PDM Avanzado AsyncTask

Sumario

Introdución

Máis información en: http://developer.android.com/reference/android/os/AsyncTask.html

A clase AsyncTask vainos permitir facer o mesmo que un fío de execución (visto anteriormente) coa diferenza que **dende un AsyncTask imos poder ter acceso ós compoñentes gráficos** do fío principal.

AsyncTask

Ao crear unha clase que derive da clase AsyncTask, teremos acceso aos seguintes métodos:

- onPreExecute(). Executarase antes do código principal da nosa tarefa.
- doInBackground(). Código principal da nosa tarefa. Aquí é onde irá o código que poñemos no método run cando utilizamos un Thread.Non temos acceso ós elementos gráficos
- onProgressUpdate(). Executarase cada vez que chamemos ó método publishProgress() dende o método doInBackground().
- onPostExecute(). Executarase cando finalice a nosa tarefa (cando remate o método doInBackground().
- onCancelled(). Executarase cando se cancele a execución da tarefa antes de súa finalización normal.

O que nos interesa é a relación entre **doInBackground** é **onProgressUpdate**.

Se poñemos dentro do código do método doInBackground a orde **publishProgress()**, automaticamente vai chamar ó método onProgressUpdate.

O bo de todo é que onProgressUpdate se executa no fío principal e polo tanto imos poder referenciar calquera elemento gráfico da interface, mentres que doInBackground se executa nun fío diferente.

O resto de métodos tamén se executan no fío principal e polo tanto podemos ter acceso a todos os elementos gráficos do fío principal.

Definición de parámetros en AsyncTask

O facer unha clase que derive de AsyncTask necesitamos definir tres parámetros:

```
private class MiñaTarefa extends AsyncTask<Params, Progress, Result>{

};
```

- **Params**: O tipo de dato que recibiremos como entrada no método doInBackground. Se non leva parámetros poñeremos Void (ningún). Podemos enviarlle datos cando dende a activity chamemos ó método execute da AsyncTask.

É dicir, cando eu cre un obxecto da clase MiñaTarefa e poña:

```
MiñaTarefa tarefa = new MiñaTarefa();
tarefa.execute(valor_enviar);
```

No método execute podo enviarlle valores en forma de parámetros. Estes valores chegarán ó método doInBackGround en forma de parámetros. Cabe sinalar que podemos enviar varios valores separados por comas ou ben en forma dun array de valores de tipo que sexa.

Se por exemplo, valor_enviar é un integer, teríamos que indicalo na clase MiñaTarefa:

```
private class MiñaTarefa extends AsyncTask<Integer, Progress, Result>{

};
```

No exemplo que imos desenvolver, non enviamos parámetros e por tanto quedaría como Void.

- **Progress**: O tipo de datos có que actualizaremos o progreso da tarefa. Cando chamamos o método publishProgress podemos enviarlle un parámetro e aquí indicamos o tipo de dato.

Num exemplo que imos facer enviaremos un contador que indicará por onde ter que debuxarse unha barra de progreso e polo tanto será un Integer. Como comentamos anteriormente ó chamar ó método publishProgress estamos chamando ó método onProgressUpdate que recibirá dito valor. Polo tanto teremos que definir un parámetro de tipo Integer nese método.

```
private class MiñaTarefa extends AsyncTask<Void, Integer, Result>{

    @Override
    protected Boolean doInBackground(Void... params) {
            publishProgress(valor_integer);
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
  int progreso = values[0].intValue();
    }


};
```

Nota: Os **puntos suspensivos ...** indican que se envía ou se pode enviar un array. Por iso cando o recibimos accedemos ó elemento 0 do array.

- **Result**: O tipo de dato que devolveremos cando finalice á nosa tarefa. Por exemplo, poderiamos indicar que cando finalice a tarefa esta enviará un Boolean. Ten que ser definido no método doInBackGround e o seu valor será recibido polo método onPostExecute, en forma de parámetro:

```
private class MiñaTarefa extends AsyncTask<Void, Integer, Boolean>{

  @Override
  protected Boolean doInBackground(Void... params) {

// CANDO FINALICE
return true;
  }
  @Override
  protected void onPostExecute(Boolean result) {

  }
};
```

Nota: Lembrar que indicamos tipos de datos: Void, Boolean, Integer.

Nota: Importar a clase AsyncTask premendo Ctrol+Shift+O.

Cancelando unha tarefa

Tanto dende a activity principal (a través do obxecto da clase MiñaTarefa) como dentro da propia AsyncTask podemos cancelar a tarefa.

