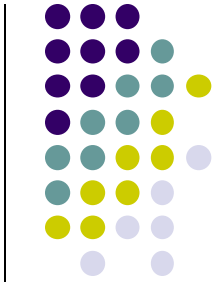
Mobile Application Develpment

Background Tasks in Android

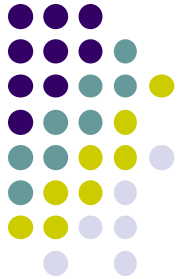
AsyncTask

What is AsyncTask



- **AsyncTask** enables proper and easy use of the **UI thread**. This class allows to **perform background operations and publish results on the UI thread** without having to manipulate **threads and/or handlers**.
- **AsyncTask** is designed to be a helper class. around Thread and Handler and does not constitute a generic **threading framework**.
- **AsyncTasks** should ideally be used for **short operations (a few seconds at the most.)** If you need to keep threads running for long periods of time, it is highly recommended you use the various **APIs** provided by the **java.util.concurrent pacakge** such as **Executor**, **ThreadPoolExecutor** and **FutureTask**.

What is AsyncTask



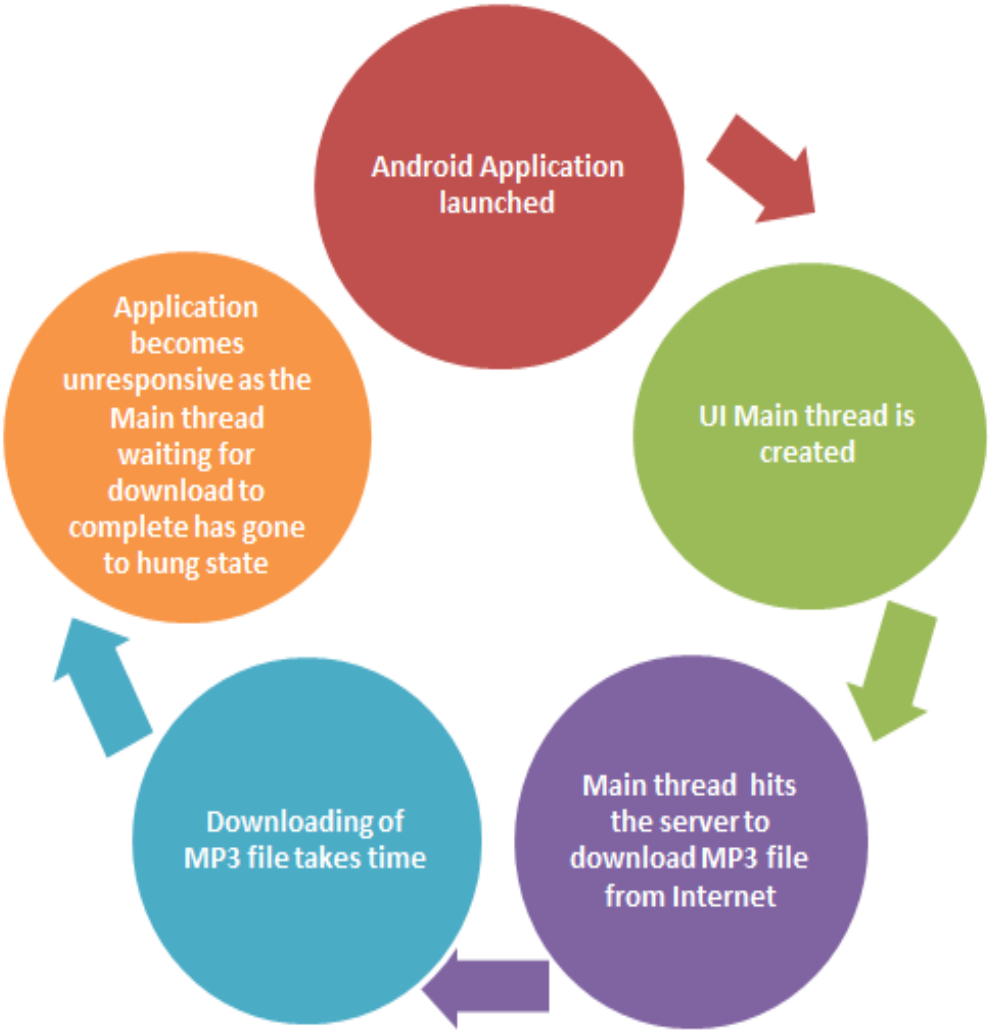
- **An asynchronous task** is defined by a computation that runs on a background thread and whose result is published on the UI thread.

