Scala 正则表达式 Scala 提取器(Extractor)

Scala 异常处理

Scala 的异常处理和其它语言比如 Java 类似。

Scala 的方法可以通过抛出异常的方法的方式来终止相关代码的运行，不必通过返回值。

抛出异常

Scala 抛出异常的方法和 Java一样，使用 throw 方法，例如，抛出一个新的参数异常：

```
throw new IllegalArgumentException
```

捕获异常

异常捕捉的机制与其他语言中一样，如果有异常发生，catch字句是按次序捕捉的。因此，在catch字句中，越具体的异常越要靠前，越普遍的异常越靠后。 如果抛出的异常不在catch字句中，该异常则无法处理，会被升级到调用者处。

捕捉异常的catch子句，语法与其他语言中不太一样。在Scala里，借用了模式匹配的思想来做异常的匹配，因此，在catch的代码里，是一系列case字句，如下例所示：

```
import java.io.FileReader
import java.io.FileNotFoundException
import java.io.IOException

object Test {
   def main(args: Array[String]) {
      try {
         val f = new FileReader("input.txt")
      } catch {
         case ex: FileNotFoundException =>{
            println("Missing file exception")
         }
         case ex: IOException => {
            println("IO Exception")
         }
      }
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
Missing file exception
```

catch字句里的内容跟match里的case是完全一样的。由于异常捕捉是按次序，如果最普遍的异常，Throwable，写在最前面，则在它后面的case都捕捉不到，因此需要将它写在最后面。

finally 语句

finally 语句用于执行不管是正常处理还是有异常发生时都需要执行的步骤，实例如下：

```
import java.io.FileReader
import java.io.FileNotFoundException
import java.io.IOException

object Test {
   def main(args: Array[String]) {
      try {
         val f = new FileReader("input.txt")
      } catch {
         case ex: FileNotFoundException => {
            println("Missing file exception")
         }
         case ex: IOException => {
            println("IO Exception")
         }
      } finally {
         println("Exiting finally...")
      }
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
Missing file exception
Exiting finally...
```

← Scala 正则表达式 Scala 提取器(Extractor) →

点我分享笔记