Android AsyncTask Example Tutorial

Today we will look into Android AsyncTask. We will develop an Android example application that performs an abstract AsyncTask in background.

Android AsyncTask

Android AsyncTask is an abstract class provided by Android, which gives us the liberty to perform heavy tasks in the background, and keep the UI thread light thus making the application more responsive.

Android application runs on a single thread when launched. Due to this single thread model tasks that take longer time to fetch the response can make the application non-responsive. To avoid this we use android AsyncTask to perform the heavy tasks in background on a dedicated thread and passing the results back to the UI thread. Hence use of AsyncTask in android application keeps the UI thread responsive at all times.

The basic methods used in an android AsyncTask class are defined below:

- **doInBackground()** : This method contains the code which needs to be executed in background. In this method we can send results multiple times to the UI thread by publishProgress() method. To notify that the background processing has been completed we just need to use the return statements
- **onPreExecute()** : This method contains the code which is executed before the background processing starts
- **onPostExecute()** : This method is called after doInBackground method completes processing. Result from doInBackground is passed to this method
- **onProgressUpdate()** : This method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update the UI thread

The three generic types used in an android AsyncTask class are given below :

- **Params** : The type of the parameters sent to the task upon execution
- **Progress** : The type of the progress units published during the background computation
- **Result** : The type of the result of the background computation

Android AsyncTask Example

To start an AsyncTask the following snippet must be present in the MainActivity class :

```
MyTask myTask = new MyTask();

myTask.execute();
```

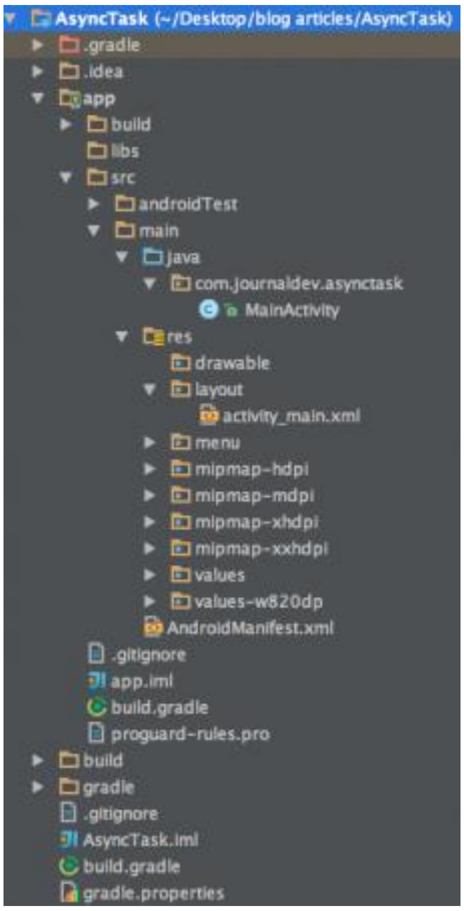
In the above snippet we've used a sample classname that extends AsyncTask and execute method is used to start the background thread.

Note:

- The AsyncTask instance must be created and invoked in the UI thread.
- The methods overridden in the AsyncTask class should never be called. They're called automatically
- AsyncTask can be called only once. Executing it again will throw an exception

In this tutorial we'll implement an AsyncTask that makes a process to go to sleep for a given period of time as set by the user.

Android Async Task Project Structure



```
AsyncTask (~/Desktop/blog articles/AsyncTask)
  .gradle
  .idea
  app
    build
    libs
    src
      androidTest
      main
        java
          com.journaldev.asynctask
            MainActivity
        res
          drawable
          layout
            activity_main.xml
          menu
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          values
          values-w820dp
        AndroidManifest.xml
    .gitignore
    app.iml
    build.gradle
    proguard-rules.pro
  build
  gradle
  .gitignore
  AsyncTask.iml
  build.gradle
  gradle.properties
```

Android AsyncTask Example Code

The xml layout is defined in the activity_main.xml and its given below:

`activity_main.xml`

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity" >


    <TextView

        android:id="@+id/tv_time"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:textSize="10pt"

        android:textColor="#444444"

        android:layout_alignParentLeft="true"

        android:layout_marginRight="9dip"

        android:layout_marginTop="20dip"

        android:layout_marginLeft="10dip"

        android:text="Sleep time in Seconds:"/>

    <EditText

        android:id="@+id/in_time"

        android:layout_width="150dip"

        android:layout_height="wrap_content"

        android:background="@android:drawable/editbox_background"

        android:layout_toRightOf="@id/tv_time"

        android:layout_alignTop="@id/tv_time"

        android:inputType="number"

        />

    <Button

        android:id="@+id/btn_run"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Run Async task"
        android:layout_below="@+id/in_time"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="64dp" />
    <TextView
        android:id="@+id/tv_result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="7pt"
        android:layout_below="@+id/btn_run"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

In the above layout we have used a predefined drawable as the border of the EditText.

