**Android 定时器的实现方式**

参考：

<http://wenku.baidu.com/link?url=1SQ5Q_tcct4FFK1fnXBwqs1TjaC1_aMgXj2-iCVwZLIaaY4Zy6Y_C62JUqm06Nzq0kXxR3EW_gI8qLLEtRTpGNyH27XIlOVjzrc7yneMt8m>

1. **Timer、TimerTask和Handle实现定时**

Timer timer = new Timer(); //定义定时器

//定义定时任务

TimerTask task = new TimerTask(){

        public void run(){

                 Message message = new Message();

                 message.what = 1 ;

                 handler.sendMessage(message);

        }

};

//开启定时器

timer.schedule(task, 2000, 2000);

//处理UI线程

final Handler handler = new Handler(){

        public void handleMessage(Message msg){

                switch(msg.what){

                  case 1:

                       System.out.print(" 定时器");

                        break;

                }

                super.handleMessage(msg);

        }

};

注： java.util.Timer.schedule(TimerTask task, long delay, long period); delay/1000秒执行task,然后经过period/1000秒再次执行task

//取消定时器

if (timer != null) {

        timer.cancel();

        timer = null;

}

if (timerTask != null) {

        timerTask.cancel();

        timerTask = null;

}

1. **Handle的public final boolean postDelayed(Runnable r, long delayMillis)方法实现定时**

1，首先创建一个Handler对象

Handler handler=new Handler();

2，然后创建一个Runnable对象

Runnable runnable=new Runnable(){

   @Override

   public void run() {

    //调用PostDelayed方法，两秒后再执行此Runnable对象，以实现每两秒实现一次的定时器操作

    handler.postDelayed(this, 2000);

   }

};

handle. postDelayed(runnable, 2000);

3，关闭此定时器

handler.removeCallbacks(runnable);

1. **线程sleep(long millis)方法和Handle实现定时**

Handler handler = new Handler() {

public void handleMessage(Message msg) {

switch(msg.what){

                case 1:

System.out.print(" 定时器");

break();

                }

    super.handleMessage(msg);

  }

};

public class MyThread implements Runnable{

@Override

 public void run() {

while (true) {

try {

Thread.sleep(10000);//线程暂停10秒

  Message message=new Message();

message.what=1;

      handler.sendMessage(message);//发送消息

} catch (InterruptedException e) {

  e.printStackTrace();

 }

   }

}

}

**new** Thread(**new** MyThread ()).start();

1. **ScheduledExecutorService（API24）和handle**

ScheduledExecutorService mExecutor = Executors .newSingleThreadScheduledExecutor();

mScheduledExecutorService.scheduleAtFixedRate(new myTask(), 5, 5, TimeUnit.SECONDS);

class myTask implements Runnable {

@Override public void run() {

if (images != null) {

Message message=new Message();

message.what=1;

handler.sendMessage(message);//发送消息

}

}

}

Handler handler = new Handler() {

public void handleMessage(Message msg) {

switch(msg.what){

                case 1:

System.out.print(" 定时器");

break();

                }

    super.handleMessage(msg);

  }

};

注：

1. 线程睡眠是帮助所有线程获得运行机会的最好方法。
2. 线程睡眠到期自动苏醒，并返回到可运行状态，不是运行状态。sleep()中指定的时间是线程不会运行的最短时间。因此，sleep()方法不能保证该线程睡眠到期后就开始执行。