ListenableFuture和CompletableFuture简单小结

时间 2018-05-19

标签 Future (http://www.voidcn.com/tag/Future) ListenableFuture (http://www.voidcn.com/tag/ListenableFuture) CompletableFuture (http://www.voidcn.com/tag/CompletableFuture)

原文 http://blog.csdn.net/androidlushangderen/article/details/80372711

前言

最近花了点时间熟悉了下ListenableFuture和CompletableFuture的使用。二者都是原生JDK中老版Future-Get模式的改进。本文将结合demo程序来直观的学习一下这两大Future的使用特点。

老版Future模式的缺点

老版Future模式一个最大的问题是需要获取结果做后续处理操作的时候，还是需要阻塞等待。这样的话，和同步调用方式就没有多大区别了。而ListenableFuture和CompletableFuture对于这种情况则是提供了很多易用的API。

如果说按照先后顺序来讲的话，首先是ListenableFuture，这是由Google Guava工具包提供的Future扩展类，随后，JDK在1.8版本中马上也提供了类似这样的类，就是CompletableFuture。

ListenableFuture

先来聊聊ListenableFuture，一句话概括ListenableFuture和JDK原生Future最大的区别是前者做到了一个可以监听结果的Future。换个更通俗的讲法，就是它可以监听异步执行的过程，执行完了，自动触发什么操作。除此之外，可以分别针对成功的情况，或者失败的情况做各种后续处理。具体使用可以看下面笔者写的demo程序。

```
package

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.Executors;

import org.junit.Test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.google.common.util.concurrent.FutureCallback;
import com.google.common.util.concurrent.Futures;
import com.google.common.util.concurrent.ListenableFuture;
import com.google.common.util.concurrent.ListeningExecutorService;
import com.google.common.util.concurrent.MoreExecutors;

/** * The unit test for ListenableFuture/CompletableFuture. * Created by yiqun01.lin * on 2018/5/3. */
public class TestFutures {
  //线程池中线程个数
  private static final int POOL_SIZE = 50;
  //带有回调机制的线程池
  private static final ListeningExecutorService service = MoreExecutors
          .listeningDecorator(Executors.newFixedThreadPool(POOL_SIZE));

  private static Logger LOG = LoggerFactory.getLogger(TestFutures.class);

  @Test
  public void testListenableFuture() {
    final List<String> value = Collections
        .synchronizedList(new ArrayList<String>());
    try {
      List<ListenableFuture<String>> futures = new ArrayList<ListenableFuture<String>>();
      // 将实现了callable的任务放入到线程池中，得到一个带有回调机制的ListenableFuture实例，
      // 通过Futures.addCallback方法对得到的ListenableFuture实例进行监听，一旦得到结果就进入到onSuccess方法中，
      // 在onSuccess方法中将查询的结果存入到集合中
      for (int i = 0; i < 1; i++) {
        final int index = i;
        if (i == 9) {
          Thread.sleep(500 * i);
        }
        ListenableFuture<String> sfuture = service
            .submit(new Callable<String>() {
              @Override
              public String call() throws Exception {
                long time = System.currentTimeMillis();
                LOG.info("Finishing sleeping task{}: {}", index, time);
                return String.valueOf(time);
              }
            });
        sfuture.addListener(new Runnable() {
          @Override
          public void run() {
            LOG.info("Listener be triggered for task{}.", index);
          }
        }, service);
```

```
        Futures.addCallback(sfuture, new FutureCallback<String>() {
          public void onSuccess(String result) {
            LOG.info("Add result value into value list {}.", result);
            value.add(result);
          }

          public void onFailure(Throwable t) {
            LOG.info("Add result value into value list error.", t);
            throw new RuntimeException(t);
          }
        });
        // 将每一次查询得到的ListenableFuture放入到集合中
        futures.add(sfuture);
      }

      // 这里将集合中的若干ListenableFuture形成一个新的ListenableFuture
      // 目的是为了异步阻塞，直到所有的ListenableFuture都得到结果才继续当前线程
      // 这里的时间取的是所有任务中用时最长的一个
      ListenableFuture<List<String>> allAsList = Futures.allAsList(futures);
      allAsList.get();
      LOG.info("All sub-task are finished.");
    } catch (Exception ignored) {
    }
  }

  @Test
  public void testCompletableFuture() throws Exception {
    ...
  }
}
```

