3

android CountDownTimer - last onTick not called - what clean solution to use?

android timer countdown

frustration post ....

I just stumbled into the "CountDownTimer - last onTick not called" problem many have reported here.

Simple demo showing the problem

```
    @Override
    public void onFinish() {
       // TODO Auto-generated method stub
       Log.e(TAG, " onFinish (" + getSeconds() + ")");
    }

    @Override
    public void onTick(long millisUntilFinished) {
       // TODO Auto-generated method stub
       Log.e(TAG, millisUntilFinished + " millisUntilFinished" + " (" + getSeconds() + ")");

    }

    protected long getSeconds() {
       return  (((System.currentTimeMillis() - startSec) / 1000) % 60);

    }

 }

 }
```

The logcat output from a test run ...

Time	PID	TID	Application	Tag	Text
09-15 01:18:41.595	8094	8094	com.example.gosh	CountDownTim…	9971 millisUntilFinished (0)
09-15 01:18:42.605	8094	8094	com.example.gosh	CountDownTim…	8963 millisUntilFinished (1)
09-15 01:18:43.605	8094	8094	com.example.gosh	CountDownTim…	7963 millisUntilFinished (2)
09-15 01:18:44.605	8094	8094	com.example.gosh	CountDownTim…	6963 millisUntilFinished (3)
09-15 01:18:45.605	8094	8094	com.example.gosh	CountDownTim…	5963 millisUntilFinished (4)
09-15 01:18:46.605	8094	8094	com.example.gosh	CountDownTim…	4963 millisUntilFinished (5)
09-15 01:18:47.605	8094	8094	com.example.gosh	CountDownTim…	3963 millisUntilFinished (6)
09-15 01:18:48.605	8094	8094	com.example.gosh	CountDownTim…	2963 millisUntilFinished (7)
09-15 01:18:49.605	8094	8094	com.example.gosh	CountDownTim…	1963 millisUntilFinished (8)
09-15 01:18:51.575	8094	8094	com.example.gosh	CountDownTim…	onFinish (10)

As you can see the last call onTick is happening with 1963ms millisUntilFinished, then the next call is onFinished nearly 2 seconds later. Surely a buggy behavior. I found many posts on this yet no clean solution yet. One I included in the source code, if you set the iDontWantThis field to 100 it works.

I dont mind workarounds in minor fields yet this seems such a core functionality that i cant fathom it wasnt fixed yet. What are you people doing to have a clean solution for this?

Thanks a lot

martin

UPDATE:

A very useful modification of the CountDownTimer by Sam which does not surpresses the last tick due to internal ms delay and also prevents the accumulation of ms delay with each tick over time can be found here

share improve this question

dorjeduck
1,571 1 12 34

Asked
Sep 14 '12 at 19:58

Edited
Oct 7 '12 at 2:50

3 Answers order by votes

3

The behavior you are experiencing is actually explicitly defined in the CountdownTimer code; have a look at the source.

Notice inside of handleMessage(), if the time remaining is less than the interval, it explicitly does not call onTick() and just delays until complete.

Notice, though, from the source that CountdownTimer is just a very thin wrapper on Handler, which is the real timing component of the Android framework. As a workaround, you could very easily create your own timer from this source (less than 150 lines) and remove this restriction to get your final tick callback.

share improve this answer

Devunwired
27.1k 4 52 83

Answered
Sep 14 '12 at 20:09

Thanks a lot, quite new to android and I havent made the step to actually look into its source code in cases like that. Thanks a lot for pointing me to that step. I will implement my own CountDownTimer as suggested which always calls onTick if millisUntilFinished > 0 and i can check easily if the remaining millis allow for what i want to do (actually i just need countdown timer which displays 6,5,4,3,2,1,0) – **dorjeduck** Sep 15 '12 at 2:34

just for the sake of completeness, removing the following else if block does the job to have onTick called whenever there are millis left ---------- } else if (millisLeft < mCountdownInterval) { // no tick, just delay until done sendMessageDelayed(obtainMessage(MSG), millisLeft); } – **dorjeduck** Sep 15 '12 at 2:48

add a comment

2

I think the frustration comes from an incorrect expectation of what a tick should be. As the other answer noted, this behavior is intentional. Another possible way of handling this is to simply specify a smaller interval. If you were implementing some sort of

countdown clock for example, it wouldn't hurt to change the interval to 500. If it's important that some work is only done when the seconds change, then you can do that too by storing the result of getSeconds() and only doing that work when that value changes.

If CountdownTimer were changed to always fire that last tick even if the remaining time is less than the interval, I'm sure StackOverflow would have a bunch of questions like "why do I not have enough time in the last tick of CountdownTimer?"

share improve this answer

kabuko
23.1k 4 29 54

Answered
Sep 14 '12 at 20:35

You are right I just expected different behaviour which is the source of the frustration. I am surprised the intended behaviour it is not mentioned in the API docu, particular as in the logcat above it can be seen the countdown timer it is actually "late" with every tick as it fires 47 ms later than the given secondsLeft/intervals and that very 47ms causes that onTick is not called as I was expecting it. Just imagine the case you give 4 hours as intervall, who would expect onTick not to be called because of 43ms. But yes that is the problem with expectations ;) – **dorjeduck** Sep 15 '12 at 2:28

add a comment

1

I don't understand why you say that it is intentional behaviour, the API says exactly:

"Schedule a countdown until a time in the future, with regular notifications on intervals along the way."

```
new CountDownTimer(30000, 1000) {

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

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

if you set the time to 30 seconds, and the countDownInterval to 1000, as the API says regular, it should be fired exactly 30 times. I think it's not an intentional behaviour but a wrong implementation.

The solution should be the one proposed by Sam here:

android CountDownTimer - additional milliseconds delay between ticks

share improve this answer

Juan Luis Carrasco
50 8

Answered
Aug 7 '13 at 11:36

Thanks Juan - Sam's solution is already referred to in the update of the post. – **dorjeduck** Aug 7 '13 at 13:00

add a comment

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