- **An asynchronous task** is defined by **3 generic types**, called **Params**, **Progress** and **Result**,

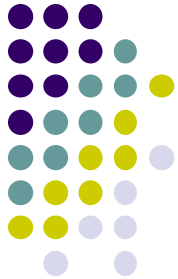
- **and 4 steps**,called
  - **onPreExecute**
  - **doInBackground**
  - **onProgressUpdate**
  - **onPostExecute**

When to use AsyncTask?

- Assume you have created a simple Android application which **downloads MP3** file from Internet on launching the application.
- The state diagram shows the series of operations that will take place when you launch the application you created:



When to use AsyncTask? cont

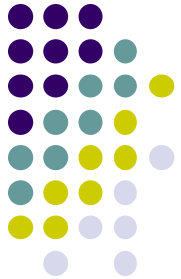


- As the response **(MP3 file)** from server is awaited, the **application** has **become unresponsive** since the **Main thread** is still **waiting** for download operation to complete. **To overcome this** we can create **new thread** and implement **run method** to perform this network call as similar as we usually do in normal Java applications, so that UI remains responsive.

- **But handling it** with separate thread may create some **additional issues** when we try to update UI based on the result of the operation performed since **Android UI toolkit** is not thread safe.

- Android took all these issues in consideration and **created a dedicated class called 'AsyncTask'** to handle the tasks/operations that need to be performed at the **background asynchronously**.

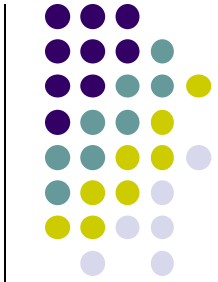
How to implement AsyncTask in Android applications?



- Create a **new class** inside **Activity class** and **subclass** it by extending **AsyncTask** as shown below.

```
private class DownloadMp3Task extends AsyncTask<URL, Integer, Long>
{
 protected Long doInBackground(URL... urls) {
     //Yet to code
    }
 protected void onProgressUpdate(Integer... progress) {
    //Yet to code
    }
 protected void onPostExecute(Long result) {
    //Yet to code
    }
}
```

How to implement AsyncTask in Android applications?



- **Execute** the task simply by invoking **execute method** as shown below:
- new DownloadMp3Task().**execute**(mp3URL);

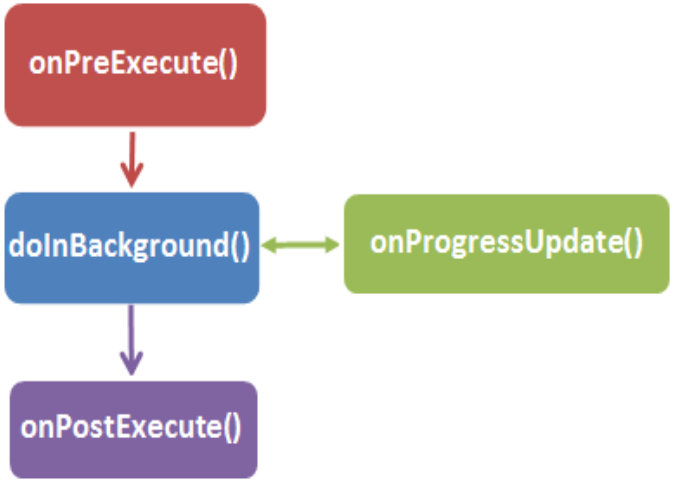
When an asynchronous task is executed, the task goes through 4 steps:

- **onPreExecute:** Invoked **before** the task is **executed** ideally before **doInBackground method** is called on the UI thread. This method is normally used to setup the task like showing progress bar in the UI.

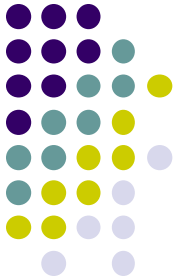
- **doInBackground:** Code running for long lasting time should be put in **doInBackground** method. When **execute method** is called in **UI main thread**, this method is called with the **parameters passed**.

- **onProgressUpdate:** Invoked by calling **publishProgress** at anytime from **doInBackground**. This method can be used to display any form of progress in the user interface.

- **onPostExecute:** Invoked after background computation in **doInBackground** method completes processing. Result of the **doInBackground** is passed to this method.