根据测试输出结果，来验证其中的执行顺序，是不是我们预期的那样。

```
2018-05-19 11:06:34,870 [pool-1-thread-1] INFO  records.TestFutures (TestFutures.java:call(53)) - Finishing sl
eeping task0: 1526699194868
2018-05-19 11:06:34,874 [pool-1-thread-2] INFO  records.TestFutures (TestFutures.java:run(60)) - Listener be t
riggered for task0.
2018-05-19 11:06:34,896 [main] INFO  records.TestFutures (TestFutures.java:onSuccess(66)) - Add result value i
nto value list 1526699194868.
2018-05-19 11:06:34,924 [main] INFO  records.TestFutures (TestFutures.java:testListenableFuture(84)) - All sub
-task are finished.
```

CompletableFuture

我们再来看看CompletableFuture的使用，这个是在JDK8中开始引入的，这个在一定程度上与ListenableFuture非常类似。比如说ListenableFuture的listener监听回调，在这个类中，相当于thenRun或者whneComplete操作原语。CompletableFuture提供的API其实有很多，从大的方向上来划分的话，有下面几类：

```
public static CompletableFuture<Void>     runAsync(Runnable runnable)
public static CompletableFuture<Void>     runAsync(Runnable runnable, Executor executor)
public static <U> CompletableFuture<U>  supplyAsync(Supplier<U> supplier)
public static <U> CompletableFuture<U>  supplyAsync(Supplier<U> supplier, Executor executor)
```

注意到这里，runAsync是不带类型返回的，Void，而supplyAsync API需要传入类型的，整型，字符串或者其它，然后是否需要在额外的线程池里执行这些Async操作，如果没有指定，会默认在ForkJoinPool提供的common pool里跑。

同样笔者也写了一个简单的demo程序，

```
package

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.Executors;

import org.junit.Test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.google.common.util.concurrent.FutureCallback;
import com.google.common.util.concurrent.Futures;
import com.google.common.util.concurrent.ListenableFuture;
import com.google.common.util.concurrent.ListeningExecutorService;
import com.google.common.util.concurrent.MoreExecutors;

/** * The unit test for ListenableFuture/CompletableFuture. * Created by yiqun01.lin * on 2018/5/3. */
public class TestFutures {
  //线程池中线程个数
  private static final int POOL_SIZE = 50;
  //带有回调机制的线程池
  private static final ListeningExecutorService service = MoreExecutors
          .listeningDecorator(Executors.newFixedThreadPool(POOL_SIZE));

  private static Logger LOG = LoggerFactory.getLogger(TestFutures.class);

  @Test
  public void testListenableFuture() {
    ...
  }

  @Test
  public void testCompletableFuture() throws Exception {
    // case1: supplyAsync
    CompletableFuture<String> future = CompletableFuture.supplyAsync(() -> {
      LOG.info("Run supplyAsync.");
      return "Return result of Supply Async";
    });

    // case2: thenRun，与supplyAsync同线程
    future.thenRun(new Runnable() {

      @Override
      public void run() {
        LOG.info("Run action.");
      }
    });

    // case2: thenRunAsync，另启动线程执行
    future.thenRunAsync(new Runnable() {

      @Override
      public void run() {
        LOG.info("Run async action.");
      }
```

```
    });
    // 主动触发Complete结束方法
    // future.complete("Manual complete value.");
    future.whenComplete((v, e) -> {
      LOG.info("WhenComplete value: " + v);
      LOG.info("WhenComplete exception: " + e);
    });
    CompletableFuture<Void> future2 = CompletableFuture.runAsync(() -> {
      LOG.info("Return result of Run Async.");
    });

    CompletableFuture<String> future3 = CompletableFuture.supplyAsync(() -> {
      return "hello";
    });
    CompletableFuture<String> future4 = CompletableFuture.supplyAsync(() -> {
      return "world";
    });
    CompletableFuture<String> f = future3.thenCombine(future4,
        (x, y) -> x + "-" + y);
    LOG.info(f.get());
  }
}
```

