Scala 异常处理 Scala 文件 I/O

Scala 提取器(Extractor)

提取器是从传递给它的对象中提取出构造该对象的参数。

Scala 标准库包含了一些预定义的提取器，我们会大致的了解一下它们。

Scala 提取器是一个带有unapply方法的对象。unapply方法算是apply方法的反向操作：unapply接受一个对象，然后从对象中提取值，提取的值通常是用来构造该对象的值。

以下实例演示了邮件地址的提取器对象：

```
object Test {
   def main(args: Array[String]) {

      println ("Apply 方法 : " + apply("Zara", "gmail.com"));
      println ("Unapply 方法 : " + unapply("Zara@gmail.com"));
      println ("Unapply 方法 : " + unapply("Zara Ali"));

   }
   // 注入方法 (可选)
   def apply(user: String, domain: String) = {
      user +"@"+ domain
   }

   // 提取方法（必选）
   def unapply(str: String): Option[(String, String)] = {
      val parts = str split "@"
      if (parts.length == 2){
         Some(parts(0), parts(1))
      }else{
         None
      }
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
Apply 方法 : Zara@gmail.com
Unapply 方法 : Some((Zara,gmail.com))
Unapply 方法 : None
```

以上对象定义了两个方法： **apply** 和 **unapply** 方法。通过 apply 方法我们无需使用 new 操作就可以创建对象。所以你可以通过语句 Test("Zara", "gmail.com") 来构造一个字符串 "Zara@gmail.com"。

unapply方法算是apply方法的反向操作：unapply接受一个对象，然后从对象中提取值，提取的值通常是用来构造该对象的值。

实例中我们使用 Unapply 方法从对象中提取用户名和邮件地址的后缀。

实例中 unapply 方法在传入的字符串不是邮箱地址时返回 None。代码演示如下：

```
unapply("Zara@gmail.com") 相等于 Some("Zara", "gmail.com")
unapply("Zara Ali") 相等于 None
```

提取器使用模式匹配

在我们实例化一个类的时，可以带上0个或者多个的参数，编译器在实例化的时会调用 apply 方法。我们可以在类和对象中都定义 apply 方法。

就像我们之前提到过的，unapply 用于提取我们指定查找的值，它与 apply 的操作相反。 当我们在提取器对象中使用 match 语句是，unapply 将自动执行，如下所示：

```
object Test {
   def main(args: Array[String]) {

      val x = Test(5)
      println(x)

      x match
      {
         case Test(num) => println(x + " 是 " + num + " 的两倍! ")
         //unapply 被调用
         case _ => println("无法计算")
      }

   }
   def apply(x: Int) = x*2
   def unapply(z: Int): Option[Int] = if (z%2==0) Some(z/2) else None
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
10
10 是 5 的两倍!
```

← Scala 异常处理 Scala 文件 I/O →

点我分享笔记