81

# How to set a timer in android

android

What is the proper way to set a timer in android in order to kick off a task (a function that I create which does not change the UI)? Use this the Java way: http://java.sun.com/j2se/1.4.2/docs/api/java/util/Timer.html

Or there is a better way in android (android's handler)?

share improve this question

n179911
3,019 ● 11 ● 45 ● 75

**Asked**
Dec 9 '09 at 22:35

RedBlueThing
21.8k ● 9 ● 62 ● 96

Edited
Sep 22 '10 at 23:29

10 Answers

order by votes

90

Standard Java way to use timers via java.util.Timer and java.util.TimerTask works fine in Android, but you should be aware that this method creates a new thread.

You may consider using the very convenient Handler class (android.os.Handler) and send messages to the handler via sendMessageAtTime(android.os.Message, long) or sendMessageDelayed(android.os.Message, long). Once you receive a message, you can run desired tasks. Second option would be to create a Runnable object and schedule it via Handler's functions postAtTime(java.lang.Runnable, long) or postDelayed(java.lang.Runnable, long).

share improve this answer

MannyNS
3,600 ● 1 ● 12 ● 14

**Answered**
Dec 10 '09 at 0:30

6 This is the wrong way to do things in android. In android, you want to user the Alarm Manager ( developer.android.com/reference/android/app/AlarmManager.html ) for running tasks far in the future. – **Kurtis Nusbaum** Oct 15 '11 at 4:16

6 @Kurtis Nusbaum The question doesn't say how far in the future the task is. – **Christopher Perry** Nov 5 '11 at 7:25

48 @KurtisNusbaum That's not necessarily true, it depends on the context. The docs on AlarmManager say, "Note: The Alarm Manager is intended for cases where you want to have your application code

run at a specific time, even if your application is not currently running. For normal timing operations (ticks, timeouts, etc) it is easier and much more efficient to use Handler." – **Christopher Perry** Nov 5 '11 at 18:42

3 @Scienceprodigy Ah, I see. Fair enough. – **Kurtis Nusbaum** Nov 5 '11 at 19:07

6 using the Handler method of scheduling a task is reliable only if the application has acquired a wakeLock and hence you are sure that the phone shall not go into the sleep state. If phone does go to the sleep state then sendMessageDelayed as well as sendMessageAtTime will not work. Hence in that scenario AlarmManager would be reliable choice. – **crazyaboutliv** Jan 27 '12 at 21:59

show **3** more comments

79

As I have seen it, java.util.Timer is the most used for implementing a timer.

**For a repeating task:**

```
new Timer().scheduleAtFixedRate(task, after, interval);
```

**For a single run of a task:**

```
new Timer().schedule(task, after);
```

**task** being the method to be executed
**after** the time to initial execution
(**interval** the time for repeating the execution)

share improve this answer

**MrThys**
1,990●14●60●100

**Answered**
Dec 10 '09 at 22:03

**naxa**
1,343●1●9●23

**Edited**
Sep 30 '13 at 7:55

5 I'll just add that time is in milliseconds – **Uri** Dec 27 '12 at 3:40

awesome thanks.. – **Houston** Apr 3 '13 at 6:12

1 android dev docs for scheduleAtFixedRate – **naxa** Sep 30 '13 at 7:51

1 task may be an instance of your class that inherits from java.util.TimerTask and overrides void run(). – **naxa** Sep 30 '13 at 7:54

So many upvotes. You could to this. But @Samuel describes the better way. See his "Which is explained Here" link for why. Also, can't update UI from a Timer thread! And see these other Handler-based answers in other threads: (simple) stackoverflow.com/a/6702767/199364, or (shows various alternatives) stackoverflow.com/a/4598737/199364 – **ToolmakerSteve** Sep 12 at 16:38

add a comment

66

yes java's timer *can be used*, but as the question asks for **better way** (for mobile). Which is explained **Here**.

---

**For the sake of StackOverflow:**

Since **Timer** creates a new thread it may be considered heavy,

if all you need is to get is a call back while the activity is running a **Handler** can be used in conjunction with a

**Runnable**:

```
private final int interval = 1000; // 1 Second
private Handler handler = new Handler();
private Runnable runnable = new Runnable(){
    public void run() {
        Toast.makeText(MyActivity.this, "C'Mom no hands!", Toast.LENGTH_SHORT).show();
    }
};
...
handler.postAtTime(runnable, System.currentTimeMillis()+interval);
handler.postDelayed(runnable, interval);
```

or a **Mesage**

```
private final int EVENT1 = 1;
private Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case Event1:
            Toast.makeText(MyActivity.this, "Event 1", Toast.LENGTH_SHORT).show();
            break;

        default:
            Toast.makeText(MyActivity.this, "Unhandled", Toast.LENGTH_SHORT).show();
            break;
        }
    }
};

...

Message msg = handler.obtainMessage(EVENT1);
handler.sendMessageAtTime(msg, System.currentTimeMillis()+interval);
handler.sendMessageDelayed(msg, interval);
```

on a side note this approach can be used, if you want to run a piece of code in the UI thread from an another thread.

if you need to get a call back even if your activity is not running then, you can use an **AlarmManager**

share improve this answer

Samuel
4,670●1●16●30

Answered
Oct 20 '10 at 6:42

Adrian Toman
5,025●3●23●38

Edited
Jul 15 at 8:24

1 Oh yea, I googled exactly this answer. Take my upvote for the useful link. – **ulidtko** Jul 24 '11 at 10:15

1 Link doesn't work :( – **SSH This** Jul 7 '12 at 5:34

1 updated the link, talk about google having trouble maintaining links. – **Samuel** Jul 10 '12 at 1:17

1 This article from 2007 - not saying it's wrong, but I'm always suspicious of a mobile article if it's older than 3 years. Things change pretty quickly. – **StackOverflowed** Sep 22 '12 at 13:13

1 it would be more like StackOverflow if you could quote the main points of your link here in your answer. – **naxa** Sep 30 '13 at 7:56

show **4** more comments

9

It is situational.

