Aplicaciones Telemáticas

Tema 2.4. Tareas en segundo plano

J. E. López Patiño, F. J. Martínez Zaldívar

ETSIT-UPV

Índice

Introducción

- Las actividades están previstas para finalizar la interacción con el usuario de manera rápida.
- Si una actividad tarda mucho en ejecutarse:
  - Puede no permitir la actualización simultánea de la interfaz gráfica (UI)
  - Puede que lo haga *torpemente*
  - Puede provocar la aparición de cierto mensaje de error
  - Ejemplo URL:
    `https://github.com/AATT-ETSIT/U2T4-Ej1-CuentaAtras.git`
- Solución: hacerlo en un hilo alternativo, en segundo plano (empleando la clase `Thread`). Problema
  - La UI sólo puede actualizarse desde el hilo en el que se creó
  - Soluciones:
    - Emplear métodos `post` sobre el control de la interfaz
    - Emplear el método `runOnUiThread`
    - `Handler`
    - `IntentService`
    - `AsyncTask`
    - ...
  - Mostraremos `AsyncTask`

Índice

Índice

Esquema de AsyncTask

```
class miTareaAsincrona extends AsyncTask < Parametros, Progreso, Resultado >
  { ... }
```

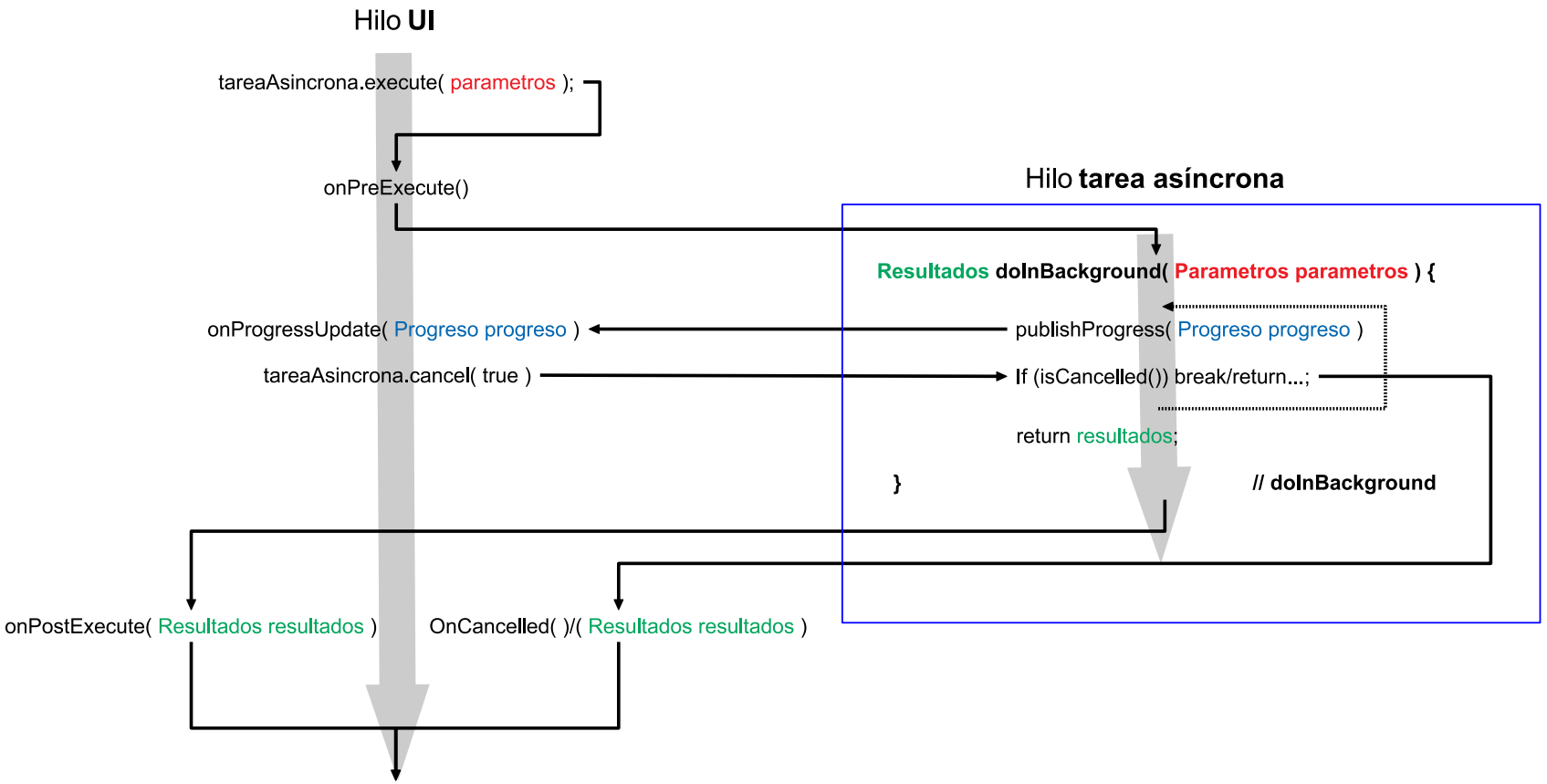
- Especificación de la computación a realizar en un hilo alternativo en segundo plano o *background*
- Su resultado se podrá *publicar* en el hilo de la UI
- `Asynctask` es una clase abstracta y genérica donde hay que especificar
  - Un método abstracto y protegido:
    - `Resultado doInBackground(Parametros ... parametros)`
  - Opcionalmente se pueden sobreescribr los métodos protegidos:
    - `void onPostExecute(Resultado resultado)`
    - `void onPreExecute()`
    - `void onProgressUpdate(Progreso ... progreso)`
    - `void onCancelled(Resultado resultado)` / `void onCancelled()`
  - Tres clases:
    - `Parametros`: clase de los objetos a proporiconar a `doInBackground`
    - `Progreso`: clase de los objetos a proporcionar a `onProgressupdate`
    - `Resultados`: clase del objeto entregado por `doInProgressUpdate` a `onPostExecute`

