView(R.layout.main) lo sustituiremos por su equivalente en Java:
3. Creamos un LinearLayout: //LinearLayout linearLayout = new LinearLayout(this) *¿Por qué el this?*
4. Creamos un textView y le añadimos el texto a presentar por pantalla:
   ```
   TextView textView = new TextView(this);
   textView.setText(R.string.hello);
   ```
5. Añadimos el textView al linearLayout: linearLayout.addView(textView);
6. Añadimos el linearLayout a nuestra actividad con el método setContentView(linearLayout);

Probamos….

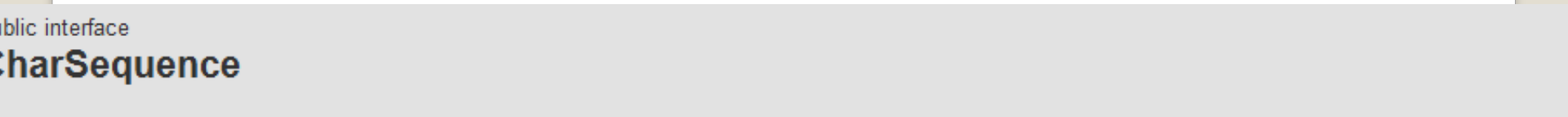
Como modificaríamos el texto del TextView sin cambiar el string de strings.xml??

Si miramos el método setText

GUI en Java o XML? Paso de uno a otro

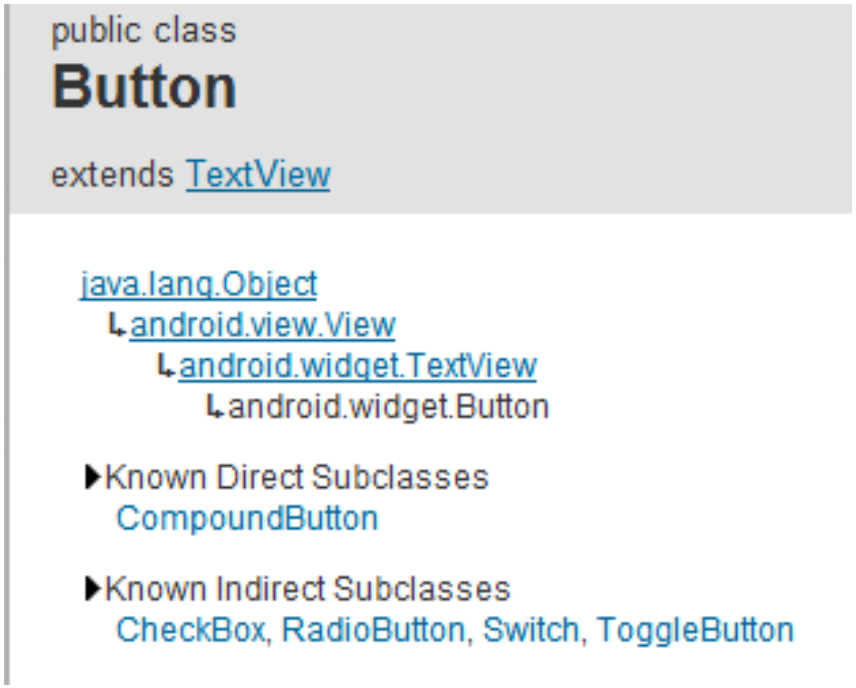


GUI en Java o XML? Paso de uno a otro



GUI en Java o XML? Paso de uno a otro

1. Ahora creamos un boton: Button button = new Button(this);
2. Y lo añadimos al linearLayout …..



GUI en Java o XML? Paso de uno a otro

Indicamos el texto que queremos en el boton: button.setText(“”);

Nos sale a la derecha, pero lo queremos abajo!

Tenemos que cambiar la orientación del layout de horizontal a vertical:

linearLayout.setOrientation(…)

http://developer.android.com/reference/android/widget/LinearLayout.html

Constants

public static final int **HORIZONTAL**

Constant Value: 0 (0x00000000)

public static final int **SHOW_DIVIDER_BEGINNING**

Show a divider at the beginning of the group.

Constant Value: 1 (0x00000001)

public static final int **SHOW_DIVIDER_END**

Show a divider at the end of the group.

Constant Value: 4 (0x00000004)

public static final int **SHOW_DIVIDER_MIDDLE**

Show dividers between each item in the group.

Constant Value: 2 (0x00000002)

public static final int **SHOW_DIVIDER_NONE**

Don't show any dividers.

Constant Value: 0 (0x00000000)

public static final int **VERTICAL**

Constant Value: 1 (0x00000001)

GUI en Java o XML? Paso de uno a otro

1. Para que el boton active algo al ser pulsado, tenemos que configurar un Listener. Para ello: button.setOnClickListener(this);
2. Tenemos que implementar la clase OnClickListener para poder hacer esto: *Repaso de Java*
3. Rellenamos los métodos obligatorios: *onClick*

Queremos que al pulsar el boton se cambie el texto desplegado

Probamos

GUI en Java o XML? Paso de uno a otro

Y ahora… como lo hacemos con xml?

Tenemos que crear un botón en main.xml con el siguiente código

```
<Button xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/button"
android:text=""
android:layout_width="fill_parent"
android:layout_height="wrap_content"/>
```

Para comunicarnos desde Java con un boton creado en XML necesitamos el método findViewById:

Button button = (Button) findViewById (R.id.button);

Soporte multilenguaje

Android tiene características de soporte de multilenguaje muy fáciles de utilizar.

Hacemos uso de la carpeta values para escribir Strings en varios idiomas

values-es, values-en, values-fr

No solo podemos cambiar los Strings sino también cualquier propiedad del layout, como el tipo de fuente, el tamaño, los colores, etc.

Ejercicio 1

- Aplicación que muestre el hola mundo en varios idiomas (dependiendo de la configuración….)
- Crear xml para cada lenguaje y cada tamaño de fuente

http://developer.android.com/guide/topics/resources/localization.html

Ejercicio 1

- Crear xml para cada lenguaje y cada tamaño de fuente

Crear carpetas: values-es values-ko
Replicar strings.xml dentro de esas carpetas

Crear en cada carpeta values-xx un archivo xml que se llame dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="tamano">50sp</dimen>
</resources>
```

En main.xml añadimos:

```
android:textSize="@dimen/tamano">
```

http://developer.android.com/guide/topics/resources/localization.html

Values para español

- Project_name/res/**values-es**/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="base_text">Español</string>
<string name="app_name">Multi-lenguaje</string>
</resources>
```

- Project_name/res/**values-es**/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="tamano">80sp</dimen>
</resources>
```

Values para ingles

- Project_name/res/**values-en**/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="base_text">English</string>
<string name="app_name">Multi-language</string>
</resources>
```

- Project_name/res/**values-en**/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="tamano">50sp</dimen>
</resources>
```

Values para koreano

- Project_name/res/**values-ko**/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="base_text">Korea</string>
<string name="app_name">멀티 언어</string>
</resources>
```

- Project_name/res/**values-ko**/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<dimen name="tamano">30sp</dimen>
</resources>
```

Practice - Layout

- Project_name/res/layout/main.xml

```
<TextView android:text="@string/base_text"
 android:id="@+id/TextView01"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textSize="@dimen/tamano">
</TextView>
```

Recursos

http://developer.android.com/reference/packages.html