Scala模式匹配泛型类型擦除问题

在Scala中一个很强大的功能就是模式匹配，本文并不打算介绍模式匹配的概念以及如何使用。本文的主要内容是讨论Scala模式匹配泛型类型擦除问题。先来看看泛型类型擦除是什么情况：

```
scala> def test(a:Any) = a match {
     |       case a :List[String] => println("iteblog is ok");
     |       case _ =>
     |}
```

按照代码的意思应该是匹配List[String]类型的数据，但是这个test(List("Iteblog")) 和test(List(2015))都是可以通过的。

```
scala> test(List("Iteblog"))
iteblog is ok

scala> test(List(2015))
iteblog is ok
```

这是因为类型被擦除了，你在编译这个代码的时候也会看到编译器给的提示：

```
<console>:14: warning: non-variable type argument String in type pattern List[String] is unchec
ked since it is eliminated by erasure
               case a :List[String] => println("iteblog is ok");
```

上面测试结果都得到了结果，虽然不是我们要的，但是并没有出现什么异常。下面的代码却会出现异常：

```
scala> case class Container[+A](value: A)
defined class Container

scala> val double = Container(3.3)
double: Container[Double] = Container(3.3)
```

```
scala> double match {
     |         case c: Container[String] => println(c.value.toUpperCase)
     |         case c: Container[Double] => println(math.sqrt(c.value))
     |         case _ => println("_")
     |   }
```

因为类型信息被编译器擦除了，所以你的double虽然是Container[Double]类型的，但是只能匹配到第一个case，而你想把一个double转换成String，所以会出现以下的异常：

```
java.lang.ClassCastException: java.lang.Double cannot be cast to java.lang.String
 at $iwC$$iwC$$iwC$$iwC$$iwC$$iwC.<init>(<console>:19)
 at $iwC$$iwC$$iwC$$iwC$$iwC.<init>(<console>:27)
 at $iwC$$iwC$$iwC$$iwC.<init>(<console>:29)
 at $iwC$$iwC$$iwC.<init>(<console>:31)
 at $iwC$$iwC.<init>(<console>:33)
 at $iwC.<init>(<console>:35)
 at <init>(<console>:37)
 at .<init>(<console>:41)
 at .<clinit>(<console>)
 at .<init>(<console>:7)
 at .<clinit>(<console>)
 at $print(<console>)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
 at java.lang.reflect.Method.invoke(Method.java:606)
 at org.apache.spark.repl.SparkIMain$ReadEvalPrint.call(SparkIMain.scala:1065)
 at org.apache.spark.repl.SparkIMain$Request.loadAndRun(SparkIMain.scala:1340)
 at org.apache.spark.repl.SparkIMain.loadAndRunReq$1(SparkIMain.scala:840)
 at org.apache.spark.repl.SparkIMain.interpret(SparkIMain.scala:871)
 at org.apache.spark.repl.SparkIMain.interpret(SparkIMain.scala:819)
 at org.apache.spark.repl.SparkILoop.reallyInterpret$1(SparkILoop.scala:857)
 at org.apache.spark.repl.SparkILoop.interpretStartingWith(SparkILoop.scala:902)
 at org.apache.spark.repl.SparkILoop.reallyInterpret$1(SparkILoop.scala:875)
 at org.apache.spark.repl.SparkILoop.interpretStartingWith(SparkILoop.scala:902)
 at org.apache.spark.repl.SparkILoop.reallyInterpret$1(SparkILoop.scala:875)
 at org.apache.spark.repl.SparkILoop.interpretStartingWith(SparkILoop.scala:902)
 at org.apache.spark.repl.SparkILoop.reallyInterpret$1(SparkILoop.scala:875)
 at org.apache.spark.repl.SparkILoop.interpretStartingWith(SparkILoop.scala:902)
 at org.apache.spark.repl.SparkILoop.command(SparkILoop.scala:814)
```

```
 at org.apache.spark.repl.SparkILoop.processLine$1(SparkILoop.scala:657)
 at org.apache.spark.repl.SparkILoop.innerLoop$1(SparkILoop.scala:665)
 at org.apache.spark.repl.SparkILoop.org$apache$spark$repl$SparkILoop$$loop(SparkILoop.scala:670)
 at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1.apply$mcZ$sp(SparkILoop.scala:997)
 at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1.apply(SparkILoop.scala:945)
 at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1.apply(SparkILoop.scala:945)
 at scala.tools.nsc.util.ScalaClassLoader$.savingContextLoader(ScalaClassLoader.scala:135)
 at org.apache.spark.repl.SparkILoop.org$apache$spark$repl$SparkILoop$$process(SparkILoop.scala:945)
 at org.apache.spark.repl.SparkILoop.process(SparkILoop.scala:1059)
 at org.apache.spark.repl.Main$.main(Main.scala:31)
 at org.apache.spark.repl.Main.main(Main.scala)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
 at java.lang.reflect.Method.invoke(Method.java:606)
 at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain(SparkSubmit.scala:673)
 at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:180)
 at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:205)
 at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:120)
 at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

那如何来处理这种问题呢？其实你可以用下面的代码解决：

```
def matchContainer[A: Manifest](c: Container[A]) = c match {
   case c: Container[String] if manifest <:< manifest[String] => println(c.value.toUpperCase)
   case c: Container[Double] if manifest <:< manifest[Double] => println(math.sqrt(c.value))
   case c: Container[_] => println("other")
}
```

它可以判断出c的类型，但是Manifest已经被标记为deprecated，我们可以使用TypeTag替代Manifest：

```
import reflect.runtime.universe._
def matchContainer[A: TypeTag](c: Container[A]) = c match {
```

```
  case c: Container[String] if typeOf[A] <:< typeOf[String] => println(c.value.toUpperCase)
  case c: Container[Double] if typeOf[A] <:< typeOf[Double] => println(math.sqrt(c.value))
  case c: Container[_] => println("other")
}
```

不过TypeTags 是线程不安全的，可以参见：http://docs.scala-lang.org/overviews/reflection/thread-safety.html。

如果在多线程环境下，我们不能使用上面的代码。我们可以用下面代码解决：

```
var container: Container[Any] = double
container match {
  case Container(x: String) => println("string")
  case Container(x: Double) => println("double")
  case _ => println("iteblog")
}
```

本文整理自：http://stackoverflow.com/questions/16056645/how-to-pattern-match-on-generic-type-in-scala/16057002#16057002