Entorno de una App Android: Hello World!

Curso de Desarrollo en Android

GSyC/LibreSoft

Marzo de 2012

©2012 GSyC/LibreSoft
Algunos derechos reservados.
Este trabajo se distribuye bajo la licencia
Creative Commons Attribution Share-Alike
disponible en http://creativecommons.org/licenses/by-sa/3.0/es

Contenidos

Contenidos

Pasos para la creación y ejecución del proyecto Hello World

1) Crear un proyecto para la primera aplicación:

Se creará automáticamente un proyecto que no está vacío: imprime una cadena de texto cuando arranca la aplicación.

- Seleccionar "File" → "New" → "Project"
- Seleccionar el wizard "Android Project" en la carpeta "Android"
- Pulsar "Next". Rellenar los datos siguientes:
  - Project Name: Hello World
  - Build Target: Android 2.2
  - Application Name: Hello World
  - Package Name: com.clau.helloworld
  - Create Activity: HelloWorld
- Pulsar "Next", pulsar "Finish".

2) Crear la configuración de ejecución para este proyecto:

Se pueden crear varias configuraciones de ejecución, y configuraciones de depuración. Creamos una configuración de ejecución:

- Seleccionar "Run" → "Run Configurations"
- Pulsar con el botón derecho sobre "Android Application" y seleccionar "New". Rellenar los datos siguientes:
  - Name: Hello World
  - Project: Hello World
- Pulsar la pestaña "Target"
- Pulsar "Manager..."
- Pulsar "New" para crear un nuevo Android Virtual Device (AVD). Rellenar los siguientes datos
  - Name: MyAndroid
  - Target: Android 2.2 - API Level 8
- Pulsar "Create AVD" y esperar a que una ventana confirme la creación
- Cerrar la ventana "Android SDK"
- Esperar a que en el panel aparezca el nuevo AVD con nombre "MyAndroid"
- Activar la casilla del AVD "MyAndroid", pulsar "Apply", pulsar "Close".

3) Ejecutar la aplicación:

Se puede pulsar directamente el botón Run, o hacerlo a través de una de las configuraciones de ejecución preexistentes:

Seleccionar la configuración de ejecución "Hello World" recién creada en "Run" → "Run Configurations"

Pasos para la creación y ejecución

- Cuando desde Eclipse, con el plugin de Android instalado, se ejecuta una aplicación a través de una configuración de ejecución o de depuración, Eclipse hace lo siguiente:
  1. Compila la aplicación, generando un ejecutable para dalvik (`.dex`)
  2. Empaqueta el ejecutable y otros recursos externos en un paquete Android (`.apk`)
  3. Arranca el emulador (si no estaba ya arrancado)
  4. Instala el paquete de la aplicación (`.apk`) en el emulador
  5. Arranca la aplicación en el emulador
- Si se utiliza una configuración de depuración el depurador de Eclipse se conecta a la aplicación, pudiéndose depurar entonces desde la perspectiva de depuración de Eclipse

Carpetas y ficheros generados para un proyecto

Ficheros Android	Descripción
`AndroidManifest.xml`	Fichero que describe la aplicación: permisos, capacidades que exporta, cómo correrá.
`default.properties`	Fichero generado automáticamente. Define cómo construir la aplicación.
Carpeta `src`	Código fuente de la aplicación.
`src/com.clau.helloworld/HelloWorld.java`	Fichero con el código de la *Activity* `HelloWorld` de esta aplicación. Es el punto de entrada a la aplicación.
Carpeta `gen`	Carpeta en la que se almacenan ficheros relacionados con recursos autogenerados.
`gen/com.clau.helloworld/R.java`	Fichero fuente para manejar recursos desde la aplicación: no debe modificarse.
Carpeta `res`	Recursos de la aplicación: animaciones, imágenes, ficheros de *layout*, ficheros XML, strings, y otros ficheros.
`res/drawable/icon.png`	Icono de la aplicación que se muestra en el lanzador de aplicaciones del teléfono.
`res/layout/main.xml`	Fichero XML que define el *layout*.
`res/values/strings.xml`	Fichero XML con los *Strings* de la aplicación.

Explicación de la aplicación Hello World

Actividad HelloWorld (src/com.clau.helloworld/HelloWorld.java)

```
1  package com.clau.helloworld;

3  import android.app.Activity;
   import android.os.Bundle;
5
   public class HelloWorld extends Activity {
7    /** Called when the activity is first created. */
     @Override
9    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
11     setContentView(R.layout.main);
     }
13 }
```

- En la línea 6 (l-6) se extiende la clase Activity, clase utilizada en una aplicación para soportar tareas que requieren una interfaz gráfica
- El punto de entrada es el método `onCreate()`, que se redefine (l-9)
- En el código Java no vemos ni el *string* ni la View (el componente visual que se utiliza para mostrarlo en pantalla): se han definido como recursos externos
- En l-11 se *infla* la interfaz gráfica a partir de los recursos definidos en el fichero `res/layout/main.xml`

Recursos de la aplicación Hello World

- Definir en XML los aspectos visuales permite desacoplarlos de la lógica de la aplicación
- Los recursos están en la carpeta `res` del proyecto, con subcarpetas `drawable`, `layout` y `values`
- Desde el programa se puede acceder a los recursos externos a través de la variable **R**
- El fichero `res/layout/main.xml` define el *layout* de la vista de la actividad:

res/layout/main.xml

```
1  <?xml version=''1.0'' encoding=''utf-8''?>
   <LinearLayout xmlns:android=''http://schemas.android.com/apk/res/android''
3    android:orientation=''vertical''
     android:layout_width=''fill_parent''
5    android:layout_height=''fill_parent''>
     <TextView
7      android:layout''width=''fill_parent''
       android:layout_height=''wrap_content''
9      android:text=''@string/hello''
     />
11 </LinearLayout>
```

