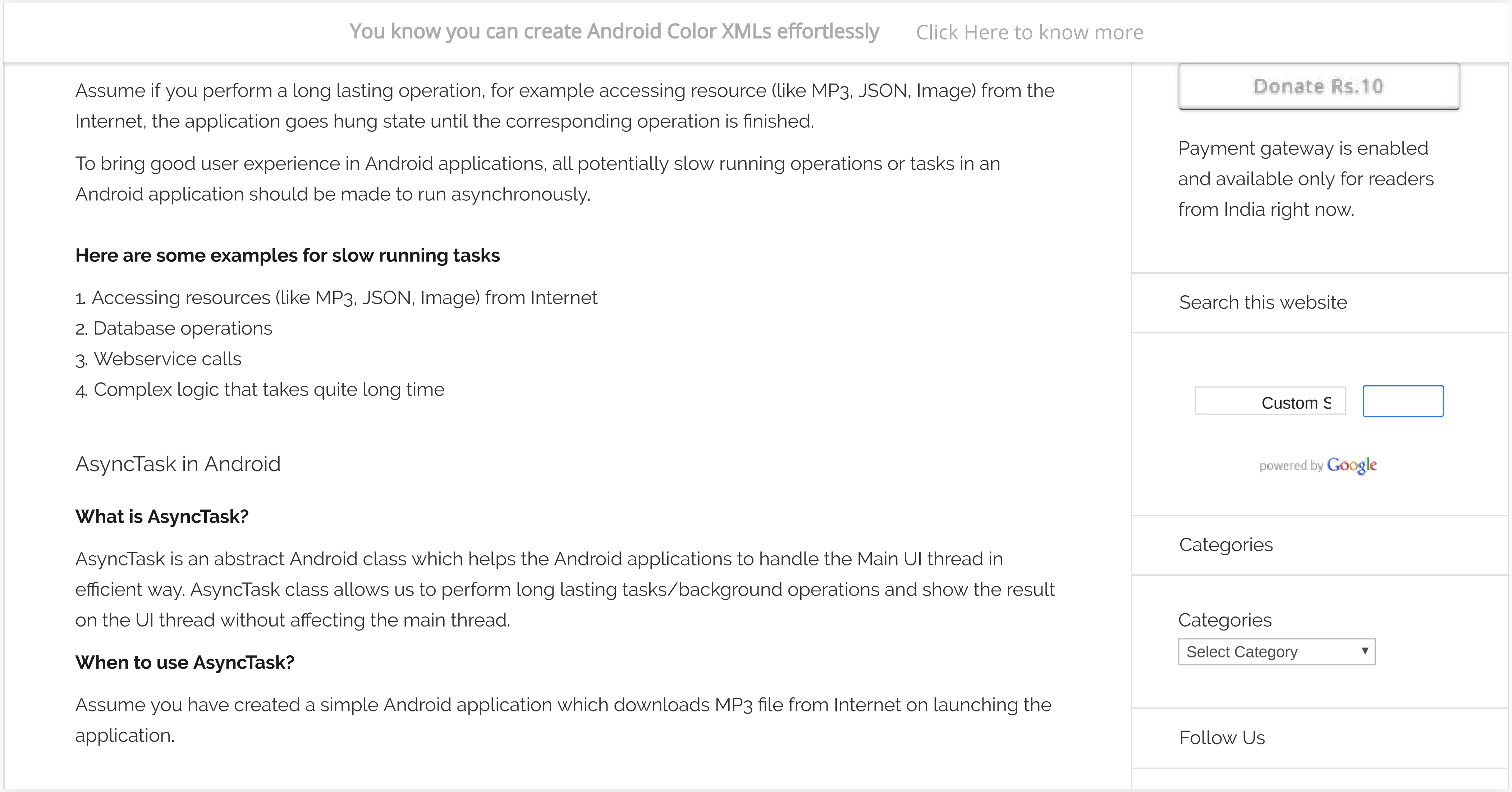
What is AsyncTask in Android?

Posted By Udhay on Jan 12, 2014 | 17 Comments

Android UI Main Thread

Android handles input events/tasks with a single User Interface (UI) thread and the thread is called Main thread. Main thread cannot handle concurrent operations as it handles only one event/operation at a time.

Concurrent Processing in Android



Assume if you perform a long lasting operation, for example accessing resource (like MP3, JSON, Image) from the Internet, the application goes hung state until the corresponding operation is finished.

To bring good user experience in Android applications, all potentially slow running operations or tasks in an Android application should be made to run asynchronously.

