Kostenloses eBook

LERNEN android-asynctask

Free unaffiliated eBook created from
Stack Overflow contributors.

#android-asynctask

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: android-asynctask

It is an unofficial and free android-asynctask ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official android-asynctask.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit android-asynctask

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick darüber, was Android-Asynktask ist und warum ein Entwickler sie verwenden möchte.

Es sollte auch alle großen Themen in der Android-Asynktask erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für Android-Asynktask neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

AsyncTask vom Konzept bis zur Implementierung

Konzept

AsyncTask ist eine Klasse, mit der Vorgänge im Hintergrund ausgeführt werden können. Die Ergebnisse werden im UI-Thread veröffentlicht. Der Hauptzweck besteht darin, den gesamten Boilerplate-Code zum Starten / Ausführen eines Threads zu eliminieren, indem die Handler und alles, was zum Manipulieren der Threads benötigt wird, entfernt werden. Der Zweck von AsyncTask besteht auch darin, kurze Vorgänge für einen Hintergrund-Thread (höchstens einige Sekunden) auszuführen, keine langzeitigen Vorgänge. Daher ist es wichtig, dass AsyncTask nicht mit einem generischen Threading-Framework verwechselt wird. Wenn langwierige Operationen erforderlich sind, wird das gleichzeitige Paket empfohlen.

Allgemeine Überlegungen

AsyncTask wird durch drei generische Typen definiert: Parameter, Fortschritt und Ergebnisse. Ab dem Moment, in dem es ausgeführt wird, durchläuft es 4 Schritte (Methoden). Der erste ist ***onPreExecute*** , wo jemand ein ***Ladedialogfeld*** definieren kann oder eine UI-Nachricht, die den Benutzer darüber informieren kann, dass die Ausführung gerade gestartet wird. Als nächstes ist ***doInBackground*** die Methode, die asynchron in einem anderen Thread als dem Ui-Thread ausgeführt wird. Die dritte Methode ist ***onProgressUpdate,*** die auch auf dem UI-Thread ausgeführt werden kann, der den Benutzer über den Status informieren kann. Die letzte aufgerufene Methode ist ***onPostExecute.*** Sie wird hauptsächlich für die Veröffentlichung der Ergebnisse verwendet.

Im Folgenden finden Sie ein Beispiel für die Verwendung einer AsyncTask, wobei eine Zeichenfolge zurückgegeben wird.
Beispiel 1

```
public class MainActivity extends AppCompatActivity {
```

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

        Button button = (FloatingActionButton) findViewById(R.id.btn);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                executeAsyncTaskOperation();
            }
        });
    }


      private void executeAsyncTaskOperation() {
        new CustomAsyncTask(this).execute();
    }

    private static class CustomAsyncTask extends AsyncTask<Void, Void, String> {

        private Context context;
        private ProgressDialog progressDialog;

        public CustomAsyncTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(context, "Please wait...", "Loading data from
web");
        }

        @Override
        protected String doInBackground(Void... params) {
            String object = null;
            try {
                Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground");
                Thread.sleep(500);
                //bject = "new object";
            } catch (Exception exc) {
                Log.e(CustomAsyncTask.class.getCanonicalName(), "exception");
                object = null;
            }
            return object;
        }

        @Override
        protected void onPostExecute(String s) {
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
            if (s != null) {
                Toast.makeText(context, "finished successfully!", Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(context, "finished unsuccessfully!", Toast.LENGTH_LONG).show();