测试输出结果:

```
2018-05-19 11:16:36,358 [ForkJoinPool.commonPool-worker-1] INFO  records.TestFutures (TestFutures.java:lambda
$0(93)) - Run supplyAsync.
2018-05-19 11:16:36,381 [main] INFO  records.TestFutures (TestFutures.java:run(102)) - Run action.
2018-05-19 11:16:36,393 [ForkJoinPool.commonPool-worker-1] INFO  records.TestFutures (TestFutures.java:run(11
1)) - Run async action.
2018-05-19 11:16:36,394 [main] INFO  records.TestFutures (TestFutures.java:lambda$1(118)) - WhenComplete value
: Return result of Supply Async
2018-05-19 11:16:36,394 [main] INFO  records.TestFutures (TestFutures.java:lambda$1(119)) - WhenComplete excep
tion: null
2018-05-19 11:16:36,396 [ForkJoinPool.commonPool-worker-1] INFO  records.TestFutures (TestFutures.java:lambda
$2(122)) - Return result of Run Async.
2018-05-19 11:16:36,397 [main] INFO  records.TestFutures (TestFutures.java:testCompletableFuture(133)) - hello
-world
```

这些API使用起来还是非常灵活的，大家可以自行本地继续调试调试，包括哪些是阻塞执行的，哪些是异步的，哪些是需要额外开线程执行的等等。

相关文章
- 1. 简单小结 (http://www.voidcn.com/article/p-qyybsgqt-ez.html)

- 2. ListenableFuture (http://www.voidcn.com/article/p-smjzxmfr-bow.html)
- 3. CompletableFuture (http://www.voidcn.com/article/p-zysexeyt-np.html)
- 4. Guava ListenableFuture 小试牛刀 (http://www.voidcn.com/article/p-wtfqbcnl-bu.html)
- 5. Spring.xml 简单小结 (http://www.voidcn.com/article/p-xfbolneb-nk.html)
- 6. XQuery简单小结 (http://www.voidcn.com/article/p-stmrlldx-bg.html)
- 7. jQuery 简单小结 (http://www.voidcn.com/article/p-ueqrkytt-d.html)
- 8. const简单小结 (http://www.voidcn.com/article/p-uhcrbpgh-es.html)
- 9. TURN简单小结 (http://www.voidcn.com/article/p-qhzgbrdb-vw.html)
- 10. ICE简单小结 (http://www.voidcn.com/article/p-naqlnbla-vw.html)
- 更多相关文章... (http://www.voidcn.com/relative/p-hyatqavs-bru.html)

