LEARNING

android-asynctask

#android-
asynctask

Table of Contents

About

Chapter 1: Getting started with android-asynctask

Remarks

This section provides an overview of what android-asynctask is, and why a developer might want to use it.

It should also mention any large subjects within android-asynctask, and link out to the related topics. Since the Documentation for android-asynctask is new, you may need to create initial versions of those related topics.

Examples

AsyncTask from concept to implementation

Concept

AsyncTask is a class that allows running operations in the background, with the results being published on the UI thread. The main purpose is to eliminate all the boilerplate code for starting/running a thread by eliminating the handlers and all the stuff that are needed for manipulating the threads. Also, the purpose of AsyncTask is to have short-time operations on a background thread (a few seconds at most), not long-time operations. Therefore, it is important that AsyncTask not be confused with a generic threading framework. If one needs to do long-time operations then the concurrent package is recommended.

General considerations

AsyncTask is defined by three generic types: Params, Progress and Results. From the moment it is executed, it goes through 4 steps (methods). First is ***onPreExecute***, where someone can define a loading dialog, or some UI message that can notify the user the execution is about to start. Next, ***doInBackground*** which is the method that is run on asynchronously on a different thread than the Ui thread. The third method is ***onProgressUpdate*** which can also run on the UI thread that can notify the user about the status. The last method called it is ***onPostExecute*** it is mainly used for publishing the results.

Below is an example on how to use an AsyncTask, returning a string.
Example 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();

            }
        }
    }
}
```

Example 2

Here, the AsyncTask is a bit different, the execute method receive a list of data to be analyzed in the background. The return result depends on this check.

```
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();

            }
        }
    }
}
```

Read Getting started with android-asynctask online: https://riptutorial.com/android-asynctask/topic/8779/getting-started-with-android-asynctask

Chapter 2: Cancelling an AsyncTask

Introduction

Cancelling an AsyncTask

Examples

Cancelling an AsyncTask

In the following example if someone if someone presses the home button while the task is running, then the task is cancelled. In this particular cancelling it should interrupt if running.

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

Read Cancelling an AsyncTask online: https://riptutorial.com/android-asynctask/topic/8783/cancelling-an-asynctask

Credits

S. No	Chapters	Contributors
1	Getting started with android-asynctask	Andrei T, Community, rossettistone
2	Cancelling an AsyncTask	Andrei T