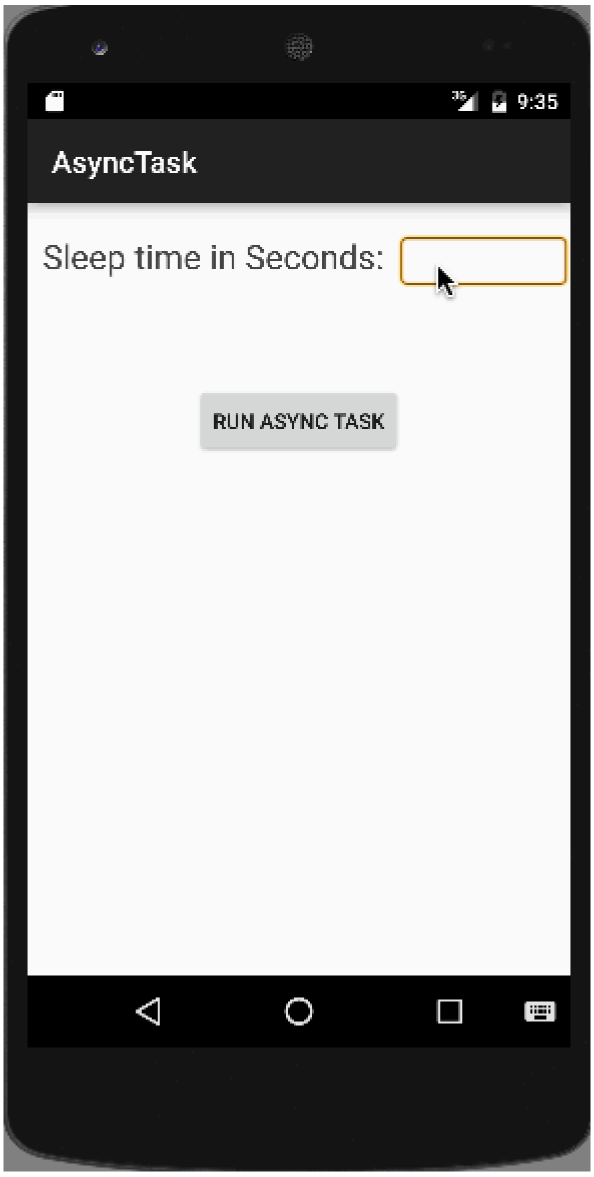
The MainActivity.java is defined below:

```
package com.journaldev.asynctask;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    private Button button;
    private EditText time;

    private TextView finalResult;


    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        time = (EditText) findViewById(R.id.in_time);

        button = (Button) findViewById(R.id.btn_run);

        finalResult = (TextView) findViewById(R.id.tv_result);

        button.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                AsyncTaskRunner runner = new AsyncTaskRunner();

                String sleepTime = time.getText().toString();

                runner.execute(sleepTime);

            }

        });

    }


    private class AsyncTaskRunner extends AsyncTask<String, String, String>
{

        private String resp;

        ProgressDialog progressDialog;

        @Override

        protected String doInBackground(String... params) {

            publishProgress("Sleeping..."); // Calls onProgressUpdate()

            try {

                int time = Integer.parseInt(params[0])*1000;

                Thread.sleep(time);
                resp = "Slept for " + params[0] + " seconds";
            } catch (InterruptedException e) {
                e.printStackTrace();
                resp = e.getMessage();
            } catch (Exception e) {
                e.printStackTrace();
                resp = e.getMessage();
            }
            return resp;
        }
        @Override
        protected void onPostExecute(String result) {
            // execution of result of Long time consuming operation
            progressDialog.dismiss();
            finalResult.setText(result);
        }
        @Override
        protected void onPreExecute() {
            progressDialog = ProgressDialog.show(MainActivity.this,
```

```
                    "ProgressDialog",
                    "Wait for "+time.getText().toString()+ " seconds");
        }



        @Override
        protected void onProgressUpdate(String... text) {
            finalResult.setText(text[0]);


        }
        }

}
```

In the above code we've used `AsyncTaskRunner` class to perform the AsyncTask operations. The time in seconds is passed as a parameter to the class and a ProgressDialog is displayed for the given amount of time.



Remote Connection

public class MyAsyncTask extends AsyncTask<String, String, Void> {

```
    private TextView resultTextView;

    public MyAsyncTask(TextView textView) {

        this.resultTextView = textView;
    }

    @Override
    protected Void doInBackground(String... params) {

        String id = params[0];
        try {
          URL url = new URL("http://192.168.43.71/testandroidconnection/getUser.php?id=" + id);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.connect();

            InputStream inputStream = connection.getInputStream();

            BufferedReader rd = new BufferedReader(new InputStreamReader(inputStream));
            String line = "";
            String answer = "";
            while ((line = rd.readLine()) != null) {
                answer += line;
            }

            publishProgress(answer);

        } catch (Exception e) {
            System.out.println("Error:   " + e.toString());
        }
        return null;
    }

    @Override
    protected void onProgressUpdate(String... progress) {
        super.onProgressUpdate();

        // Update the UI
        this.resultTextView.setText(progress[0]);

    }

}
```