二

48 Callable 和 Runnable 的不同?

你好，欢迎来到第 48 课时，在本课时我们将讲解 Callable 和 Runnable 的不同。

为什么需要 Callable？Runnable 的缺陷

先来看一下，为什么需要 Callable？要想回答这个问题，我们先来看看现有的 Runnable 有哪些缺陷?

不能返回一个返回值

第一个缺陷，对于 Runnable 而言，它不能返回一个返回值，虽然可以利用其他的一些办法，比如在 Runnable 方法中写入日志文件或者修改某个共享的对象的办法，来达到保存线程执行结果的目的，但这种解决问题的行为千曲百折，属于曲线救国，效率着实不高。

实际上，在很多情况下执行一个子线程时，我们都希望能得到执行的任务的结果，也就是说，我们是需要得到返回值的，比如请求网络、查询数据库等。可是 Runnable 不能返回一个返回值，这是它第一个非常严重的缺陷。

不能抛出 checked Exception

第二个缺陷就是不能抛出 checked Exception，如下面这段代码所示:

```
public class RunThrowException {

    /**
     * 普通方法内可以 throw 异常，并在方法签名上声明 throws
     */
    public void normalMethod() throws Exception {
        throw new IOException();
    }
```

```
    Runnable runnable = new Runnable() {
        /**
         *  run方法上无法声明 throws 异常，且run方法内无法 throw 出 checked Exception，
         */
        @Override
        public void run() {
            try {
                throw new IOException();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

在这段代码中，有两个方法，第一个方法是一个普通的方法，叫作 normalMethod，可以看到，在它的方法签名中有 throws Exception，并且在它的方法内也 throw 了一个 new IOException()。

然后在下面的的代码中，我们新建了一个 Runnable 对象，同时重写了它的 run 方法，我们没有办法在这个 run 方法的方法签名上声明 throws 一个异常出来。同时，在这个 run 方法里面也没办法 throw 一个 checked Exception，除非如代码所示，用 try catch 包裹起来，但是如果不用 try catch 是做不到的。

这就是对于 Runnable 而言的两个重大缺陷。

为什么有这样的缺陷

为什么有这样的缺陷呢？我们来看一下 Runnable 接口的定义：

```
public interface Runnable {
    public abstract void run();
}
```

代码比较短小，Runnable 是一个 interface，并且里面只有一个方法，叫作 public abstract void run()。这个方法已经规定了 run() 方法的返回类型是 void，而且这个方法没有声明抛出任何异常。所以，当实现并重写这个方法时，我们既不能改返回值类型，也不能更改对于异常抛出的描述，因为在实现方法的时候，语法规定是不允许对这些内容进行修改的。

回顾课程之前小节的众多代码，从来没有出现过可以在 run 方法中返回一个返回值这样的情况。

Runnable 为什么设计成这样

我们再深入思考一层，为什么 Java 要把它设计成这个样子呢?

假设 run() 方法可以返回返回值，或者可以抛出异常，也无济于事，因为我们并没有办法在外层捕获并处理，这是因为调用 run() 方法的类（比如 Thread 类和线程池）是 Java 直接提供的，而不是我们编写的。

所以就算它能有一个返回值，我们也很难把这个返回值利用到，如果真的想弥补 Runnable 的这两个缺陷，可以用下面的补救措施——使用 Callable。

Callable 接口

Callable 是一个类似于 Runnable 的接口，实现 Callable 接口的类和实现 Runnable 接口的类都是可以被其他线程执行的任务。 我们看一下 Callable 的源码：

```
public interface Callable<V> {

    V call() throws Exception;

}
```

可以看出它也是一个 interface，并且它的 call 方法中已经声明了 throws Exception，前面还有一个 V 泛型的返回值，这就和之前的 Runnable 有很大的区别。实现 Callable 接口，就要实现 call 方法，这个方法的返回值是泛型 V，如果把 call 中计算得到的结果放到这个对象中，就可以利用 call 方法的返回值来获得子线程的执行结果了。

Callable 和 Runnable 的不同之处

最后总结一下 Callable 和 Runnable 的不同之处：

- **方法名**，Callable 规定的执行方法是 call()，而 Runnable 规定的执行方法是 run()；

- **返回值**，Callable 的任务执行后有返回值，而 Runnable 的任务执行后是没有返回值的；
- **抛出异常**，call() 方法可抛出异常，而 run() 方法是不能抛出受检查异常的；
- 和 Callable 配合的有一个 Future 类，通过 Future 可以了解任务执行情况，或者取消任务的执行，还可获取任务执行的结果，这些功能都是 Runnable 做不到的，Callable 的功能要比 Runnable 强大。

以上就是本课时的内容了。首先介绍了 Runnable 的两个缺陷，第一个是没有返回值，第二个是不能抛出受检查异常；然后分析了为什么会有这样的缺陷，以及为什么设计成这样；接下来分析了 Callable 接口，并且把 Callable 接口和 Runnable 接口的区别进行了对比和总结。

上一页 下一页