链滴

实例带你获取多线程 Thread 的返回值之 (壹) - Callable 的运行

作者：adlered

原文链接：https://ld246.com/article/1552640897014

来源网站：链滴

前言

阅读本篇文章，你需要先理解以下知识：

- 多线程Thread的基本使用 （ 点我跳转）
- extends和implements
- 重写Override
- try catch错误处理
- Java基础知识

回顾

回顾一下我们学过的Thread多线程，是继承了一个Thread类，然后调用run()方法来执行定义的类。

那么问题来了，有时候我们需要多线程传回一个计算的值给我们，Callable就很好地解决了我们的需。

拷贝

在你的IDE中新建一个项目或一个类，并将类命名为CallableDemo，然后拷贝下面这些代码：

```
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.FutureTask;

public class CallableDemo {
    public static void main(String[] args) {
        //实例化当前类
        CallableDemo callableDemo = new CallableDemo();
        //调用动态类
        callableDemo.run();
    }

    public void run() {
        ThreadC threadC = new ThreadC();
        //使用FutureTask接收Callable运行的结果
        FutureTask futureTask = new FutureTask(threadC);
```

```
        //启动线程
        new Thread(futureTask).start();
        //获取返回结果
        try {
            Object ret = futureTask.get();
            System.out.println("获得返回结果: " + ret);
        } catch (InterruptedException | ExecutionException e) {
            System.out.println("获取结果出错");
        }
    }
}

class ThreadC implements Callable {
    @Override
    public Object call() throws Exception {
        String a = "WeChat: 1101635162";
        return a;
    }
}
```

让我们**无视掉主方法**，它用于实例化当前类并调用动态的run()方法。将注意力转移到run()方法和ThreadC类。

继承了Callable接口的类

```
class ThreadC implements Callable {
    @Override
    public Object call() throws Exception {
        String a = "WeChat: 1101635162";
        return a;
    }
}
```

可以看到，我们的ThreadC类继承了Callable接口，且重写了Callable接口的call()方法，使其返回一字符串。

调用方法

```
    public void run() {
        ThreadC threadC = new ThreadC();
        //使用FutureTask接收Callable运行的结果
        FutureTask futureTask = new FutureTask(threadC);
        //启动线程
        new Thread(futureTask).start();
        //获取返回结果
        try {
            Object ret = futureTask.get();
            System.out.println("获得返回结果: " + ret);
        } catch (InterruptedException | ExecutionException e) {
            System.out.println("获取结果出错");
        }
    }
```

在该方法中，我们实例化了ThreadC。与普通的Thread不同的是我们使用了FutureTask类。该类用接收Callable的返回结果。

剩下的代码便很好理解了，获取结果并输出。

运行!

现在让我们运行该类，并查看结果：

获得返回结果: WeChat: 1101635162

后语

本篇文章描述了简单的Callable使用方法，下一章我们将把Callable套用到线程池当中来。（点我跳转）