Procesos largos

Créditos

- Tutorial https://developer.android.com/guide/components/processes-and-threads.html[1]

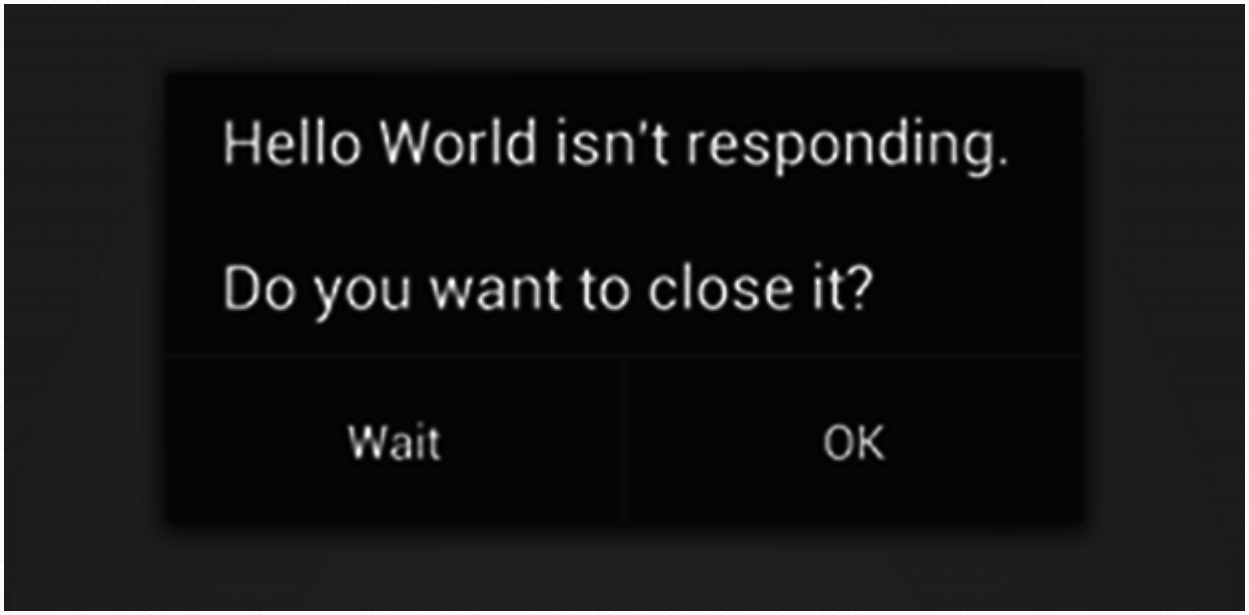
[1] Content is licensed under Creative Commons Attribution 2.5

Hilo principal

- Por default, todos los componentes de una aplicación corren en el mismo hilo (hilo principal – main thread).
- La interface de usuario (UI) también corre en el hilo principal.
- Es importante no bloquear el hilo principal.
- Ejemplos de procesos largos:
  - Cálculos numéricos.
  - Bajar un archivo remoto.
  - Accesar una página web externa.

ANR dialog

- ANR - Application Not Responding



Solución genérica

- Correr el proceso largo en otro hilo.

Primer intento

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork("http://example.com/image.png");
            mImageView.setImageBitmap(b);
        }
    }).start();
}
```

- Problema: accesar elementos de la UI fuera del hilo principal genera comportamiento indefinido e inesperado.

Soluciones específicas

- *View.post*(*Runnable*);
- *View.postDelayed*(*Runnable*, *long*);
- Clase *AsyncTask*

View.post(Runnable)

```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            final Bitmap bitmap =
                    loadImageFromNetwork("http://example.com/image.png");
            mImageView.post(new Runnable() {
                public void run() {
                    mImageView.setImageBitmap(bitmap);
                }
            });
        }
    }).start();
}
```

AsyncTask

- Ejecuta el proceso largo y publica el resultado en el hilo principal.
- Lo mínimo es extender la clase *AsyncTask* e implementar *doInBackground*().
- Para actualizar la UI se implementa *onPostExecute*() que corre en el hilo principal y recibe los resultados de *doInBackground*().
- Para correr se crea un objeto de tipo *AsyncTask* y se llama a *execute*().

AsyncTask

```
public void onClick(View v) {
    new DownloadImageTask().execute("http://example.com/image.png");
}

private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    /** The system calls this to perform work in a worker thread and
      * delivers it the parameters given to AsyncTask.execute() */
    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }

    /** The system calls this to perform work in the UI thread and delivers
      * the result from doInBackground() */
    protected void onPostExecute(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
```

Explicación

- class Foo extends AsyncTask<Tipo1, Tipo2, Tipo3>
  - Tipo1 – tipo del argumento en *doInBackGround*().
  - Tipo2 – tipo del argumento para reportar el progreso, o *Void* si no se usa.
  - Tipo3 – tipo del objeto regresado por *doInBackground*() y que se pasa como argumento a *onPostExecute*(), o *Void* si no se usa.
- *doInBackground*() corre en otro hilo.
- *onPreExecute*(), *onPostExecute*(), y *onProgressUpdate*() corren en el hilo principal.

Explicación

- El valor regresado por *doInBackground*() se envía a *onPostExecute*().
- Se puede llamar a *publishProgress*() en cualquier momento en *doInBackground*() para ejecutar a *onProgressUpdate*() en el hilo principal.
- Se puede cancelar la tarea en cualquier momento desde cualquier hilo.

Ejemplo

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

Ejemplo

- Para ejecutar la tarea anterior.

