* Scala中==,eq与equals的区别

根据官方API的定义：

final def ==(arg0: Any): Boolean

The expression x == that is equivalent to if (x ***eq*** null) that eq null else x.equals(that).

final def eq(arg0: AnyRef): Boolean

Tests whether the argument (that) is a reference to the receiver object (this).

def equals(arg0: Any): Boolean

The equality method for reference types.

简言之，equals方法是检查值是否相等，而eq方法检查的是引用是否相等。所以如果比较的对象是null那么==调用的是eq，不是null的情况调用的是equals。

看一个例子：

在Scala中为开发者提供了case class，默认实现了equals 和hashCode方法。至于case class我正以后讲解class的时候会讲解。为了方便我直接在REPL中演示

scala> case class Bread(brand:String, price:Int)

defined class Bread

scala> val b1 = Bread("BreadTalk", 50)

b1: Bread = Bread(BreadTalk,50)

scala> val b2 = Bread("BreadTalk", 50)

b2: Bread = Bread(BreadTalk,50)

scala> b1 eq b2 //即b1.eq(b2),判断b2和b1是否指向同一实例

res2: Boolean = false

scala> b1 equals b2 //即b1.equals(b2),类似java中的equals，会逐一判断其中的成员变量是否相等

res3: Boolean = true

创建两个Bread的实例，eq判断两个的引用是否相等，equals判断两个实例是否相等。

而对于Array或者Map对象不能简单点使用equals进行值比较，要通过sameElements方法，例如：

scala> val a1 = Array("x", "y")

a1: Array[String] = Array(x, y)

scala> val a2 = Array("x", "y")

a2: Array[String] = Array(x, y)

scala> a1 equals a2

res4: Boolean = false

scala> a1 eq a2

res5: Boolean = false

scala> a1 sameElements a2

res6: Boolean = true

scala> val m1 = Map(1->"x", 2->"y")

m1: scala.collection.immutable.Map[Int,String] = Map(1 -> x, 2 -> y)

scala> val m2 = Map(1->"x", 2->"y")

m2: scala.collection.immutable.Map[Int,String] = Map(1 -> x, 2 -> y)

scala> m1 sameElements m2

res7: Boolean = true

scala> val m3 = Map(1->"x", 2->"z")

m3: scala.collection.immutable.Map[Int,String] = Map(1 -> x, 2 -> z)

scala> m1 sameElements m3

res8: Boolean = false

* Tuples

A tuple value is formed by enclosing individual values in parentheses.

大概意思就是元组就是将各个独立的值用括号包含起来。

举个例子：

(1, 3.14, "Fred")这就是一个元组，它的类型是

Tuple3[Int, Double, java.lang.String]，前面的Tuple3代表有三个元素，中括号中表示元素的类型，也可以这么写(Int, Double, java.lang.String)

与Array,String不同，Tuple内部元素的位置是从1开始的，而不是0.

你可以直接从过模式匹配访问tuple内的元素，下面例子中的

**val** (first, second, third) = t

如果你不需要某个元素，你也可以通过\_来去掉某个元素

val (first1, second1, \_)

也可以通过tuple来返回不只一个值。

"New York".partition(\_.isUpper)

Partition是StringOps类的一个方法，它返回一个String的Tuple，包含满足条件的字符和不满足条件的字符。

DEMO，结合输出结果来看：

**def** main(args: Array[String]): Unit = {  
  
 **val** t = (1, 3.14, **"Fred"**) //Constructs a tuple  
  
 **val** sec = t.\_2 //you can access its components with \_1,\_2 method  
  
 *println*(**"sec: "** + sec)  
  
 **val** (first, second, third) = t // Sets first to 1, second to 3.14, third to "Fred  
  
 *println*(**"first: "** + first)  
 *println*(**"second: "** + second)  
 *println*(**"third: "** + third)  
  
 **val** (first1, second1, \_) = t  
 *println*(**"first1: "** + first1)  
 *println*(**"second1: "** + second1)  
  
 **val** pair = **"New York"**.partition(\_.isUpper) // Yields the pair ("NY", "ew ork")  
  
 *println*(**"uppercase: "** + pair.\_1)  
 *println*(**"not uppercase: "** + pair.\_2)  
}

输出结果：

sec: 3.14

first: 1

second: 3.14

third: Fred

first1: 1

second1: 3.14

uppercase: NY

not uppercase: ew ork

Process finished with exit code 0