ImportNew

什么时候使用CountDownLatch

2015/04/14 | 分类： 基础技术 | 1 条评论 | 标签： CountDownLatch, 工具类

本文由 ImportNew - 张涛 翻译自 howtodoinjava。欢迎加入翻译小组。转载请见文末要求。

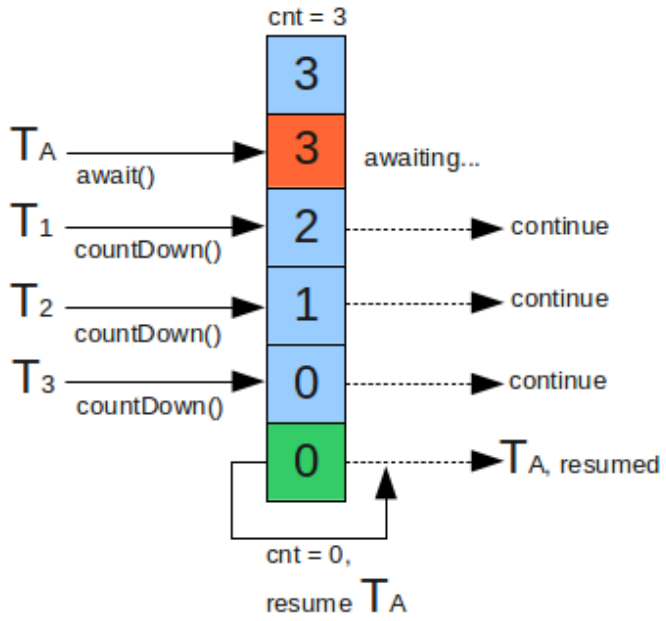
正如每个Java文档所描述的那样，CountDownLatch是一个同步工具类，它允许一个或多个线程一直等待，直到其他线程的操作执行完后再执行。在Java并发中，countdownlatch的概念是一个常见的面试题，所以一定要确保你很好的理解了它。在这篇文章中，我将会涉及到在Java并发编 程中跟CountDownLatch相关的以下几点：

目录

CountDownLatch是什么

CountDownLatch是在java1.5被引入的，跟它一起被引入的并发工具类还有CyclicBarrier、Semaphore、ConcurrentHashMap和BlockingQueue，它们都存在于java.util.concurrent包下。CountDownLatch这个类能够使一个线程等待其他线程完成各自的工作后再执行。例如，应用程序的主线程希望在负责启动框架服务的线程已经启动所有的框架服务之后再执行。

CountDownLatch是通过一个计数器来实现的，计数器的初始值为线程的数量。每当一个线程完成了自己的任务后，计数器的值就会减1。当计数器值到达0时，它表示所有的线程已经完成了任务，然后在闭锁上等待的线程就可以恢复执行任务。



CountDownLatch的伪代码如下所示：

```
//Main thread start
//Create CountDownLatch for N threads
//Create and start N threads
//Main thread wait on latch
//N threads completes there tasks are returns
//Main thread resume execution
```

CountDownLatch如何工作

CountDownLatch.java类中定义的构造函数：

```
//Constructs a CountDownLatch initialized with the given count.
public void CountDownLatch(int count) {...}
```

构造器中的计数值（count）实际上就是闭锁需要等待的线程数量。这个值只能被设置一次，而且CountDownLatch没有提供任何机制去重新设置这个计数值。

与CountDownLatch的第一次交互是主线程等待其他线程。主线程必须在启动其他线程后立即调用CountDownLatch.await()方法。这样主线程的操作就会在这个方法上阻塞，直到其他线程完成各自的任务。

其他N 个线程必须引用闭锁对象，因为他们需要通知CountDownLatch对象，他们已经完成了各自的任务。这种通知机制是通过 CountDownLatch.countDown()方法来完成的；每调用一次这个方法，在构造函数中初始化的count值就减1。所以当N个线程都调 用了这个方法，count的值等于0，然后主线程就能通过await()方法，恢复执行自己的任务。

在实时系统中的使用场景

让我们尝试罗列出在java实时系统中CountDownLatch都有哪些使用场景。我所罗列的都是我所能想到的。如果你有别的可能的使用方法，请在留言里列出来，这样会帮助到大家。

1. 实现最大的并行性：有时我们想同时启动多个线程，实现最大程度的并行性。例如，我们想测试一个单例类。如果我们创建一个初始计数为1的CountDownLatch，并让所有线程都在这个锁上等待，那么我们可以很轻松地完成测试。我们只需调用 一次countDown()方法就可以让所有的等待线程同时恢复执行。
2. 开始执行前等待n个线程完成各自任务：例如应用程序启动类要确保在处理用户请求前，所有N个外部系统已经启动和运行了。
3. 死锁检测：一个非常方便的使用场景是，你可以使用n个线程访问共享资源，在每次测试阶段的线程数目是不同的，并尝试产生死锁。