```
new DownloadFilesTask().execute(url1, url2, url3);
```

Ejemplo

- Calcular PI con 100 dígitos decimales usando la fórmula de John Machin.

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}$$

- Dónde el arcotangente se aproxima mediante una serie de Taylor:

$$\arctan x = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

Versiones

- Se programan 3 versiones:
  - Original – el proceso largo se ejecuta en el hilo principal.
  - Hilo – el proceso largo se ejecuta en otro hilo.
  - AsyncTask – el proceso largo se ejecuta en un objeto *AsyncTask*.

BigDecimal

- Maneja números con signo y precisión arbitraria.
- El valor se representa por un "valor sin escala" de precisión arbitraria y una "escala" con signo de 32 bits.

$$valor = unscaled \times 10^{-scale}$$

- La escala es el número de dígitos después del punto decimal.
- Hay métodos para hacer las operaciones básicas.
- Se puede definir la forma de redondear (i.e. hacia arriba, hacia abajo, etc.)

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tesi.pi1.MainActivity">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Número de dígitos:"/>

        <EditText
            android:id="@+id/numDigitos"
            android:text="100"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
```

Layout

```
        <Button
            android:text="Calcular PI"
            android:onClick="calculaPI"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="PI:"/>

        <EditText
            android:id="@+id/pi"
            android:minLines="10"
            android:gravity="top|left"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

    </LinearLayout>

</ScrollView>
```

Calcular PI

```
private BigDecimal computePI(int numDigitos)
{
    BigDecimal t1 = BigDecimal.ONE.divide(BigDecimal.valueOf(5),
                                          numDigitos, RoundingMode.HALF_EVEN);
    BigDecimal atan5 = arctan(t1, numDigitos);
    BigDecimal t2 = BigDecimal.ONE.divide(BigDecimal.valueOf(239),
                                          numDigitos, RoundingMode.HALF_EVEN);
    BigDecimal atan239 = arctan(t2, numDigitos);
    BigDecimal pi = atan5.multiply(BigDecimal.valueOf(4));
    pi = pi.subtract(atan239);
    pi = pi.multiply(BigDecimal.valueOf(4));

    return pi;
}
```

Calcular arctan(*x*)

```
private BigDecimal arctan(BigDecimal x, int numDigitos)
{
    BigDecimal arriba, term;
    BigDecimal result = BigDecimal.ZERO;
    int i = 0;
    double abajo;
    do {
        arriba = x.pow(2 * i + 1);
        abajo = 2 * i + 1;
        term = arriba.divide(BigDecimal.valueOf(abajo),
                    numDigitos, RoundingMode.HALF_EVEN);
        if ((i % 2) == 0) {
            result = result.add(term);
        }
        else {
            result = result.subtract(term);
        }
        i++;
    } while (term.compareTo(BigDecimal.ZERO) != 0);

    return result;
}
```

Versión original

- El proceso largo corre en el hilo principal.

```
// Callback del boton
public void calculaPI(View view)
{
    EditText editText = (EditText) findViewById(R.id.numDigitos);
    int numDigitos = 1;

    try {
        numDigitos = Integer.parseInt(campo.getText().toString());
    }
    catch (Exception e) {
        Toast.makeText(this, "Error. Se usará un lugar decimal",
                       Toast.LENGTH_LONG).show();
    }

    BigDecimal pi = computePI(numDigitos);
    editText = (EditText) findViewById(R.id.pi);
    editText.setText(pi.toString());
}
```

Versión con hilo

```
// Callback del boton
public void calculaPI(View view)
{
    EditText editText = (EditText) findViewById(R.id.numDigitos);
    int numDigitos = 1;

    try {
        numDigitos = Integer.parseInt(campo.getText().toString());
    }
    catch (Exception e) {
        Toast.makeText(this, "Error. Se usará un lugar decimal",
                       Toast.LENGTH_LONG).show();
    }
```

Versión con hilo

```
    final int d = numDigitos;
    dialog = ProgressDialog.show(this, "PI", "Trabajando...", true);
    EditText resultText = (EditText) findViewById(R.id.pi);
    new Thread(new Runnable() {
        public void run()
        {
            final BigDecimal pi = computePI(d);
            resultText.post(new Runnable() {
                public void run()
                {
                    resultText.setText(pi.toString());
                }
            });
        }
    }).start();
}
```

Versión con AsyncTask

```
private class ComputePITask extends AsyncTask<Integer, Integer, BigDecimal> {
    private ProgressDialog dialog;

    // Constructor
    public ComputePI(Context context)
    {
        dialog = new ProgressDialog(context);
    }

    // Corre en el hilo principal
    protected void onPreExecute()
    {
        dialog.setMessage("Trabajando...");
        dialog.setIndeterminate(true);
        dialog.show();
    }
```

Versión con AsyncTask

```
// Corre en otro hilo
protected BigDecimal doInBackground(Integer... digits)
{
    return computePI(digits[0]);
}

// Corre en el hilo principal
protected void onPostExecute(BigDecimal result)
{
    EditText editText = (EditText) findViewById(R.id.pi);
    editText.setText(result.toString());
    dialog.cancel();
}
```

- Nota: el método *computePI* va dentro de esta clase.

Versión con AsyncTask

- Callback del botón en la actividad principal.

```
// Callback del boton
public void calculaPI(View view)
{
    EditText editText = (EditText) findViewById(R.id.numDigitos);
    int numDigitos = 1;

    try {
        numDigitos = Integer.parseInt(editText.getText().toString());
    }
    catch (Exception e) {
        Toast.makeText(this, "Error. Se usará un lugar decimal",
                       Toast.LENGTH_LONG).show();
    }

    // Crear el async task
    new ComputePITask(this).execute(numDigitos);
}
```