



EBook Gratis

APRENDIZAJE android-asyncTask

Free unaffiliated eBook created from
Stack Overflow contributors.

#android-
asyncTask

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con android-asynctask.....	2
Observaciones.....	2
Examples.....	2
AsyncTask desde el concepto hasta la implementación.....	2
Capítulo 2: Cancelando una AsyncTask.....	6
Introducción.....	6
Examples.....	6
Cancelando una AsyncTask.....	6
Creditos.....	9

Acerca de

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capítulo 1: Empezando con android-asyncTask

Observaciones

Esta sección proporciona una descripción general de qué es android-asyncTask y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de android-asyncTask, y vincular a los temas relacionados. Dado que la Documentación para android-asyncTask es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

AsyncTask desde el concepto hasta la implementación

Concepto

AsyncTask es una clase que permite ejecutar operaciones en segundo plano, con los resultados publicados en el subproceso de la interfaz de usuario. El propósito principal es eliminar todo el código de repetición para iniciar / ejecutar un subproceso eliminando los controladores y todo lo necesario para manipular los subprocesos. Además, el propósito de AsyncTask es tener operaciones de corto tiempo en un hilo de fondo (unos segundos como máximo), no operaciones de largo tiempo. Por lo tanto, es importante que AsyncTask no se confunda con un marco de subprocesamiento genérico. Si es necesario realizar operaciones de larga duración, se recomienda el paquete concurrente.

Consideraciones Generales

AsyncTask se define por tres tipos genéricos: Parámetros, Progreso y Resultados. Desde el momento en que se ejecuta, pasa por 4 pasos (métodos). Primero está ***onPreExecute***, donde alguien puede definir un cuadro de diálogo de carga, o algún mensaje de UI que pueda notificar al usuario que la ejecución está a punto de comenzar. A continuación, ***doInBackground***, que es el método que se ejecuta de forma asíncrona en un subproceso diferente que el subproceso de la ***interfaz de usuario***. El tercer método es ***onProgressUpdate***, que también puede ejecutarse en el subproceso de la interfaz de usuario que puede notificar al usuario sobre el estado. El último método llamado ***onPostExecute*** se usa principalmente para publicar los resultados.

A continuación se muestra un ejemplo sobre cómo usar una AsyncTask, devolviendo una cadena.

Ejemplo 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);
            //object = "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();
        }
    }
}
}

```

Ejemplo 2

Aquí, la AsyncTask es un poco diferente, el método de ejecución recibe una lista de datos que se analizarán en segundo plano. El resultado de retorno depende de esta verificación.

```
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();
    }
}
}
}
}

```

Lea Empezando con android-asyncTask en línea: <https://riptutorial.com/es/android-asyncTask/topic/8779/empezando-con-android-asyncTask>

Capítulo 2: Cancelando una AsyncTask

Introducción

Cancelando una AsyncTask

Examples

Cancelando una AsyncTask

En el siguiente ejemplo, si alguien presiona el botón de inicio mientras la tarea se está ejecutando, se cancela la tarea. En este caso, la cancelación debería interrumpirse si se está ejecutando.

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

Lea Cancelando una AsyncTask en línea: <https://riptutorial.com/es/android-async-task/topic/8783/cancelando-una-async-task>

Creditos

S. No	Capítulos	Contributors
1	Empezando con android-asyncTask	Andrei T , Community , rossettistone
2	Cancelando una AsyncTask	Andrei T