for循环

2020年2月29日 14:18

基本介绍

Scala也为for 循环这一常见的控制结构提供了非常多的特性，这些for 循环的特性被称为**for 推导式**（for comprehension）或**for 表达式**（for expression）

```
object ForDemo01 {
  def main(args: Array[String]): Unit = {
    //输出10句 "hello,尚硅谷!"
    val start = 1
    val end = 10
    //说明
    //1. start 从哪个数开始循环
    //2. to 是关键字
    //3. end 循环结束的值
    //4. start to end 表示前后闭合
    for (i <- start to end) {
      println("你好，尚硅谷" + i)
    }

    //说明for 这种推导时，也可以直接对集合进行遍历
    var list = List("hello", 10, 30, "tom")
    for (item <- list) {
      println("item=" + item)
    }
  }
```

范围数据循环方式2

➢ 基本案例

```
for(i <- 1 until 3) {
  print(i + " ")
}
println()
```

说明:

1) 这种方式和前面的区别在于 i 是从**1 到 3-1**

2) **前闭合后开**的范围,和java的arr.length()类似

➢ 输出10句 "hello,尚硅谷!"

注：for中的迭代变量默认是val类型的

循环控制

• for循环控制

循环守卫

➢ 基本案例

```
for(i <- 1 to 3 if i != 2) {
 print(i + " ")
}
println()
```

➢ 基本案例说明

1) 循环守卫，即循环保护式（也称条件判断式，守卫）。保护式为true则进入循环体内部，为false则跳过，类似于continue

2) 上面的代码等价

```
for (i <- 1 to 3) {
        if (i != 2) {
                println(i+"")
        }
}
```

• for循环控制

引入变量

➢ 基本案例

```
for(i <- 1 to 3; j = 4 - i) {
  print(j + " ")
}
```

➢ 对基本案例说明

1) 没有关键字，所以范围后一定要加；来隔断逻辑

2) 上面的代码等价

```
for ( i <- 1 to 3) {
        val j = 4 –i
        print(j+"")
}
```

• for循环控制

嵌套循环

➢ 基本案例

```
for(i <- 1 to 3; j <- 1 to 3) {
  println(" i =" + i + " j = " + j)
}
```

➢ 对基本案例说明

1) 没有关键字，所以范围后一定要加；来隔断逻辑

2) 上面的代码等价

```
for ( i <- 1 to 3) {
        for ( j <- 1to 3){
                println(i + " " + j + " ")
        }
}
```

- **for循环控制**

循环返回值

➢ 基本案例

```
val res = for(i <- 1 to 10) yield i
println(res)
```

➢ 对基本案例说明

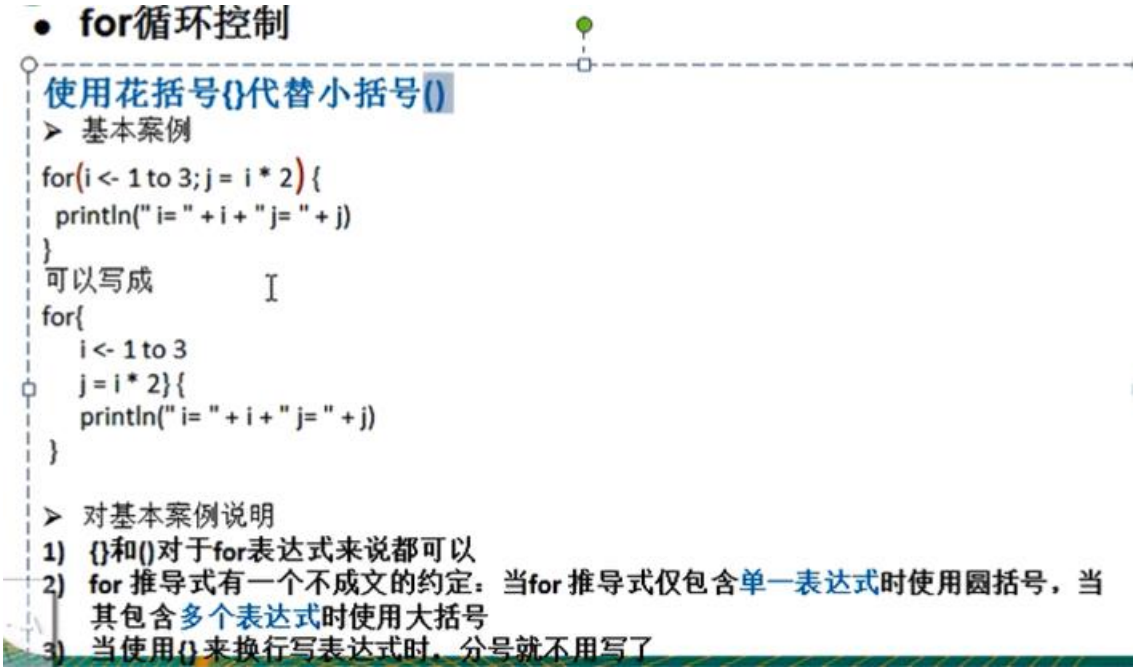
1) 将遍历过程中处理的结果返回到一个新Vector集合中，使用yield关键字

```
val res = for(i <- 1 to 10) yield {
  if (i % 2 == 0) {
    i
  }else {
    "不是偶数"
  }
}
println(res)
```

```
D:\program\jdk8\bin\java ...
Vector(不是偶数, 2, 不是偶数, 4, 不是偶数, 6, 不是偶数, 8, 不是偶数, 10)

Process finished with exit code 0
```

```
val ss=for(i <- 1 to 5;j <- 1 to 5) yield {
  "i="+i+"\tj="+j+"---------------"
}
for(s <- ss)
  println(s)
println(ss.isInstanceOf[Vector[String]])//true
```



步长控制：

```
for (i <- 1 to 10) {
  println("i=" + i)
}
//步长控制为2
println("-------------")
//Range(1,10,2)的对应的构建方法是
//def apply(start: Int, end: Int, step: Int): Range = new Range(start, end, step)
for (i <- Range(1, 10, 2)) {
  println("i=" + i)
}

//控制步长的第二种方式-for循环守卫
println()
for (i <- 1 to 10 if i % 2 == 1) {
  println("i=" + i)
}
```