Scheduling a task in java

3

In my java web application, I want to schedule a task.

I have searched the web alot and couldn't find a suitable scheduler for my case. In the application I have different types of users. For a specific user type, I want to schedule a task.

By the time a critical action is taken by a user:

- I want to send an email to that specific user and after 15 minutes
- I want to send another email and after 30 minutes
- I want to send another email and shut down the scheduler.

I know when the users take critical actions and how to send email but I don't have much experience about scheduling.

Can anyone help me for the case?

asked Mar 11 '14 at 9:04

nudaStck
1061416

4 Answers

×
2

Why don't you use a ScheduledExecutor?
http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ScheduledExecutorService.html

It has a method `schedule` which allows you to schedule whatever task you want (you pass a runnable). So basically, for each user you schedule a task of sending an e-mail and scheduling another task.

```
import static java.util.concurrent.TimeUnit.*;

class Task implements Runnable {
   private final User user;
   private final int rep;
   private final ScheduledExecutorService scheduler;

   public Task(User user, int rep, ScheduledExecutorService scheduler) {
       this.user = user;
       this.rep = rep;
       this.scheduler = scheduler;
   }

   public void run() {
       // send an e-mail to user
       if (rep > 0)
           scheduler.schedule(new Task(user, rep - 1, scheduler), 15, MINUTES);
   }
}

class Example {
    private final ScheduledExecutorService scheduler =
        Executors.newScheduledThreadPool(1);

    public void sendEmails() {
        // foreach user
        scheduler.submit(new Task(user, 3, scheduler));
    }
}
```

You might want to use a scheduled thread pool with more than one thread.

2

Use Quartz Scheduler to schedule a task

Steps Required -

1) Quartz job

```
public class HelloJob implements Job {
    public void execute(JobExecutionContext context) throws JobExecutionException {
    System.out.println("Hello Quartz!");
    }
}
```

2) Creating a trigger - CronTrigger – Run every 30 seconds

```
CronTrigger trigger = new CronTrigger();
trigger.setName("dummyTriggerName");
trigger.setCronExpression("0/30 * * * * ?");
```

3) Creating a scheduler

```
Scheduler scheduler = new StdSchedulerFactory().getScheduler();
scheduler.start();
scheduler.scheduleJob(job, trigger);
```

1

Here's a tutorial on how to use Java Timers:
http://enos.itcollege.ee/~jpoial/docs/tutorial/essential/threads/timer.html

You can create multiple Timer tasks in sequence to fufill your objective.

Example Code Quote:

```
import java.util.Timer;
import java.util.TimerTask;

/**
 * Simple demo that uses java.util.Timer to schedule a task
 * to execute once 5 seconds have passed.
 */

public class Reminder {
Timer timer;

public Reminder(int seconds) {
    timer = new Timer();
    timer.schedule(new RemindTask(), seconds*1000);
}

class RemindTask extends TimerTask {
    public void run() {
        System.out.format("Time's up!%n");
        timer.cancel(); //Terminate the timer thread
    }
}

public static void main(String args[]) {
    new Reminder(5);
    System.out.format("Task scheduled.%n");
}
}
```

answered Mar 11 '14 at 9:09

Harold
15517

0

I would recommend you to take a look at the quartz scheduling API http://quartz-scheduler.org/

I have used it in several projects so far and it is really easy to setup and configure new jobs in. It supports cron based triggers or simpletriggers so you can either calculate the times for the scheduled events in your java code or you can simply pass it a cron string.

Another advantage is that its really easy to configure with spring.

answered Mar 11 '14 at 10:32

steelshark

579310