Scala 模式匹配 Scala 异常处理

Scala 正则表达式

Scala 通过 scala.util.matching 包中的 **Regex** 类来支持正则表达式。以下实例演示了使用正则表达式查找单词 **Scala** :

```
import scala.util.matching.Regex

object Test {
   def main(args: Array[String]) {
      val pattern = "Scala".r
      val str = "Scala is Scalable and cool"

      println(pattern findFirstIn str)
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
Some(Scala)
```

实例中使用 String 类的 r() 方法构造了一个Regex对象。

然后使用 findFirstIn 方法找到首个匹配项。

如果需要查看所有的匹配项可以使用 findAllIn 方法。

你可以使用 mkString( ) 方法来连接正则表达式匹配结果的字符串，并可以使用管道(|)来设置不同的模式：

```
import scala.util.matching.Regex

object Test {
   def main(args: Array[String]) {
      val pattern = new Regex("(S|s)cala")  // 首字母可以是大写 S 或小写 s
      val str = "Scala is scalable and cool"

      println((pattern findAllIn str).mkString(","))   // 使用逗号 , 连接返回结果
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
```

```
Scala,scala
```

如果你需要将匹配的文本替换为指定的关键词，可以使用 **replaceFirstIn( )** 方法来替换第一个匹配项，使用 **replaceAllIn( )** 方法替换所有匹配项，实例如下:

```
object Test {
   def main(args: Array[String]) {
      val pattern = "(S|s)cala".r
      val str = "Scala is scalable and cool"

      println(pattern replaceFirstIn(str, "Java"))
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
Java is scalable and cool
```

正则表达式

Scala 的正则表达式继承了 Java 的语法规则，Java 则大部分使用了 Perl 语言的规则。

下表我们给出了常用的一些正则表达式规则：

表达式	匹配规则
^	匹配输入字符串开始的位置。
$	匹配输入字符串结尾的位置。
.	匹配除"\r\n"之外的任何单个字符。
[...]	字符集。匹配包含的任一字符。例如，"[abc]"匹配"plain"中的"a"。
[^...]	反向字符集。匹配未包含的任何字符。例如，"[^abc]"匹配"plain"中"p"，"l"，"i"，"n"。
\\A	匹配输入字符串开始的位置（无多行支持）
\\z	字符串结尾(类似$，但不受处理多行选项的影响)
\\Z	字符串结尾或行尾(不受处理多行选项的影响)
re*	重复零次或更多次
re+	重复一次或更多次

re?	重复零次或一次
re{ n}	重复n次
re{ n,}	
re{ n, m}	重复n到m次
a|b	匹配 a 或者 b
(re)	匹配 re,并捕获文本到自动命名的组里
(?: re)	匹配 re,不捕获匹配的文本，也不给此分组分配组号
(?> re)	贪婪子表达式
\\w	匹配字母或数字或下划线或汉字
\\W	匹配任意不是字母，数字，下划线，汉字的字符
\\s	匹配任意的空白符,相等于 [\t\n\r\f]
\\S	匹配任意不是空白符的字符
\\d	匹配数字，类似 [0-9]
\\D	匹配任意非数字的字符
\\G	当前搜索的开头
\\n	换行符
\\b	通常是单词分界位置，但如果在字符类里使用代表退格
\\B	匹配不是单词开头或结束的位置
\\t	制表符
\\Q	开始引号：**\Q(a+b)*3\E** 可匹配文本 "(a+b)*3"。
\\E	结束引号：**\Q(a+b)*3\E** 可匹配文本 "(a+b)*3"。

正则表达式实例

实例	描述
.	匹配除"\r\n"之外的任何单个字符。

[Rr]uby	匹配 "Ruby" 或 "ruby"
rub[ye]	匹配 "ruby" 或 "rube"
[aeiou]	匹配小写字母 ：aeiou
[0-9]	匹配任何数字，类似 [0123456789]
[a-z]	匹配任何 ASCII 小写字母
[A-Z]	匹配任何 ASCII 大写字母
[a-zA-Z0-9]	匹配数字，大小写字母
[^aeiou]	匹配除了 aeiou 其他字符
[^0-9]	匹配除了数字的其他字符
\\d	匹配数字，类似: [0-9]
\\D	匹配非数字，类似: [^0-9]
\\s	匹配空格，类似: [\t\r\n\f]
\\S	匹配非空格，类似: [^ \t\r\n\f]
\\w	匹配字母，数字，下划线，类似: [A-Za-z0-9_]
\\W	匹配非字母，数字，下划线，类似: [^A-Za-z0-9_]
ruby?	匹配 "rub" 或 "ruby": y 是可选的
ruby*	匹配 "rub" 加上 0 个或多个的 y。
ruby+	匹配 "rub" 加上 1 个或多个的 y。
\\d{3}	刚好匹配 3 个数字。
\\d{3,}	匹配 3 个或多个数字。
\\d{3,5}	匹配 3 个、4 个或 5 个数字。
\\D\\d+	无分组： + 重复 \d
(\\D\\d)+/	分组： + 重复 \D\d 对
([Rr]uby(,)?)+	匹配 "Ruby"、"Ruby, ruby, ruby"，等等

注意上表中的每个字符使用了两个反斜线。这是因为在 Java 和 Scala 中字符串中的反斜线是转义字符。所以如果你要输出 .\. ，你需要在字符串中写成 .\\. 来获取一个反斜线。查看以下实例：

```
import scala.util.matching.Regex

object Test {
   def main(args: Array[String]) {
      val pattern = new Regex("abl[ae]\\d+")
      val str = "ablaw is able1 and cool"

      println((pattern findAllIn str).mkString(","))
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
able1
```

← Scala 模式匹配 Scala 异常处理 →