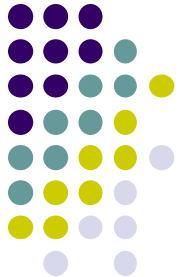


AsyncTask – Rules to be followed



1. The **AsyncTask** class must be **loaded on the UI thread**.
2. The **task instance** must be created on the **UI thread**.
3. Method **execute(Params…)** must be invoked on the **UI thread**.
4. Should not call **onPreExecute**(), **onPostExecute**(Result), **doInBackground**(Params…), **onProgressUpdate**(Progress…) manually.
5. The task can be **executed only once** (an exception will be thrown if a second execution is attempted.)

The basic structure of AsyncTask is as follows:



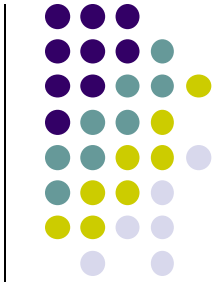
```
private class SampleAsyncTask extends AsyncTask<String, Integer, Integer> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
     @Override
    protected Integer doInBackground(String... params) {
        return null;
    }
     @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
    }
     @Override
    protected void onPostExecute(Integer result) {
        super.onPostExecute(result);
    }
}
```

Asynctask Example:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.uot96.myapplicationasynctask.MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:padding="20dp"
        android:text="Background Processing Using ASyncTask"
        tools:context=".AsyncTaskActivity" />
```

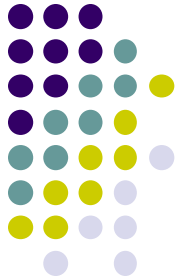
```
    <ProgressBar
        android:id="@+id/progressbar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="34dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/progressbar"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="40dp"
        android:minWidth="120dp"
        android:text="Process" />

    <TextView
        android:id="@+id/txtpercentage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/progressbar"
        android:text="Processing 0%"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```



MainActivity.java


```
package com.mycompany.asynctaskexample;

    import android.os.AsyncTask;
    import android.os.SystemClock;
    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import android.widget.ProgressBar;
    import android.widget.TextView;
    import android.widget.Toast;

    public class MainActivity extends AppCompatActivity {
        Button btnprocess;
        ProgressBar progressBar;
        TextView txtpercentage;
```

