EBook Gratuito

APPRENDIMENTO android-asynctask

Free unaffiliated eBook created from
Stack Overflow contributors.

#android-asynctask

Sommario

Di

Capitolo 1: Iniziare con android-asynctask

Osservazioni

Questa sezione fornisce una panoramica di cosa sia Android-asynctask e perché uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare qualsiasi argomento di grandi dimensioni all'interno di android-asynctask e collegarsi agli argomenti correlati. Poiché la documentazione di android-asynctask è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Examples

AsyncTask dal concetto all'implementazione

Concetto

AsyncTask è una classe che consente di eseguire operazioni in background, con i risultati pubblicati sul thread dell'interfaccia utente. Lo scopo principale è quello di eliminare tutto il codice boilerplate per l'avvio / esecuzione di un thread eliminando i gestori e tutto il materiale necessario per manipolare i thread. Inoltre, lo scopo di AsyncTask è di avere operazioni di breve durata su un thread in background (pochi secondi al massimo), non operazioni a lungo termine. Pertanto, è importante che AsyncTask non venga confuso con un framework di threading generico. Se è necessario eseguire operazioni a lungo termine, si consiglia di utilizzare il pacchetto simultaneo.

Considerazioni generali

AsyncTask è definito da tre tipi generici: Params, Progress e Results. Dal momento in cui viene eseguito, passa attraverso 4 passaggi (metodi). Il primo è ***onPreExecute*** , dove qualcuno può definire una finestra di caricamento o un messaggio di interfaccia utente che può avvisare l'utente che sta per essere ***avviata*** l'esecuzione. Quindi, ***doInBackground*** , che è il metodo che viene eseguito in modo asincrono su un thread diverso rispetto al thread Ui. Il terzo metodo è ***suProgressUpdate*** che può anche essere eseguito sul thread dell'interfaccia utente che può notificare all'utente lo stato. L'ultimo metodo chiamato ***onPostExecute*** è utilizzato principalmente per pubblicare i risultati.

Di seguito è riportato un esempio su come utilizzare un AsyncTask, restituendo una stringa.
Esempio 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();

            }
        }
    }
}
```

Esempio 2

Qui, AsyncTask è un po 'diverso, il metodo execute riceve una lista di dati da analizzare in

background. Il risultato della restituzione dipende da questo controllo.

```
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();

            }
        }
    }
}
```

Leggi Iniziare con android-asynctask online: https://riptutorial.com/it/android-asynctask/topic/8779/iniziare-con-android-asynctask

Capitolo 2: Annullare un AsyncTask

introduzione

Annullare un AsyncTask

Examples

Annullare un AsyncTask

Nell'esempio seguente, se qualcuno preme il pulsante home mentre l'attività è in esecuzione, l'attività viene annullata. In questa particolare cancellazione dovrebbe interrompersi se in esecuzione.

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

Leggi Annullare un AsyncTask online: https://riptutorial.com/it/android-asynctask/topic/8783/annullare-un-asynctask

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con android-asynctask	Andrei T, Community, rossettistone
2	Annullare un AsyncTask	Andrei T