Here are some examples for slow running tasks

1. Accessing resources (like MP3, JSON, Image) from Internet
2. Database operations
3. Webservice calls
4. Complex logic that takes quite long time

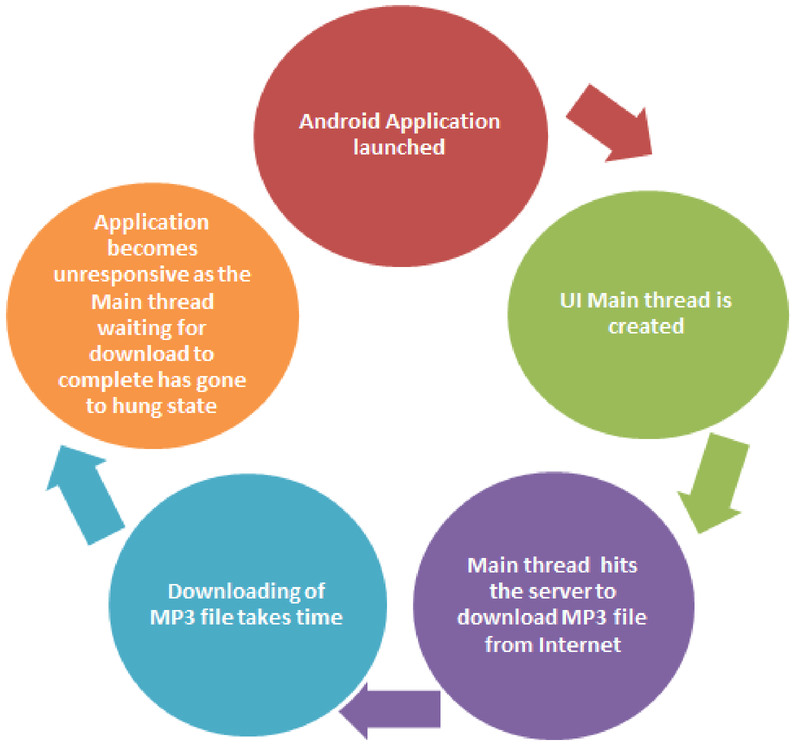
AsyncTask in Android

What is AsyncTask?

AsyncTask is an abstract Android class which helps the Android applications to handle the Main UI thread in efficient way. AsyncTask class allows us to perform long lasting tasks/background operations and show the result on the UI thread without affecting the main thread.

When to use AsyncTask?

Assume you have created a simple Android application which downloads MP3 file from Internet on launching the application.


Android Application launched
UI Main thread is created
Main thread hits the server to download MP3 file from Internet
Downloading of MP3 file takes time
Application becomes unresponsive as the Main thread waiting for download to complete has gone to hung state

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.

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.

> AsyncTask should only be used for tasks/operations that take quite few seconds. Some tasks may keep the thread running for long time so such tasks should be handled using java.util.concurrent package.

How to implement AsyncTask in Android applications?

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

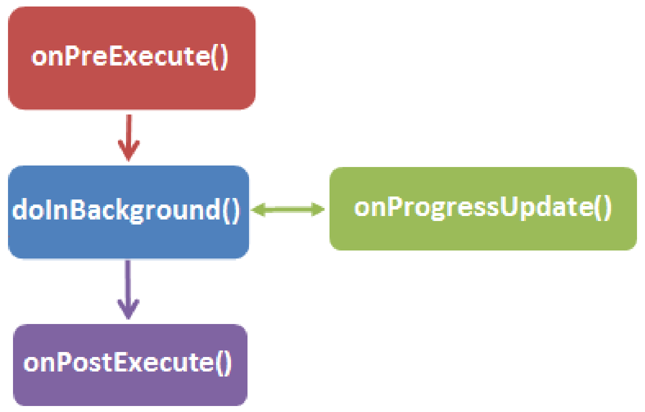
```
private class DownloadMp3Task extends AsyncTask<URL, Integer, Long> {
 protected Long doInBackground(URL... urls) {
      //Yet to code
     }
 protected void onProgressUpdate(Integer... progress) {
     //Yet to code
     }
 protected void onPostExecute(Long result) {
     //Yet to code
     }
}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

2. Execute the task simply by invoking execute method as shown below:

Android Game Trailers?
Mar 23, 2016

AsyncTask – The 4 steps

When an asynchronous task is executed from UI main thread, 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.

[pgsubscribe]

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.

Task Cancellation

The task can be cancelled by invoking cancel(boolean) method. This will cause subsequent calls to isCancelled() to return true. After invoking this method, onCancelled(Object) method is called instead of onPostExecute() after doInBackground() returns.

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.)

How to use AsyncTask in Android applications?