相关标签/搜索

ListenableFuture (http://www.voidcn.com/tag/ListenableFuture) CompletableFuture (http://www.voidcn.com/tag/CompletableFuture) 简单总结 (http://www.voidcn.com/tag/%E7%AE%80%E5%8D%95%E6%80%BB%E7%BB%93) 简单小例 (http://www.voidcn.com/tag/%E7%AE%80%E5%8D%95%E5%B0%8F%E4%BE%8B) 简简单单 (http://www.voidcn.com/tag/%E7%AE%80%E7%AE%80%E5%8D%95%E5%8D%95) jQuery简单总结 (http://www.voidcn.com/tag/jQuery%E7%AE%80%E5%8D%95%E6%80%BB%E7%BB%93) JSON简单总结 (http://www.voidcn.com/tag/JSON%E7%AE%80%E5%8D%95%E6%80%BB%E7%BB%93) 简单小命令 (http://www.voidcn.com/tag/%E7%AE%80%E5%8D%95%E5%B0%8F%E5%91%BD%E4%BB%A4) 结账和日结账单 (http://www.voidcn.com/tag/%E7%BB%93%E8%B4%A6%E5%92%8C%E6%97%A5%E7%BB%93%E8%B4%A6%E5%8D%95) 简单 (http://www.voidcn.com/tag/%E7%AE%80%E5%8D%95) 简简单单 (http://www.voidcn.com/cata/2688121) 简单的求和 (http://www.voidcn.com/cata/%25e7%25ae%2580%25e5%258d%2595%25e7%259a%2584%25e6%25b1%2582%25e5%2592%258c) 简单 (http://www.voidcn.com/cata/1557817) 简单 (http://www.voidcn.com/cata/1185052) 简单 (http://www.voidcn.com/cata/3103569) 简单 (http://www.voidcn.com/cata/phjszcgi) 简单 (http://www.voidcn.com/cata/1118185) 简单 (http://www.voidcn.com/cata/1294873) 简单 (http://www.voidcn.com/cata/865709) 简单 (http://www.voidcn.com/cata/1264723) zookeeper 程序结构和简单示例 (http://www.voidcn.com/search/pvpmkt) ListenableFuture in Guava 和 SettableFuture (http://www.voidcn.com/search/cofwtr) pygame简单小游戏 (http://www.voidcn.com/search/hdfgdt) CompletableFuture 应用 (http://www.voidcn.com/search/ddabzw) Tablayout和Viewpager简单运用 (http://www.voidcn.com/search/kbosys) tag和untag简单理解 (http://www.voidcn.com/search/dbhcdf) ListenableFuture isDone (http://www.voidcn.com/search/covvhh) mdat的结构非常简单 (http://www.voidcn.com/search/afosyi) ACdream 简单数据结构 专题 (http://www.voidcn.com/search/bfupse) listenablefuture 应用 (http://www.voidcn.com/search/pyirrt)

每日一句

每一个你不满意的现在，都有一个你没有努力的曾经。

最新文章

- 1. python 图像上写中文字体 (http://www.voidcn.com/article/p-mbzwetow-bza.html)
- 2. 二、二进制 (http://www.voidcn.com/article/p-xjyfgqsk-bza.html)
- 3. 匿名函数和闭包 (http://www.voidcn.com/article/p-vvhwcjvh-bza.html)
- 4. 206.反转链表 (http://www.voidcn.com/article/p-fwmablpj-bza.html)
- 5. 树莓派4B更换国内源 (http://www.voidcn.com/article/p-wedaltxv-bza.html)
- 6. Example15.闭包 (http://www.voidcn.com/article/p-kxemxbzk-bza.html)
- 7. php访问url的四种方式 (http://www.voidcn.com/article/p-wdwrrgmi-bza.html)
- 8. oj随笔 (http://www.voidcn.com/article/p-ugrvtejl-bza.html)
- 9. 847. 图中点的层次 (http://www.voidcn.com/article/p-mdyjzoha-bza.html)
- 10. 阿里java开发规范---日志规约，单元测试，安全规约 (http://www.voidcn.com/article/p-hffmtpxd-bza.html)

公众号推荐 (/contact)

一个历史类的公众号，欢迎关注

程序园

(http://www.voidcn.com/)

本站公众号 (/contact)

欢迎关注本站公众号,获取更多程序园信息

相关文章

- 1. 简单小结 (http://www.voidcn.com/article/p-qyybsgqt-ez.html)
- 2. ListenableFuture (http://www.voidcn.com/article/p-smjzxmfr-bow.html)
- 3. CompletableFuture (http://www.voidcn.com/article/p-zysexeyt-np.html)
- 4. Guava ListenableFuture 小试牛刀 (http://www.voidcn.com/article/p-wtfqbcnl-bu.html)
- 5. Serialization 简单小结 (http://www.voidcn.com/article/p-xfbolneb-nk.html)
- 6. XQuery简单小结 (http://www.voidcn.com/article/p-stmrlldx-bg.html)
- 7. jQuery 简单小结 (http://www.voidcn.com/article/p-ueqrkytt-d.html)
- 8. const简单小结 (http://www.voidcn.com/article/p-uhcrbpgh-es.html)
- 9. TURN简单小结 (http://www.voidcn.com/article/p-qhzgbrdb-vw.html)
- 10. ICE简单小结 (http://www.voidcn.com/article/p-naqlnbla-vw.html)

>>更多相关文章<< (http://www.voidcn.com/relative/p-hyatqavs-bru.html)