09: Java FutureTask example

Posted on May 2, 2015 by Arulkumaran Kumaraswamipillai

Java 5 introduced the concurrent package for more efficient multi-threading.

Q. What is the difference between Future and FutureTask in asynchronous processing?
A. Future is the interface and FutureTask is the base implementation of the Future with methods to start and cancel a computation. The FutureTask provides asynchronous computation with methods to start and cancel a computation, query to see if the computation is complete, and retrieve the result of the computation. The result can only be retrieved when the computation has completed. The get method will block if the computation has not yet completed. Once the computation has completed, the computation cannot be restarted or cancelled.

Here is an example with 2 tasks. One is an internal short task that takes ~1 second. The second task is an external long running task taking 4 ~ 10 seconds. It is imperative that long running tasks need to have proper processing timeouts.

```
1
2  import java.util.concurrent.Callable;
3  import java.util.concurrent.ExecutionException;
4  import java.util.concurrent.ExecutorService;
5  import java.util.concurrent.Executors;
6  import java.util.concurrent.FutureTask;
7  import java.util.concurrent.TimeUnit;
8  import java.util.concurrent.TimeoutException;
9
10 public class FutureTaskExample {
11
12  // inner class
13  static class InternalProcess implements Callabl
14
15   @Override
16   public Integer call() throws Exception {
17
18      // just to simulate short internal process
19      TimeUnit.SECONDS.sleep(1);
20      return 2;
21   }
22
23  }
24
25  // inner class
26  static class ExternalProcess implements Callabl
27
28    @Override
29    public Integer call() throws Exception {
30       // just to simulate long running external
31       TimeUnit.SECONDS.sleep(15);
32       return 12;
33    }
34  }
35
36  public static void main(String[] args) {
37    InternalProcess callable1 = new InternalProce
38    ExternalProcess callable2 = new ExternalProce
39
40    FutureTask<Integer> futureTask1 = new FutureT
41    FutureTask<Integer> futureTask2 = new FutureT
42
43    // create a fixed thread pool with 2 threads
44    ExecutorService executor = Executors.newFixed
45    // add future tasks to the pool
46    executor.execute(futureTask1);
47    executor.execute(futureTask2);
48
49    while (true) {
50     try {
51       if (futureTask1.isDone() &amp;amp;&amp;amp; future
52          System.out.println("Shutting down the e
53          // shut down executor service
54          executor.shutdown();
55          return;
56        }
57
```


As a Java Architect

Java architecture & design concepts

```
58          //if not done do it once
59          if (!futureTask1.isDone()) {
60             // wait indefinitely for future task t
61             System.out.println("Task1 output = " +
62          }
63
64          System.out.println("Waiting for FutureTask
65          //try the external task with the timeout o
66          Integer result = futureTask2.get(5, TimeUn
67          if (result != null) {
68              System.out.println("Task2 output = " +
69          }
70
71      } catch (InterruptedException ie) {
72          ie.printStackTrace();
73      } catch (TimeoutException e) {
74          // do nothing as we want to process it asy
75          // if you want to time out then uncomment
76
77          //System.out.println("Cancelling Task2 due
78          //futureTask2.cancel(true); // true means
79      } catch (ExecutionException e) {
80          e.printStackTrace();
81      }
82    }
83  }
84
85 }
86
```

The output is

```
1
2 Task1 output = 2
3 Waiting for FutureTask2 to complete
4 Waiting for FutureTask2 to complete
5 Waiting for FutureTask2 to complete
6 Task2 output = 12
7 Shutting down the executor.
8
```

If you re-run it by uncommenting the last 2 lines in the TimeoutException catch block, you will get task2 cancelled.

```
1
2 Task1 output = 2
3 Waiting for FutureTask2 to complete
4 Cancelling Task2 due to timeout
5 Shutting down the executor.
6
```

Popular Posts

interview Q&As with diagrams | What should be a typical Java EE architecture?

Senior Java developers must have a good handle on

open all | close all

- Best Practice (6)
- Coding (26)
- Concurrency (6)
- Design Concepts (7)
- Design Patterns (11)
- Exception Handling (3
- Java Debugging (21)
- Judging Experience I
- Low Latency (7)
- Memory Managemen
- Performance (13)
- QoS (8)
- Scalability (4)
- SDLC (6)
- Security (13)
- Transaction Managen

80+ step by step Java Tutorials

open all | close all

- Setting up Tutorial (6)
- Tutorial - Diagnosis (2
- Akka Tutorial (9)
- Core Java Tutorials (2
- Hadoop & Spark Tuto

♦ 11 Spring boot interview questions & answers

885 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

839 views

18 Java scenarios based interview Questions and Answers

454 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

410 views

♦ 7 Java debugging interview questions & answers

315 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

310 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

302 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

286 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

265 views

8 Git Source control system interview questions & answers

221 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are

JEE Tutorials (19)
Scala Tutorials (1)
Spring & HIbernate Tu
Tools Tutorials (19)
Other Tutorials (45)

100+ Preparing for pre-interview Java written home assignments & coding tests

open all | close all

Can you write code? (
♦ Complete the given
Converting from A to I
Designing your classe
Java Data Structures
Passing the unit tests
What is wrong with th
Writing Code Home A
Written Test Core Jav
Written Test JEE (1)

How good are your...to go places?

open all | close all

Career Making Know-
Job Hunting & Resum

outdated and replaced with this subscription based site.

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

‹ Proxy design pattern in Java with service retry example

Graph from scratch Java example adjacent matrix approach ›

Posted in Multithreading

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

↑