Nese caso, dentro do método doInBackGround, o método isCancelled() devolverá true. Se cancelamos a tarefa, non se chamará ó método onPostExecute, **se non a onCancelled**.

Por exemplo:

```
miñatarefa.cancel(true);
```

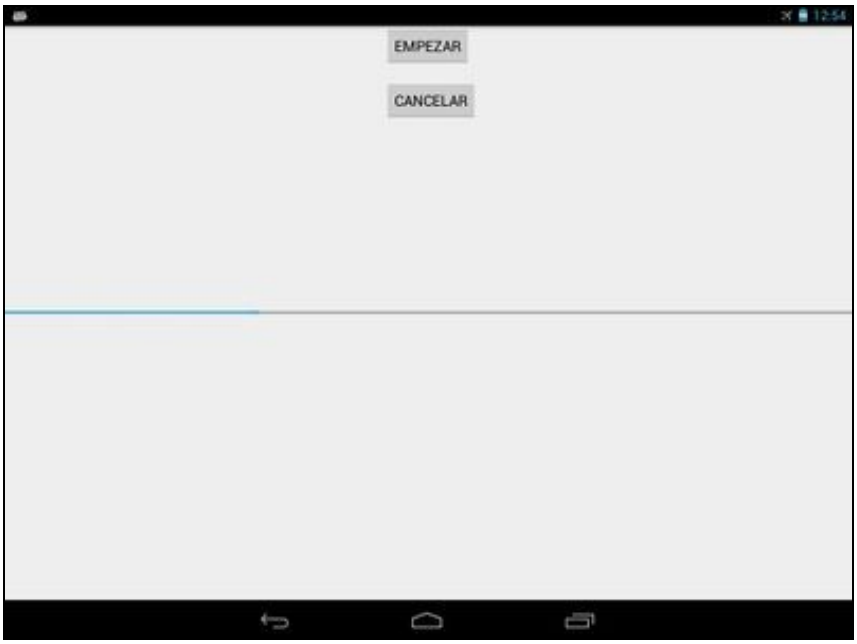
Nese caso, dentro do método doInBackGround, o método isCancelled() devolverá true. Se cancelamos a tarefa, non se chamará ao método onPostExecute, se non a onCancelled.

```
private class MiñaTarefa extends AsyncTask<Void, Integer, Boolean>{
       ............
       @Override
        protected Boolean doInBackground(Void... params) {

             if(isCancelled())
             return true;
       }

       @Override
protected void onCancelled() {
            Toast.makeText(getApplicationContext(), "Tarefa cancelada!",
                Toast.LENGTH_SHORT).show();
        }
};

       ............
}
```

Caso práctico

O obxectivo desta práctica e ver o funcionamento dun AsyncTask.

A práctica vai consistir en dous botóns cunha ProgressBar.



Ao premer sobre o botón 'Empezar' a barra de progreso se actualizará utilizando un AsyncTask. Ao premer sobre o botón 'Cancelar' o AsyncTask será cancelado e se informará ó usuario.

Creamos a activity

- Nome do proxecto: **UD3_02_AsyncTask**
- Nome da activity: **UD3_02_AsyncTask.java**

Código do layout xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".OutraActividade" >


    <Button
        android:id="@+id/UD3_02_btnEmpezarAsync"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="EMPEZAR" />

    <ProgressBar
        android:id="@+id/UD3_02_progressBarAsync"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true" />

    <Button
        android:id="@+id/UD3_02_btnCancelarAsync"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/UD3_02_btnEmpezarAsync"
        android:layout_below="@+id/UD3_02_btnEmpezarAsync"
        android:layout_marginTop="18dp"
        android:text="CANCELAR" />

</RelativeLayout>
```

Código da clase UD3_02_AsyncTask

Obxectivo: Actualizar unha barra de progreso utilizando un AsyncTask.

```
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.Toast;

public class UD3_02_AsyncTask extends Activity {

private static final int TEMPO_FINAL = 10;
private ProgressBar barraProgreso;
private MiñaTarefa miñaTarefa;


private class MiñaTarefa extends AsyncTask<Void, Integer, Boolean> {
```

```
@Override
protected Boolean doInBackground(Void... params) {
for (int i = 1; i <= TEMPO_FINAL; i++) {
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
publishProgress(i);

if (isCancelled())
break;
}
return true;
}

@Override
protected void onProgressUpdate(Integer... values) {
int progreso = values[0].intValue();
barraProgreso.setProgress(progreso);
}

@Override
protected void onPreExecute() {
barraProgreso.setProgress(0);
barraProgreso.setMax(TEMPO_FINAL);
}

