java和android子线程中抛出异常的研究

jasonhc 关注

0.104 2018.02.02 17:10:54 字数 335 阅读 2,123

Java语言

java语言里, 子线程中抛出的RuntimeException, 只会使得子线程结束运行, 不会影响主线程的执行, 进程不会因此crash. 主线程无法catch子线程抛出的exception.

如何处理子线程中抛出的RuntimeException, 有3种方式:

1. 使用interface Thread.UncaughtExceptionHandler, 在调用Thread.start()方法之前, 调用Thread实例的setUncaughtExceptionHandler()方法, 实现interface Thread.UncaughtExceptionHandler. 示例代码如下:

```java
Thread thread = new Thread(mThrowExceptionThreadGroup, new Runnable() {
…...
});
thread.setUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
    @Override
    public void uncaughtException(Thread thread, Throwable throwable) {
    }
});
thread.start();
```

2. 使用ThreadGroup.uncaughtException(), 创建ThreadGroup实例时, 覆写(override)uncaughtException()方法; 创建Thread实例时, 调用构造方法Thread(ThreadGroup , Runnable), 传入刚创建的ThreadGroup实例. 示例代码如下:

```
private final ThreadGroup mThrowExceptionThreadGroup = new ThreadGroup("ThrowException
    @Override
    public void uncaughtException(Thread thread, Throwable e) {
    }
};
Thread thread = new Thread(mThrowExceptionThreadGroup, new Runnable() {
});
thread.start()
```

注意: 上面2种处理方法, 处理子线程抛出的Exception的代码还是运行在子线程里, 如果在uncaughtException()中throw Exception, 进程还是不会crash.

3. 如果使用ExecutorService, Future运行子线程, Future.get()方法会抛出ExecutionException(Checked exception), 子线程里产生的RuntimeException会被包含在ExecutionException实例里抛出给调用者. 示例代码如下:

```
ExecutorService executorService = Executors.newFixedThreadPool(8);
Future future = executorService.submit(new Runnable() {
    @Override
    public void run() {
        int a = 5 / 0;
    }
});

try {
    future.get();
} catch (InterruptedException | ExecutionException e) {
    System.out.println(String.format("handle exception in child thread: %s", e.getCaus
} finally {
    executorService.shutdown();
}
```

Android环境下

在Android环境下, 情况有所不同. 当子线程抛出RuntimeException时, 进程会crash.
Android framework的处理如下:

在RuntimeInit.commonInit()方法内, 调用Thread.setDefaultUncaughtExceptionHandler(new UncaughtHandler()); 设置Thread.UncaughtExceptionHandler.

/*

* set handlers; these apply to all threads in the VM. Apps can replace

* the default handler, but not the pre handler.

*/

LoggingHandler loggingHandler = new LoggingHandler();

RuntimeHooks.setUncaughtExceptionPreHandler(loggingHandler);

Thread.setDefaultUncaughtExceptionHandler(new KillApplicationHandler(loggingHandler));