eBook Gratuit

APPRENEZ android-asynctask

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#android-asynctask

Table des matières

À propos

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

Chapitre 1: Démarrer avec android-asynctask

Remarques

Cette section fournit une vue d'ensemble de ce qu'est android-asynctask et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans android-asynctask, et établir un lien vers les sujets connexes. La documentation de android-asynctask étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

AsyncTask du concept à la mise en œuvre

Concept

AsyncTask est une classe qui permet d'exécuter des opérations en arrière-plan, les résultats étant publiés sur le thread d'interface utilisateur. L'objectif principal est d'éliminer tout le code passe-partout pour démarrer / exécuter un thread en éliminant les gestionnaires et tous les éléments nécessaires à la manipulation des threads. En outre, AsyncTask a pour but d'avoir des opérations de courte durée sur un thread d'arrière-plan (quelques secondes au plus), et non des opérations de longue durée. Par conséquent, il est important de ne pas confondre AsyncTask avec un framework de threading générique. Si vous devez effectuer des opérations à long terme, le package simultané est recommandé.

Considérations générales

AsyncTask est défini par trois types génériques: Params, Progress et Results. A partir du moment où il est exécuté, il passe par 4 étapes (méthodes). Le premier est ***onPreExecute*** , où quelqu'un peut définir une boîte de dialogue de chargement ou un message d'interface utilisateur capable de notifier à l'utilisateur que l'exécution est sur le point de démarrer. Ensuite, ***doInBackground*** est la méthode qui est exécutée de manière asynchrone sur un thread différent du thread Ui. La troisième méthode est ***onProgressUpdate*** qui peut également être exécutée sur le thread d'interface utilisateur qui peut informer l'utilisateur du statut. La dernière méthode appelée ***onPostExecute*** est principalement utilisée pour la publication des résultats.

Vous trouverez ci-dessous un exemple d'utilisation d'un AsyncTask, renvoyant une chaîne.
Exemple 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();

            }
        }
    }
}
```

Exemple 2

Ici, la AsyncTask est un peu différente, la méthode execute reçoit une liste de données à analyser en arrière-plan. Le résultat de retour dépend de cette vérification.

```
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();

            }
        }
    }
}
```

Lire Démarrer avec android-asynctask en ligne: https://riptutorial.com/fr/android-asynctask/topic/8779/demarrer-avec-android-asynctask

Chapitre 2: Annuler une AsyncTask

Introduction

Annuler une AsyncTask

Examples

Annuler une AsyncTask

Dans l'exemple suivant, si quelqu'un appuie sur le bouton d'accueil pendant l'exécution de la tâche, la tâche est annulée. Dans cette annulation particulière, il devrait interrompre si en cours d'exécution.

```
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.");
            }
        }
    }
}
```

Lire Annuler une AsyncTask en ligne: https://riptutorial.com/fr/android-asynctask/topic/8783/annuler-une-asynctask

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec android-asynctask	Andrei T, Community, rossettistone
2	Annuler une AsyncTask	Andrei T