@Override
protected void onPostExecute(Boolean result) {
if (result) {
Toast.makeText(getApplicationContext(), "Tarefa finalizada!",
Toast.LENGTH_SHORT).show();
}
}

@Override
protected void onCancelled() {
Toast.makeText(getApplicationContext(), "Tarefa cancelada!",
Toast.LENGTH_SHORT).show();
}
};

private void xestionarEventos(){

Button btnEmpezar = (Button)findViewById(R.id.UD3_02_btnEmpezarAsync);
btnEmpezar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub

if ((miñaTarefa==null) || (miñaTarefa.getStatus()==AsyncTask.Status.FINISHED)){
miñaTarefa = new MiñaTarefa();
miñaTarefa.execute();
}
else {
Toast.makeText(getApplicationContext(), "A tarefa non acabou!!!",
Toast.LENGTH_SHORT).show();
}
}
});

Button btnCancelar = (Button)findViewById(R.id.UD3_02_btnCancelarAsync);
btnCancelar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
if (miñaTarefa.getStatus()==AsyncTask.Status.RUNNING){
miñaTarefa.cancel(true);
```

```
}
}

});


}

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.activity_ud3_02__async_task);

barraProgreso = (ProgressBar) findViewById(R.id.UD3_02_progressBarAsync);

xestionarEventos();
}
}
```

- Liña 12: Definimos o 'tamaño' da barra de progreso.
- Liña 13: Referencia á barra de progreso.
- Liña 14: Definimos a tarefa (deriva de AsyncTask).
- Liñas 17-61: Definimos a clase que deriva de AsyncTask e que vai a actualizar a barra de progreso.

- Liña 69: Xestionamos o evento de Click sobre o botón de 'Comenzar'.
    - ◊ Liña 72: Para comenzar unha tarefa nova comprobamos que non fora instanciada ou ben rematara a anterior.
- Liña 87: Xestionamos o evento de Click sobre o botón de 'Cancelar'.
    - ◊ Liña 89-90: En caso de que a tarefa estea en execución a cancelamos.

Caso Práctico: Utilizando un ProgressDialog

Neste caso práctico anterior estamos a utilizar unha Activity para amosar un progreso.

Normalmente utilizaremos un ProgressDialog.

Un ProgressDialog. que como o seu propio nome di, é un tipo especial de diálogo no que se amosa unha barra de progreso (ou un texto). O rango da barra de progreso pode ir dende 0 a 10.000.

A idea é a de aproveitar o AsyncTask para amosar o diálogo actualizando o progreso no método onProgressUpdate, coma fixemos antes.

Podemos facelo de varias formas, unha delas é:

- Crear un obxecto da clase ProgressDialog:

```
public class MiñaActivity extends Activity {
.................
private ProgressDialog meuProgresoDialog;
...........
}
```

- Instanciamos dito obxecto no método onCreate:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_asysntask);


meuProgresoDialog = new ProgressDialog(this);
MiñaTarefa miñatarefa = new MiñaTarefa();
```

```
miñatarefa.execute();
}
```

- Modificamos a AsyncTask para inicializar o diálogo de progreso:

```
@Override
        protected void onPreExecute() {
             meuProgresoDialog.setTitle("TAREFA");
             meuProgresoDialog.setMessage("ACTUALIZANDO");
             meuProgresoDialog.setCancelable(false);
             meuProgresoDialog.setIndeterminate(true);
             meuProgresoDialog.show();

        }
```

Neste caso o diálogo de progreso non é 'cancelable', cando vai rematar tampouco o sabemos (non vai aparecer unha barra de progreso que indique inicio ? fin).

- Pechamos o ProgressDialog ó rematar a tarefa:

```
@Override
        protected void onPostExecute(Boolean result) {
            if(result)
            {
                Toast.makeText(ActivityAsysntask.this, "Tarefa finalizada!",
                    Toast.LENGTH_SHORT).show();
            }
            if (meuProgresoDialog!=null){
             meuProgresoDialog.dismiss();
            }
        }
```

- Por se acaso se cancela a tarefa tamén debemos pechar o diálogo:

```
@Override
protected void onCancelled() {
           Toast.makeText(ActivityAsysntask.this, "Tarefa cancelada!",
                Toast.LENGTH_SHORT).show();

           if (meuProgresoDialog!=null){
               meuProgresoDialog.dismiss();
           }
             }
```

Existen outros métodos que podemos usar:

Por exemplo, no noso caso sabemos cando remata a tarefa que será cando o contador chegue a 10. Podemos modificar o diálogo indicándoo:

