**THREAD**

**Android differenzia il thread principale dagli altri thread. Quali sono le operazioni non permesse nel thread principale (main) e quali quelle non permesse nei thread secondari (background)? Si motivi la risposta.**

Le app Android vengono eseguite per impostazione predefinita sul thread principale, chiamato anche thread dell'interfaccia utente. Gestisce tutti gli input dell'utente e quelli di output, quindi si evitano operazioni che richiedono tempo, a differenza di un Thread secondario, che include chiamate di rete, decodifica di bitmap o lettura e scrittura dal database.

**Un’app che utilizza una connessione Internet sfrutta la classe AsyncTask. Tale classe prevede i seguenti metodi: onPreExecute(), doInBackground(), onProgressUpdate(), onPostExecute(). Cosa viene tipicamente implementato in questi metodi? Nella risposta specificare per ogni metodo una possibile operazione da implementare nel metodo.**

Quando viene eseguita un'attività asincrona, l'attività passa attraverso 4 passaggi:

* ***onPreExecute (),*** richiamato sul thread dell'interfaccia utente prima dell'esecuzione dell'attività. Questo passaggio viene normalmente utilizzato per impostare l'attività, ad esempio mostrando una barra di avanzamento nell'interfaccia utente.
* ***doInBackground (Params ...),*** richiamato sul thread in background immediatamente dopo che ***onPreExecute()*** termina l'esecuzione. Questo passaggio viene utilizzato per eseguire calcoli in background che possono richiedere molto tempo. I parametri dell'attività asincrona vengono passati in questo passaggio. Il risultato del calcolo deve essere restituito da questo passaggio. Questo passaggio può anche utilizzare ***publishingProgress (Progress ...)*** per pubblicare una o più unità di progresso. Questi valori sono pubblicati sul thread dell'interfaccia utente, nel passaggio ***onProgressUpdate (Progress ...).***
* ***onProgressUpdate (Progress ...),*** richiamato sul thread dell'interfaccia utente dopo una chiamata a ***publishingProgress (Progress ...).*** I tempi dell'esecuzione non sono definiti. Questo metodo viene utilizzato per visualizzare qualsiasi forma di progresso nell'interfaccia utente mentre il calcolo in background è ancora in esecuzione. Ad esempio, può essere utilizzato per animare una barra di avanzamento.
* ***onPostExecute (Result),*** richiamato sul thread dell'interfaccia utente al termine del calcolo in background. Il risultato del calcolo in background viene passato a questo passaggio come parametro.

**Per quali situazione occorre utilizzare la classe AsyncTask? Si faccia un esempio.**

La classe ***AsyncTask<Params, Progress, Result>*** è una classe che consente di facilitare la gestione dei thread con Android nel caso in cui i thread in background devono comunicare con l’activity (UI) corrente.

I tipi generici che gestisce AsyncTask sono:

- ***Params***, ossia il tipo dei parametri con cui il thread in background deve effettuare la computazione. Uno o più oggetti Params verranno passati al metodo ***doInBackground()***;

- ***Progress***, ossia il tipo dei parametri utilizzato per comunicare il progresso all’activity corrente. Nel metodo ***doInBackground(),*** ogniqualvolta viene invocato ***publishProgress(Progress p),*** viene invocato il listener sull’activity ***onProgressUpdate(Progress p),*** per cui è possibile ora comunicare il progresso nell’interfaccia utente;

- ***Result***, ossia il tipo dei parametri che si vuole venga restituito da ***doInBackground()*** e, di conseguenza, passato al listener ***onPostExecute(Result result).***

Ad esempio, si vuole caricare un’immagine utilizzando un thread. Viene creata una classe che estende ***AsyncTask<Integer, Integer, Bitmap>***, dove: il primo Integer rappresenta l’id dell’immagine da caricare, il secondo Integer rappresenta il progresso da gestire con un valore intero, e Bitmap rappresenta il bitmap dell’immagine da ottenere.

**Per un oggetto AsyncTask, a cosa serve il metodo** **onProgressUpdate(Integer… progress)? Viene eseguito nel thread creato per il task asincrono o nel main thread? Perché?**

Il metodo ***onProgressUpdate (Progress ...)*** èrichiamato sul thread dell'interfaccia utente dopo una chiamata a ***publishingProgress(Progress ...).*** I tempi dell'esecuzione non sono definiti. Questo metodo viene utilizzato per visualizzare qualsiasi forma di progresso nell'interfaccia utente mentre il calcolo in background è ancora in esecuzione. Ad esempio, può essere utilizzato per animare una barra di avanzamento o mostrare i registri in un campo di testo.

Viene usato nel main Thread perché Android non permette ai thread in background di interagire con l’interfaccia utente, solo il main thread può farlo.

**Si completi il seguente codice assumendo di avere a disposizione la funzione “partialLoad()” che si occupa di caricare in ogni chiamata un 10% dell’immagine img (quindi dopo dieci chiamate a tale funzione img sarà completa). Si renda visibile la ProgressBar all’inizio del caricamento e invisibile alla fine. Si aggiorni la progress bar ad ogni 10% di caricamento e si mostri un Toast di avviso “Caricamento quasi completato” quando si raggiunge l’80% del caricamento. Si mostri l’immagine nell’imageView alla fine del caricamento.**

|  |
| --- |
| **public class ThreadAsyncTaskActivity extends Activity {**  **private ImageView imageView;**  **private ProgressBar progressBar;**  **@Override**  **protected void onCreate(Bundle savedInstanceState) {**  **super.onCreate(savedInstanceState);**  **setContentView(R.layout.main\_layout);**  **imageView = (ImageView)findViewById(R.id.imageView);**  **progressBar = (ProgressBar)findViewById(R.id.progressBar);**  **}**  **class LoadIconTask extends AsyncTask<Integer, Integer, Bitmap> {**  **private Integer index = 1;**  **@Override**  **protected void onPreExecute() {**  *progressBar.setVisibility(ProgressBar.VISIBLE);*  **}**  **@Override**  **protected Bitmap doInBackground(Integer... ids) {**  **Bitmap img =** *BitmapFactory.decodeResource(getResources(), ids[0]);*  *for(int i=1; i<=10; i++){*  *partialLoad();*  *publishProgress(i\*10);*  *}*  **return img;**  **}**  **@Override**  **protected void onProgressUpdate(Integer... values) {**  *if(values[0] == 80) Toast.makeText(getApplicationContext(), “Caricamento quasi completato”, Toast.LENGTH\_SHORT).show();*  *progressBar.setProgress(values[0]);*  **}**  **@Override**  **protected void onPostExecute(Bitmap result) {**  *progressBar.setVisibility(ProgressBar.INVISIBLE);*  *imageView.setImageBitmap(result);*  **}**  **}**  **}** |