Ing. Laura Sandoval Montaño, Ing. Manuel Enrique Castañeda Castañeda
PAPIME 104911

Tablet Epad HD 10.2 Android 32 Gb doble núcleo.

Procesador de doble núcleo
memoria RAM de 512 MB
cámara de 5 mpx
salida Hdmi
WiFi y 3G
Gps Incluye antena
bocinas Estéreo
batería de ion litio de larga duración
Incluye gastos de envío a toda la república por estafeta entrega de 1 a 2 días hábiles

$ 2, 700

Para cada actividad, Android dispone de un hilo principal, de forma que cualquier código de la actividad se ejecuta en ese hilo.

El problema viene cuando la actividad trata de hacer acciones que tardan mucho tiempo en llevarse a cabo, lo cual si no se controla puede llevar a bloqueos de ese hilo principal, y por tanto, de la interfaz de usuario asociada a la actividad en cuestión.

Por tanto, todas las operaciones que sean potencialmente lentas (acceso a red, acceso a base de datos, acceso a archivo, etc.) deberemos ejecutarlas en segundo plano, utilizando la **concurrencia de hilos (threads).**

Sin embargo, en este caso, los hilos presentan el problema de que no pueden actualizar la interfaz de usuario desde segundo plano.

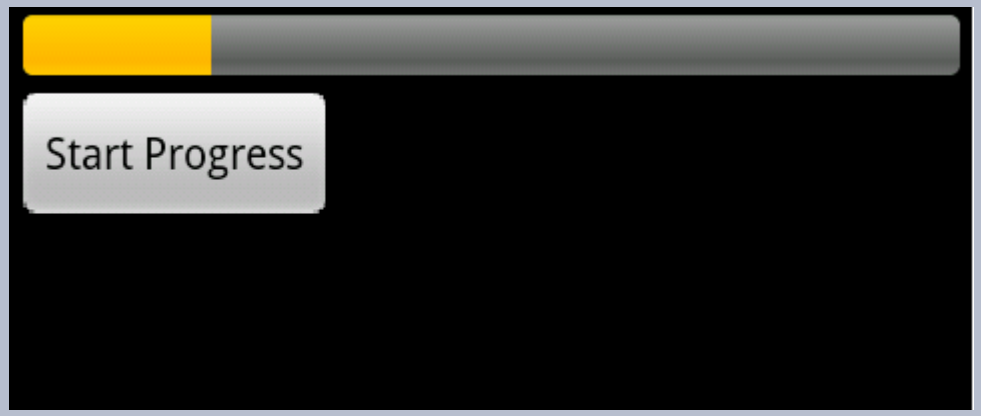
Para solventar esto, Android en sus orígenes proporcionó la clase *Handler*.

http://developer.android.com/reference/android/os/Handler.html

Un *handler* no es más que una clase que proporciona métodos para recibir mensajes, como *handleMessage()*, y objetos ***Runnable***, como *post()*. Supongamos un ejemplo como el siguiente, un botón que arranca una barra de progreso.

Android1.txt

Con esto, en cada ciclo del bucle, el *handler* actualiza la barra de progreso, y se simula su crecimiento de una forma bastante simple.



En las últimas versiones, Android presenta un mecanismo más cómodo: *AsyncTask*, que encapsula hilo y *Handler* en la misma clase.

Para ello, implementa el método *doInBackground()*, el cual define qué acción debe ser hecha en segundo plano, la cual de forma transparente se ejecuta en un hilo separado.

Una vez terminada esa acción, automáticamente se llama al método *onPostExecute()* para la actualización de la interfaz de usuario.

Supongamos ahora que quisiéramos leer una página web utilizando un objeto *AsyncTask*

Android2.txt

En definitiva, se pueden hacer cosas totalmente equivalentes usando estos dos mecanismos de concurrencia de Android.

Herramientas necesarias para preparar el entorno

1.- Instalar Jdk java, se trata de un conjunto de herramientas (programas y librerías) que permiten desarrollar programas en lenguaje Java (compilar, ejecutar, generar documentación, etc.).

http://www.oracle.com/technetwork/java/javase/downloads/index.html

2.- Descargar el SDK recomendado de Android, el SDK es un kit de desarrollo que contiene herramientas para desarrollar en Android, como por ejemplo la máquina virtual Dalvik.

http://developer.android.com/sdk/index.html

3.- Configurando SDK, abrimos la aplicación que se ha instalado en mis "Archivos de Programas->Android->Android-sdk-windows" y ejecutamos SDK Setup.exe.

4.- Una vez instalado el JDK, procedemos a descargar Eclipse. Este es un entorno de desarrollo integrado de código abierto multiplataforma. Para Eclipse 3.5 o más reciente, utilice la versión recomendada "Eclipse Classic ". Es solo descargar, descomprimir y utilizar.

http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/heliossr2

5.- Instalar componentes de Android para eclipse, en Eclipse Help->Install New Software.

6.- Falta solo un paso, que es configurar las preferencias del Android en el Eclipse, nos dirigimos a Window->Preferences.

http://www.htcmania.com/showthread.php?t=218891