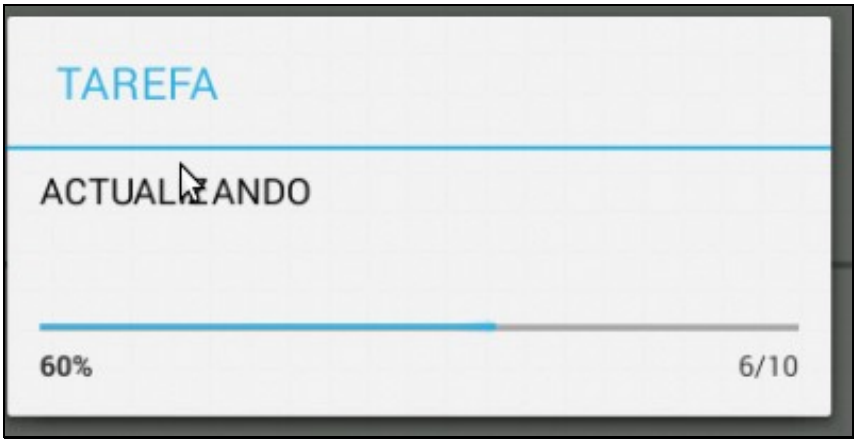
```
meuProgresoDialog.setCancelable(false);
meuProgresoDialog.setIndeterminate(false);
meuProgresoDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
meuProgresoDialog.setMax(TEMPO_FINAL);
```

Neste caso teremos que modificar a barra de progreso no método onProgressUpdate:

```
@Override
        protected void onProgressUpdate(Integer... values) {
            int progreso = values[0].intValue();
```

```
            meuProgresoDialog.setProgress(progreso);
        }
```

Obtendo este resultado:



Creamos a activity

- Nome do proxecto: **UD3_03_AsyncTask**
- Nome da clase: **UD3_03_AsyncTask.java**

- O código do layout é o mesmo que no exercicio anterior.

Código da clase UD3_03_AsyncTask
Obxectivo: Amosar o uso dun ProgressDialog xunto cun AsyncTask

```
import android.app.Activity;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class UD3_03_AsyncTask extends Activity {

private static final int TEMPO_FINAL = 10;
private MiñaTarefa miñaTarefa;

private ProgressDialog meuProgresoDialog;


private class MiñaTarefa extends AsyncTask<Void, Integer, Boolean> {

        @Override
        protected void onPreExecute() {


             meuProgresoDialog.setTitle("TAREFA");
             meuProgresoDialog.setMessage("ACTUALIZANDO");
             meuProgresoDialog.setCancelable(false);
             meuProgresoDialog.setIndeterminate(false);
             meuProgresoDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
             meuProgresoDialog.setMax(TEMPO_FINAL);
             meuProgresoDialog.show();

        }

@Override
protected Boolean doInBackground(Void... params) {
for (int i = 1; i <= TEMPO_FINAL; i++) {
try {
```

```
Thread.sleep(1000);
} catch (InterruptedException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
publishProgress(i);

if (isCancelled())
break;
}
return true;
}

@Override
protected void onProgressUpdate(Integer... values) {
int progreso = values[0].intValue();
            meuProgresoDialog.setProgress(progreso);
        }


@Override
protected void onPostExecute(Boolean result) {
if (result) {
Toast.makeText(getApplicationContext(), "Tarefa finalizada!",
Toast.LENGTH_SHORT).show();
}
            if (meuProgresoDialog!=null){
             meuProgresoDialog.dismiss();
            }
}

@Override
protected void onCancelled() {
Toast.makeText(getApplicationContext(), "Tarefa cancelada!",
Toast.LENGTH_SHORT).show();

if (meuProgresoDialog!=null){
                meuProgresoDialog.dismiss();
        }
}
};

private void xestionarEventos(){

Button btnEmpezar = (Button)findViewById(R.id.UD3_02_btnEmpezarAsync);
btnEmpezar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub

if ((miñaTarefa==null) || (miñaTarefa.getStatus()==AsyncTask.Status.FINISHED)){
miñaTarefa = new MiñaTarefa();
miñaTarefa.execute();
}
else {
Toast.makeText(getApplicationContext(), "A tarefa non acabou!!!",
Toast.LENGTH_SHORT).show();
}
}
});

Button btnCancelar = (Button)findViewById(R.id.UD3_02_btnCancelarAsync);
btnCancelar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub
if (miñaTarefa.getStatus()==AsyncTask.Status.RUNNING){
miñaTarefa.cancel(true);
}
}
```

```
        });


        }

        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ud3_03__async_task);

        meuProgresoDialog = new ProgressDialog(this);
        xestionarEventos();

        }
        }
```