En la l-9 de `main.xml` se referencia un *string* definido en la l-3 de `res/values/strings.xml`:

res/values/strings.xml

```
1  <?xml version=''1.0'' encoding=''utf-8''?>
   <resources>
3    <string name=''hello''>Hello World, HelloWorld!</string>
     <string name=''app_name''>Hello World</string>
5  </resources>
```

Identificación de recursos de la aplicación Hello World

Para acceder a los elementos de la interfaz en el código se pueden añadir atributos `id` en XML (ver I-3):

res/layout/main.xml (modificado respecto al original)

```
...
<TextView
   android:id=''@+id/myTextView''
   android:layout_width=''fill_parent''
   android:layout_height=''wrap_content''
   android:text=''Hello World, HelloWorld''
/>
...
```

Luego, desde el código Java se podría utilizar el método `findViewById` para obtener una referencia al elemento cuyo id se ha definido en la I-3:

HelloWorld.java (modificado respecto al original)

```
...
TextView myTextView = (TextView)findViewById(R.id.myTextView);
...
```

HelloWorld sin la interfaz definida externamente en los recursos

Alternativamente se podrían crear los elementos gráficos de la actividad HelloWorld en Java (no recomendado):

HelloWorld.java (modificado respecto al original)

```
1  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
3    LinearLayout.LayoutParams lp;
     lp = new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT,
5                                       LayoutParams.FILL_PARENT);
     LinearLayout.LayoutParams textViewLP;
7    textViewLP = new LinearLayout.LayoutParams
                              (LayoutParams.FILL_PARENT,
9                              LayoutParams.WRAP_CONTENT);
     LinearLayout ll = new LinearLayout(this);
11   ll.setOrientation(LinearLayout.VERTICAL);
     TextView myTextView = new TextView(this);
13   myTextView.setText(``Hello World, HelloWorld'');
     ll.addView(myTextView, textViewLP);
15   this.addContentView(ll, lp);
   }
```

Contenidos

Las perspectivas de Eclipse para depuración

- Para depurar se selecciona una configuración preexistente en el menú Run → Debug Configurations. También puedes pulsar directamente el botón de depuración (dibujo de un *bug*)
- Eclipse tiene varias perspectivas, cada una con diferentes paneles. La perspectiva por omisión es la de Java.
- Arriba a la derecha aparecen botones para acceder a otras perspectivas, como la de depuración, en la que se pueden poner puntos de parada (*breakpoints*), ver la información de Log (LogCat) y depurar (ejecución paso a paso,...)
- Hay otra perspectiva: DDMS (Dalvik Debug Monitor Service) permite monitorizar y manipular el estado del emulador (procesos arrancados en el emulador p.ej.)

Ejercicio de depuración

- Añade en un fichero el siguiente método:

```
public void forceError() {
    if(true) {
        throw new Error("Error generado adrede");
    }
}
```

- Añade una llamada a `forceError()` en algún lugar del código (por ejemplo cuando se va a añadir un elemento en la lista, tras haber pulsado el botón del teléfono).
- Ejecuta la aplicación: se producirá una excepción en el lugar en el que hayas incluido esta llamada, y la aplicación en el emulador parará con un mensaje en pantalla.
- Depura la aplicación: para depurar se selecciona una configuración preexistente en el menú Run → Debug Configurations. También puedes pulsar directamente el botón de depuración (dibujo de un *bug*)
  Aparecerá un mensaje en Android para pasar a la perspectiva de depuración
- Desde la perspectiva de depuración se puede observar en el panel LogCat este mensaje (dentro del LogCat, mira el filtro que tiene un E en rojo):
  `AndroidRuntime error: java.lang.Error: Error generado adrede`
- Añade ahora un breakpoint en la línea desde la que llamas a `forceError()`: con el botón derecho pulsa en la columna de la izda. de la línea
- Vuelve a depurar. Cuando la ejecución pare en el breakpoint, ejecuta paso a paso hasta llegar a la línea que lanza la excepción.
- Puedes ver el contenido de la excepción en el panel de Variables de la perspectiva de depuración

Generación de mensajes en LogCat

- En el panel LogCat aparecen mensajes informativos. Hay varios filtros representados por círculos arriba a la derecha.
- Desde la aplicación se pueden generar mensajes de logging utilizando la clase `android.util.Log`:

Método	Propósito
`Log.e()`	Log errors
`Log.w()`	Log warnings
`Log.i()`	Log informational messages
`Log.d()`	Log Debug messages
`Log.v()`	Log Verbose messages

Ejercicio

- Añade la siguiente línea: `import android.util.Log;`
- Añade a la clase este String: `private static final String DEBUG_TAG= "MiTag";`
- Añade llamadas del tipo `Log.i(DEBUG_TAG, "Testing informational message 1 ");`
- Corre la aplicación y busca el el panel del LogCat (filtro I) tus mensajes

Contenidos

Bibliografía

- Capítulos 1 y 2 de *Professional Android Application Development*. Reto Meier. Ed. Wrox, 2009.
- Capítulos 1 y 2 de *Wireless Android Application Development*. Shane Conder, Lauren Darcey. Ed. Addison Wesley Professional, 2009.
- Documentación del Android SDK: en la carpeta `docs` del directorio del SDK, o en `http://developer.android.com/guide/index.html`
- Documentación sobre Android (tutoriales, vídeos,...): `http://developer.android.com`
- Hay instrucciones para la instalación del SDK de Android y de Eclipse en la página web de la asignatura