Esquema de AsyncTask (cont.)

- Actualización en la UI llamando a `publishProgress(...)` desde `doInBackground()`
- Se puede cancelar la ejecución:
  - `.cancel(true)`
  - Test lo más frecuente posible de `isCancelled()`: finalización forzada
  - Ejecución de `onCancelled()` en vez de `onPostExecute()` tras acabar `doInBackground()`
- Arranque:
  - `.execute( parametros )`
  - `.executeOnExecutor( AsyncTask.THREAD_POOL_EXECUTOR, parametros )`
- Un objeto `AsyncTask` sólo puede ejecutarse **una vez** (no se pueden *reutilizar*): si se desean más veces $\Rightarrow$ nuevos objetos
- Si hay varios objetos `AsyncTask`, sus `doInBackground()` respectivos:
  - Se serializan con `.execute( parametros )`
  - Se ejecutan en paralelo con `.executeOnExecutor( AsyncTask.THREAD_POOL_EXECUTOR, parametros )`

Gráficamente

Clases genéricas: `...extends AsyncTask < Parametros, Progreso, Resultado >`



Genéricamente

```
class MiTareaAsincrona extends AsyncTask < Parametros, Progreso, Resultados > {

   @Override protected void onPreExecute( ) { // En hilo de la UI
      ...
   }

   @Override protected Resultados doInBackground( Parametros ... parametros ) { // En hilo de segundo plano o background
      ...
         publishProgress( progreso1, progreso2, ... ); // Llamada indirecta a onProgressUpdate( progreso1, progreso2, ... )
      ...
      return resultados;
   }

   @Override protected void onProgressUpdate( Progreso ... progreso ) { // en hilo de la UI
      ...
   }

   @Override protected void onPostExecute( Resultados resultados ) { // en hilo de la UI
      ...
   }

   @Override protected void onCancelled( ) { // en hilo de la UI
                  ...
   }

   @Override protected void onCancelled( Resultados resultados ) { // en hilo de la UI
                  ...
   }
}
```

Índice

Ejemplo: cuenta atrás con tareas asíncronas

- URL: `https://github.com/AATT-ETSIT/U2T4-Ej2-TareasAsincronas.gi`
- `v1.0`: versión operativa, aunque con Problemas
- `v1.1`: funcionamiento correcto aunque con serialización en arranques sucesivos
- `v1.2`: funcionamiento correcto, esta vez con posibilidad de arranques paralelos (solo se puede cancelar el último)
- `v1.3`: idem que anteriores, pero usando alguna clase `Void`
- `v1.4`: Usuando *progressbars*