            }
        }
    }
```

```
}
```

Beispiel 2

Hier ist die AsyncTask etwas anders, die Ausführungsmethode erhält eine Liste von Daten, die im Hintergrund analysiert werden sollen. Das Ergebnis der Rückgabe hängt von dieser Prüfung ab.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();

                executeAsyncTaskOperation();
            }
        });
    }


    private void executeAsyncTaskOperation() {
        Boolean[] bools = new Boolean[10];
        for (int k = 0; k < 10; k++) {
            if (k % 2 == 0) {
                bools[k] = true;
            } else {
                bools[k] = false;
            }
        }
        new CustomAsyncTask(this).execute(bools);
    }

    private static class CustomAsyncTask extends AsyncTask<Boolean, Void, Integer> {

        private Context context;
        private ProgressDialog progressDialog;

        public CustomAsyncTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(context, "Please wait...", "Loading data from
web");
        }

        @Override
        protected Integer doInBackground(Boolean... params) {
            int count = 0;
```

```
            try {
                Thread.sleep(1000);
                Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground");
                for (Boolean param : params) {
                    if (param) {
                        count++;
                    }
                }
            } catch (Exception exc) {
                Log.e(CustomAsyncTask.class.getCanonicalName(), "exception");
                count = 0;
            }
            return count;
        }

        @Override
        protected void onPostExecute(Integer s) {
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
            if (s != null && s > 0) {
                Toast.makeText(context, "finished loading: " + s + " tasks",
Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(context, "finished unsuccessfully!", Toast.LENGTH_LONG).show();

            }
        }
    }
}
```

Erste Schritte mit android-asynctask online lesen: https://riptutorial.com/de/android-asynctask/topic/8779/erste-schritte-mit-android-asynctask

Kapitel 2: Abbrechen eines AsyncTask

Einführung

Abbrechen eines AsyncTask

Examples

Abbrechen eines AsyncTask

Wenn im folgenden Beispiel jemand die Home-Taste drückt, während die Aufgabe ausgeführt wird, wird die Aufgabe abgebrochen. Bei diesem Abbruch sollte es unterbrochen werden, wenn es ausgeführt wird.

```
public class MainActivity extends AppCompatActivity {

   private static AtomicBoolean inWork;
   private CustomAsyncTask asyncTask;

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

       inWork = new AtomicBoolean(false);
       FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
       fab.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                       .setAction("Action", null).show();

               executeAsyncTaskOperation();
           }
       });
   }


   private void executeAsyncTaskOperation() {
       Boolean[] bools = new Boolean[10];
       for (int k = 0; k < 10; k++) {
           if (k % 2 == 0) {
               bools[k] = true;
           } else {
               bools[k] = false;
           }
       }
       asyncTask = new CustomAsyncTask(this);
       asyncTask.execute(bools);
   }

   //pressing the home button while the task is running will trigger the onStop being called.
   @Override
   protected void onStop() {
```

```
        if (asyncTask.getStatus() == AsyncTask.Status.RUNNING) {
            asyncTask.cancel(true);
        }
        super.onStop();
    }

    private static class CustomAsyncTask extends AsyncTask<Boolean, Void, Integer> {

        private Context context;
        private ProgressDialog progressDialog;

        public CustomAsyncTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onCancelled() {
            inWork.set(false);
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
                Log.d(CustomAsyncTask.class.getCanonicalName(), "progressdialog is
dismissed.");
            }
        }

        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(context, "Please wait...", "Loading data from
web");
        }

        @Override
        protected Integer doInBackground(Boolean... params) {
            int count = 0;
            inWork.set(true);
            try {
                Thread.sleep(1000);
                Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground");
                if (!isCancelled()) {
                    for (Boolean param : params) {
                        if (param) {

                            count++;
                        }
                    }
                } else {
                    Log.d(CustomAsyncTask.class.getCanonicalName(), "doInBackground is
cancelled.");
                }

            } catch (Exception exc) {
                Log.e(CustomAsyncTask.class.getCanonicalName(), "exception");
                count = 0;
            }
            return count;
        }

        @Override
        protected void onPostExecute(Integer s) {
            if (!isCancelled()) {
                inWork.set(false);
```

```
                if (progressDialog != null && progressDialog.isShowing()) {
                    progressDialog.dismiss();
                }
                if (s != null && s > 0) {
                    Toast.makeText(context, "finished loading: " + s + " tasks",
Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(context, "finished unsuccessfully!",
Toast.LENGTH_LONG).show();

                }
            } else {
                Log.d(CustomAsyncTask.class.getCanonicalName(), "onPostExecute is
cancelled.");
            }
        }
    }
}
```

Abbrechen eines AsyncTask online lesen: https://riptutorial.com/de/android-asynctask/topic/8783/abbrechen-eines-asynctask

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit android-asynctask	Andrei T, Community, rossettistone
2	Abbrechen eines AsyncTask	Andrei T