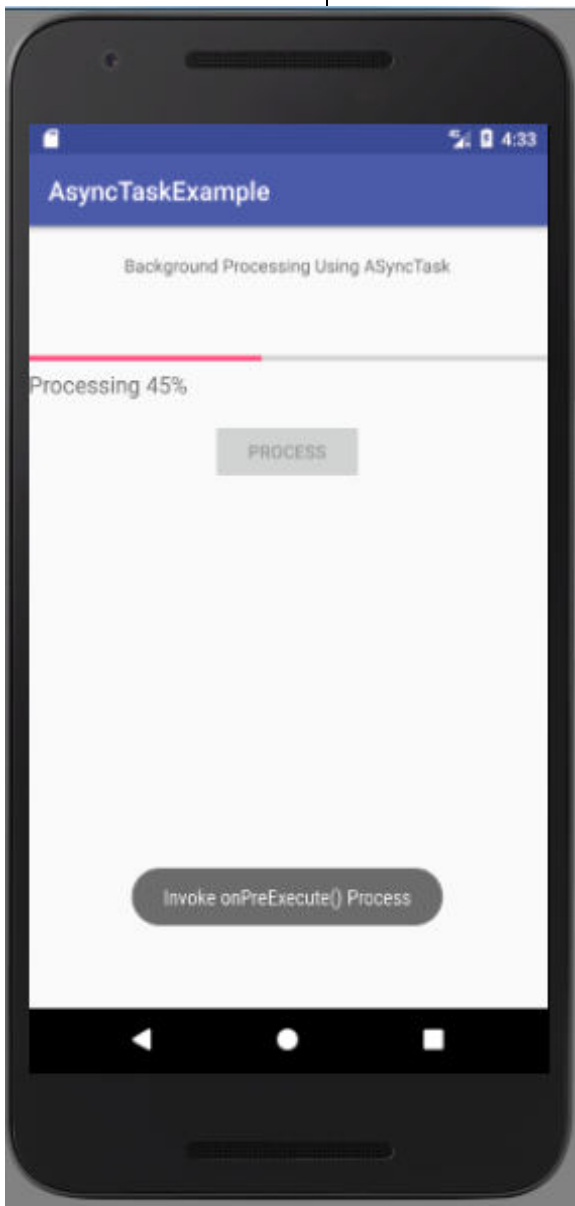
MainActivity.java



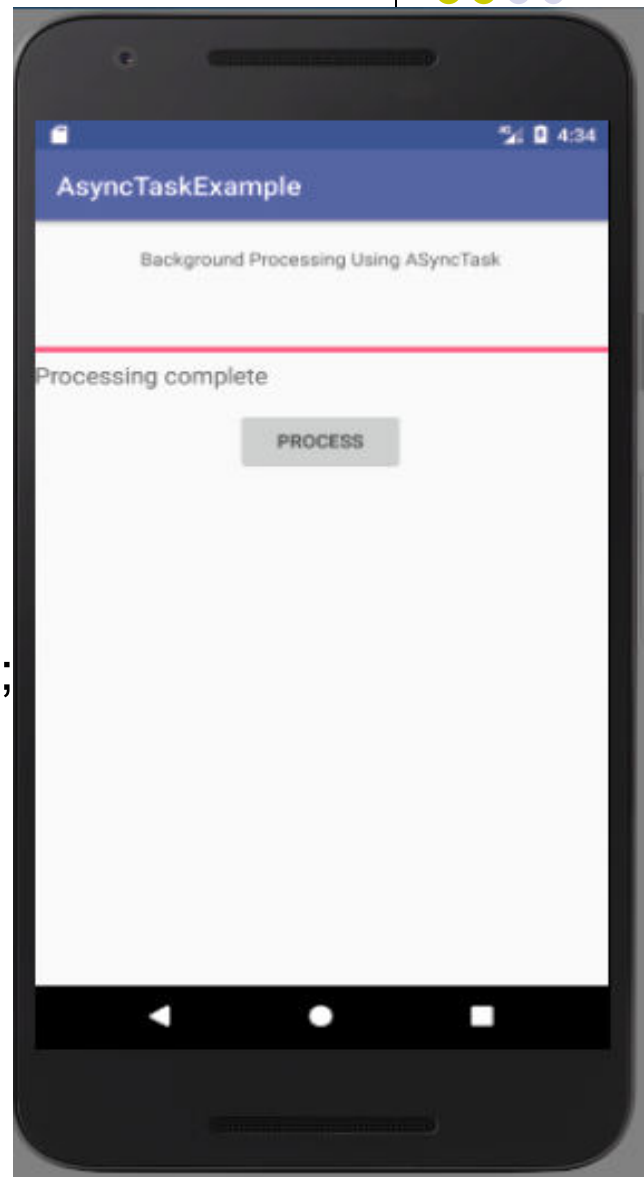
```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnprocess = (Button) findViewById(R.id.button);
        progressBar = (ProgressBar) findViewById(R.id.progressbar);
        txtpercentage= (TextView) findViewById(R.id.txtpercentage);

        btnprocess.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                btnprocess.setEnabled(false);
                new DoingAsyncTask().execute();
            }
        });
    }
```

```
private class DoingAsyncTask extends AsyncTask<Void,
    Integer, Void> {


 int progress_status;

@Override

protected void onPreExecute() {
     super.onPreExecute();
    Toast.makeText(MainActivity.this,"Invoke
    onPreExecute() Process",
    Toast.LENGTH_SHORT).show();
          progress_status = 0;
          txtpercentage.setText("Processing 0%");
       }
```

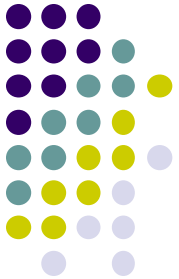
```
@Override
protected Void doInBackground(Void... params) {
        while(progress_status<100) {
            progress_status += 5;
            publishProgress (progress_status);
            SystemClock.sleep (200);
        }
          return null;
}
@Override
protected void onProgressUpdate(Integer... values)
   super.onProgressUpdate(values);
   progressBar.setProgress(values[0]);
   txtpercentage.setText("Processing+values[0]+"%")
}
```



```
@Override

protected void onPostExecute(Void result) {
    super.onPostExecute(result);

    Toast.makeText
    (MainActivity.this,"InvokeonPostExecute()
    Process", Toast.LENGTH_SHORT).show();

    txtpercentage.setText("Processing complete");
     btnprocess.setEnabled(true);
          }
       }
    }
```



Reference



- **Background tasks in Android**
  http://www.vogella.com/tutorials/AndroidBackgroundProcessing/article.html

- **Android – AsyncTask**

  https://developer.android.com/reference/android/os/AsyncTask.html