How to schedule a periodic task in Java?

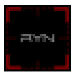
stackoverflow.com/questions/7814089/how-to-schedule-a-periodic-task-in-java

Ask Question
148

I need to schedule a task to run in at fixed interval of time. How can I do this with support of long intervals (for example on each 8 hours)?

I'm currently using `java.util.Timer.scheduleAtFixedRate`. Does `java.util.Timer.scheduleAtFixedRate` support long time intervals?

java scheduled-tasks
asked Oct 18 '11 at 21:38


RYN
6,0142087153

10 Answers

¿No encuentras la respuesta? Pregunta en Stack Overflow en español.

×
212

Use a ScheduledExecutorService:

```
private final ScheduledExecutorService scheduler = Executors.newScheduledThreadPool(1);
scheduler.scheduleAtFixedRate(yourRunnable, 8, 8, TimeUnit.HOURS);
```

answered Oct 18 '11 at 21:44

b_erb
15.9k84561

40

You should take a look to Quartz it's a java framework wich works with EE and SE editions and allows to define jobs to execute an specific time

answered Oct 18 '11 at 21:41

Jorge

10.7k1469112

20

Try this way ->

Firstly create a class TimeTask that run your task, it looks like:

```
public class CustomTask extends TimerTask  {

   public CustomTask(){

     //Constructor

   }

   public void run() {
       try {

         // Your task process

       } catch (Exception ex) {
           System.out.println("error running thread " + ex.getMessage());
       }
    }
}
```

then in main class you instantiate the task and run it periodically started by a specified date:

```
 public void runTask() {

        Calendar calendar = Calendar.getInstance();
        calendar.set(
           Calendar.DAY_OF_WEEK,
           Calendar.MONDAY
        );
        calendar.set(Calendar.HOUR_OF_DAY, 15);
        calendar.set(Calendar.MINUTE, 40);
        calendar.set(Calendar.SECOND, 0);
        calendar.set(Calendar.MILLISECOND, 0);



        Timer time = new Timer(); // Instantiate Timer Object

        // Start running the task on Monday at 15:40:00, period is set to 8 hours
        // if you want to run the task immediately, set the 2nd parameter to 0
        time.schedule(new CustomTask(), calendar.getTime(), TimeUnit.HOURS.toMillis(8));
}
```

answered Sep 19 '13 at 13:41

Shessuky
1,1581318

10

Use Google Guava `AbstractScheduledService` as given below:

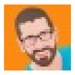
```
public class ScheduledExecutor extends AbstractScheduledService
{
   @Override
   protected void runOneIteration() throws Exception
   {
      System.out.println("Executing....");
   }

   @Override
   protected Scheduler scheduler()
   {
        return Scheduler.newFixedRateSchedule(0, 3, TimeUnit.SECONDS);
   }

   @Override
   protected void startUp()
   {
       System.out.println("StartUp Activity....");
   }


   @Override
   protected void shutDown()
   {
       System.out.println("Shutdown Activity...");
   }

   public static void main(String[] args) throws InterruptedException
   {
       ScheduledExecutor se = new ScheduledExecutor();
       se.startAsync();
       Thread.sleep(15000);
       se.stopAsync();
   }

}
```

If you have more services like this, then registering all services in ServiceManager will be good as all services can be started and stopped together. Read here for more on ServiceManager.

7

If you want to stick with `java.util.Timer`, you can use it to schedule at large time intervals. You simply pass in the period you are shooting for. Check the documentation here.

answered Oct 18 '11 at 21:44



Belizzle
4913722

3

If your application is already using Spring framework, you haveScheduling built in answered Oct 14 '14 at 22:25

Black
2,08033458

2

Do something every one second

```
Timer timer = new Timer();
timer.schedule(new TimerTask() {
       @Override
       public void run() {
           //code
       }
    }, 0, 1000);
```

answered Jun 12 '18 at 9:42

Duchic
291

1

Have you tried **Spring Scheduler** using annotations ?

```
@Scheduled(cron = "0 0 0/8 ? * * *")
public void scheduledMethodNoReturnValue(){
    //body can be another method call which returns some value.
}
```

you can do this with xml as well.

```
 <task:scheduled-tasks>
   <task:scheduled ref = "reference" method = "methodName" cron = "<cron expression here> -or-
${<cron expression from property files>}"
 <task:scheduled-tasks>
```

answered Dec 28 '18 at 14:49

Ask_Me_A_Question
442515

1

I use Spring Framework's feature. (*spring-context* jar or maven dependency).

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;


@Component
public class ScheduledTaskRunner {

    @Autowired
    @Qualifier("TempFilesCleanerExecution")
    private ScheduledTask tempDataCleanerExecution;

    @Scheduled(fixedDelay = TempFilesCleanerExecution.INTERVAL_TO_RUN_TMP_CLEAN_MS /* 1000 */)
    public void performCleanTempData() {
        tempDataCleanerExecution.execute();
    }

}
```

ScheduledTask is my own interface with my custom method ***execute***, which I call as my scheduled task.

answered Jul 21 '16 at 11:28

Yan Khonski

3,56962350

0

These two classes can work together to schedule a periodic task:

Scheduled Task

```
import java.util.TimerTask;
import java.util.Date;

// Create a class extending TimerTask
public class ScheduledTask extends TimerTask {
    Date now;
    public void run() {
        // Write code here that you want to execute periodically.
        now = new Date();                      // initialize date
        System.out.println("Time is :" + now); // Display current time
    }
}
```

Run Scheduled Task

```
import java.util.Timer;

public class SchedulerMain {
    public static void main(String args[]) throws InterruptedException {
        Timer time = new Timer();               // Instantiate Timer Object
        ScheduledTask st = new ScheduledTask(); // Instantiate SheduledTask class
        time.schedule(st, 0, 1000);             // Create task repeating every 1 sec
        //for demo only.
        for (int i = 0; i <= 5; i++) {
            System.out.println("Execution in Main Thread...." + i);
            Thread.sleep(2000);
            if (i == 5) {
                System.out.println("Application Terminates");
                System.exit(0);
            }
        }
    }
}
```

Reference https://www.mkyong.com/java/how-to-run-a-task-periodically-in-java/

answered Nov 21 '18 at 11:43

SumiSujith
296