CountDownLatch使用例子

在这个例子中，我模拟了一个应用程序启动类，它开始时启动了n个线程类，这些线程将检查外部系统并通知闭锁，并且启动类一直在闭锁上等待着。一旦验证和检查了所有外部服务，那么启动类恢复执行。

BaseHealthChecker.java：这个类是一个Runnable，负责所有特定的外部服务健康的检测。它删除了重复的代码和闭锁的中心控制代码。

```
public abstract class BaseHealthChecker implements Runnable {

    private CountDownLatch _latch;
    private String _serviceName;
    private boolean _serviceUp;

    //Get latch object in constructor so that after completing the task, thread can countDown() the la
    public BaseHealthChecker(String serviceName, CountDownLatch latch)
    {
        super();
        this._latch = latch;
        this._serviceName = serviceName;
        this._serviceUp = false;
    }

    @Override
    public void run() {
        try {
            verifyService();
            _serviceUp = true;
        } catch (Throwable t) {
            t.printStackTrace(System.err);
            _serviceUp = false;
        } finally {
            if(_latch != null) {
                _latch.countDown();
            }
        }
    }

    public String getServiceName() {
        return _serviceName;
    }

    public boolean isServiceUp() {
        return _serviceUp;
    }
    //This methos needs to be implemented by all specific service checker
    public abstract void verifyService();
}
```

NetworkHealthChecker.java：这个类继承了BaseHealthChecker，实现了verifyService()方法。DatabaseHealthChecker.java和CacheHealthChecker.java除了服务名和休眠时间外，与NetworkHealthChecker.java是一样的。

```
public class NetworkHealthChecker extends BaseHealthChecker
{
    public NetworkHealthChecker (CountDownLatch latch)  {
        super("Network Service", latch);
    }

    @Override
    public void verifyService()
    {
        System.out.println("Checking " + this.getServiceName());
        try
        {
            Thread.sleep(7000);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
        System.out.println(this.getServiceName() + " is UP");
    }
}
```

ApplicationStartupUtil.java：这个类是一个主启动类，它负责初始化闭锁，然后等待，直到所有服务都被检测完。

```
public class ApplicationStartupUtil
{
    //List of service checkers
    private static List<BaseHealthChecker> _services;

    //This latch will be used to wait on
    private static CountDownLatch _latch;

    private ApplicationStartupUtil()
    {
    }

    private final static ApplicationStartupUtil INSTANCE = new ApplicationStartupUtil();

    public static ApplicationStartupUtil getInstance()
    {
        return INSTANCE;
    }

    public static boolean checkExternalServices() throws Exception
    {
        //Initialize the latch with number of service checkers
        _latch = new CountDownLatch(3);

        //All add checker in lists
        _services = new ArrayList<BaseHealthChecker>();
        _services.add(new NetworkHealthChecker(_latch));
        _services.add(new CacheHealthChecker(_latch));
        _services.add(new DatabaseHealthChecker(_latch));

        //Start service checkers using executor framework
        Executor executor = Executors.newFixedThreadPool(_services.size());

        for(final BaseHealthChecker v : _services)
        {
            executor.execute(v);
        }

        //Now wait till all services are checked
        _latch.await();

        //Services are file and now proceed startup
        for(final BaseHealthChecker v : _services)
        {
            if( ! v.isServiceUp())
            {
                return false;
            }
        }
        return true;
    }
}
```

现在你可以写测试代码去检测一下闭锁的功能了。

```
public class Main {
    public static void main(Str[] args)
    {
        boolean result = false;
        try {
            result = ApplicationStartupUtil.checkExternalServices();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("External services validation completed !! Result was :: "+ result);
    }
}
```

```
Output in console:

Checking Network Service
Checking Cache Service
Checking Database Service
Database Service is UP
Cache Service is UP
Network Service is UP
External services validation completed !! Result was :: true
```

常见面试题

可以为你的下次面试准备以下一些CountDownLatch相关的问题：

- 解释一下CountDownLatch概念?
- CountDownLatch 和CyclicBarrier的不同之处?
- 给出一些CountDownLatch使用的例子?
- CountDownLatch 类中主要的方法?

下载上述例子的源代码，请点击如下链接：

- 源码下载

原文链接： howtodoinjava 翻译： ImportNew.com - 张涛
译文链接： http://www.importnew.com/15731.html
[转载请保留原文出处、译者和译文链接。]

关于作者： 张涛

（新浪微博：@zthenry）

查看张涛的更多文章 >>

31

相关文章

- 排名Top 16的Java实用类库
- 工具类与函数编程毫不相干
- Java8中的java.util.Random类
- SPRING SECURITY JAVA配置：简介
- 谁在关心toString的性能?
- Java NIO：浅析I/O模型
- 40个Java多线程问题总结
- Java中的几个HashMap/ConcurrentHashMap实现分析
- Java提高篇之hashCode
- Jsoup代码解读（2）：DOM相关对象

发表评论

Comment form

Name*

姓名

邮箱*

请填写邮箱

网站（请以 http://开头）

请填写网站地址

评论内容*