The Android documentation suggests that you should use AlarmManager to register an Intent that will fire at the specified time if your application may not be running.

Otherwise, you should use Handler.

> Note: The Alarm Manager is intended for cases where you want to have your application code run at a specific time, even if your application is not currently running. For normal timing operations (ticks, timeouts, etc) it is easier and much more efficient to use Handler.

share improve this answer

Timothy Lee Russell
2,558●1●12●25

Answered
Sep 22 '10 at 23:37

7

> > > Here we go.. We will need two classes. I am posting a code which changes mobile audio profile after each 5 seconds (5000 mili seconds) ...

***Our 1st Class***

```
public class ChangeProfileActivityMain extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);

        Timer timer = new Timer();
        TimerTask updateProfile = new CustomTimerTask(ChangeProfileActivityMain.this);
        timer.scheduleAtFixedRate(updateProfile, 0, 5000);
    }

}
```

***Our 2nd Class***

```
public class CustomTimerTask extends TimerTask {

    private AudioManager audioManager;
    private Context context;
    private Handler mHandler = new Handler();

    // Write Custom Constructor to pass Context
    public CustomTimerTask(Context con) {
        this.context = con;
    }

    @Override
    public void run() {
        // TODO Auto-generated method stub

        // your code starts here.
        // I have used Thread and Handler as we can not show Toast without starting new thread
when we are inside a thread.
        // As TimePicker has run() thread running., So We must show Toast through Handler.post in
a new Thread. Thats how it works in Android..
        new Thread(new Runnable() {
```

share improve this answer

Rizwan Sohaib
729●8●17

**Answered**
Apr 11 '12 at 6:47

**Edited**
Apr 11 '12 at 11:27

4

I'm an Android newbie but here is the timer class I created based on the answers above. It works for my app but I welcome any suggestions.

Usage example:

```
...{
public Handler uiHandler = new Handler();

  private Runnable runMethod = new Runnable()
    {
        public void run()
        {
              // do something
        }
    };

    timer = new UITimer(handler, runMethod, timeoutSeconds*1000);
        timer.start();
}...

public class UITimer
{
    private Handler handler;
    private Runnable runMethod;
    private int intervalMs;
    private boolean enabled = false;
```

share improve this answer

user868020
41●1

**Answered**
Jul 28 '11 at 17:49

2

I believe the way to do this on the android is that you need a background service to be running. In that background application, create the timer. When the timer "ticks" (set the interval for how long you want to wait), launch your activity which you want to start.

http://developer.android.com/guide/topics/fundamentals.html (<-- this article explains the relationship between activities, services, intents and other core fundamentals of Android development)

share improve this answer

Crowe T. Robot
1,808●9●16

**Answered**
Dec 9 '09 at 22:39

1

For timing operation you should use Handler. See http://developer.android.com/resources/articles/timed-ui-updates.html

If you need to run a background service the AlarmManager is the way to go. Have a look to the BuzzBox SDK

http://hub.buzzbox.com/android-sdk/

you can create a Task and schedule it using a cron string and it will work similarly to crontab on linux

share improve this answer

robsf
778●1●6●18

**Answered**
Mar 16 '11 at 0:56

1

Probably Timerconcept

```
new CountDownTimer(40000, 1000) { //40000 milli seconds is total time, 1000 milli seconds is time interval

 public void onTick(long millisUntilFinished) {
  }
  public void onFinish() {
 }
}.start();
```

or

Method 2 ::

Program the timer

Add a new variable of int named time. Set it to 0. Add the following code to onCreate function in MainActivity.java.

```
//Declare the timer
Timer t = new Timer();
//Set the schedule function and rate
t.scheduleAtFixedRate(new TimerTask() {

    @Override
    public void run() {
        //Called each time when 1000 milliseconds (1 second) (the period parameter)
    }

},
//Set how long before to start calling the TimerTask (in milliseconds)
0,
//Set the amount of time between each execution (in milliseconds)
1000);
```

Go into the run method and add the following code.

```
//We must use this function in order to change the text view text
runOnUiThread(new Runnable() {

    @Override
    public void run() {
        TextView tv = (TextView) findViewById(R.id.main_timer_text);
        tv.setText(String.valueOf(time));
        time += 1;
    }

});
```

share   improve this answer

vkulkarni
107●6

**Answered**
Oct 11 '13 at 8:18

1 I hope this one is helpful and may take less efforts to implement, Android CountDownTimer class

e.g.

```
 new CountDownTimer(30000, 1000) {

 public void onTick(long millisUntilFinished) {
     mTextField.setText("seconds remaining: " + millisUntilFinished / 1000);
 }

 public void onFinish() {
     mTextField.setText("done!");
 }
}.start();
```

share improve this answer

Gaurav Adurkar
17●4

**Answered**
Mar 10 at 7:25

## Your Answer

log in

or

**Name**

**Email**

**Home Page**

*By posting your answer, you agree to the privacy policy and terms of service.*

Post Your Answer