Android: Multi-threading UI and Database (Room)

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on August 16, 2018 August 16, 2018 About Programming / Android

Each time I need an application, I encounter that those publish never fit my needs. However, I am lucky! I can make them myself.

Recently, I decided to create an exercise app that fit my purpose. Therefore, I began developing for Android again.

A few months ago, my Microsoft Phone's battery gave up and a massive deployment to production was coming which required me to be available; therefore, I purchased the first Android I could find. For those who wonder why I had a Microsoft Phone, I was developing apps using Universal Windows Platform using HTML5, JavaScript and the WinJS library. Pretty slick; however, there were bugs and poor designed features which made people drop the Windows Phone. No even Verizon Wireless support it. A shame

Anyways, you folk are not reading this post to listen to my poorly written stories but to find a solution to your problem such as dealing with messages as:

- Only the original thread that created a view hierarchy can touch its views.
- Cannot access database on the main thread since it may potentially lock the UI for a long period of time

These error messages are common when working with the UI and trying to do transactions with the database via Room.

In Short

For those who don't have the time or patience here is the code I use more often from all the other solutions:

```
final Handler handler = new Handler();
(new Thread(new Runnable() {
    @Override
    public void run() {
        // DO DATABASE TRANSACTION
        handler.post(new Runnable() {
    @Override
            public void run() {
                // DO UI STUFF HERE.
     }
        });
    }
})).start();
```

Other Options

So far, this is the easier and straight forwards solution that have being working for me.

I tried with runOnUiThread:

```
runOnUiThread(new Runnable() {
    @Override
    public void run() {
  // Run code here
    }
});
```

Also, I used AsyncTask:

```
new DatabaseAsync().execute();

private class DatabaseAsync extends AsyncTask<Void, Void, Void>{
    @Override
    protected void onPreExecute(){
    super.onPreExecute();
        // Pre-operation here.
    }

    @Override
    protected Void doInBackground(Void... voids){
        //Perform database operations here.
        return null;
    }

    @Override
    protected void onPostExecute(Void aVoid){
    super.onPostExecute(aVoid);
        // Do UI operations here
    }
}
```

And used combinations of Thread and runOnUiThread:

```
new Thread(new Runnable() {
    @Override
    public void run() {
        // Database operation here
        try {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    // UI operation here
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}).start();
```

I even took a look into RxJava which is a "Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences.".

Conclusion

There are many ways to tackle this issue as you can see. The trick here is to understand how android handles threading and UI threading; however, such topics are for another post.