在 json4s 中自定义 CustomSerializer

到目前为止，Scala 环境下至少存在6种 Json 解析的类库，这里面不包括 Java 语言实现的 Json 类库。所有这些库都有一个非常相似的抽象语法树(AST)。而 json4s 项目旨在提供一个单一的 AST 树供其他 Scala 类库来使用。

json4s 的使用非常的简单，它可以将类直接转换成 json 格式输出，也支持将 json 格式的数据转换成 class 对象。对于 Scala 和 Java 常见的类型（比如String、Int、java.lang.Integer、java.lang.Long、java.lang.Boolean 等）都提供了响应的转换函数。比如下面我们直接将一个类转换成 json：

```
import org.json4s.JsonAST.{JNull, JString}
import org.json4s.{DefaultFormats, Extraction, Formats}
import org.json4s.jackson.JsonMethods.render
import org.json4s.jackson.JsonMethods.pretty

case class Person(name: String, age: Int)
implicit val formats: Formats = DefaultFormats

val person = Person("iteblog", 110)
val jvalue = Extraction.decompose(person)

println(pretty(render(jvalue)))
```

输出

```
{
  "name" : "iteblog",
  "age" : 110
}
```

同时我们也可以直接将一个 json 对象转换成类对象：

```
val person =  Extraction.extract[Person](jvalue)
println(person)
```

输出

```
Person(iteblog,110)
```

我们可以看到上面的例子使用起来都很简单的。但是如果碰到元素的类型在 json4s 中没有事先定义，结果会怎么样呢？比如在 json4s 中并没有定义对 java.sql.Date 类型的解析，那如果我们用到了这个类型，会出现什么问题呢？具体如下：

```
case class Person(name: String, age: Int, birthday: Date)
implicit val formats: Formats = DefaultFormats

val person = Person("iteblog", 110, Date.valueOf("2019-07-01"))
val jvalue = Extraction.decompose(person)

println(pretty(render(jvalue)))
```

输出

```
{
  "name" : "iteblog",
  "age" : 110,
  "birthday" : { }
}
```

可以从上面的结果看出，json4s 并没有正确的解析出 birthday 字段的值。 如果我们将 json 解析到类中，会出现什么问题呢？

```
val jvalue1 = parse("""{"name" : "iteblog", "age" : 110, "birthday" : "2019-07-01"}""")
val person = Extraction.extract[Person](jvalue1)
println(person)
```

结果

```
Exception in thread "main" org.json4s.package$MappingException: No usable value for birthd
ay
```

```
Parsed JSON values do not match with class constructor
args=
arg types=
executable=Executable(Constructor(public java.sql.Date(int,int,int)))
cause=wrong number of arguments
types comparison result=MISSING(int),MISSING(int),MISSING(int)
 at org.json4s.reflect.package$.fail(package.scala:95)
 at org.json4s.Extraction$ClassInstanceBuilder.org$json4s$Extraction$ClassInstanceBuilder$$
buildCtorArg(Extraction.scala:526)
 at org.json4s.Extraction$ClassInstanceBuilder$$anonfun$15.apply(Extraction.scala:546)
 at org.json4s.Extraction$ClassInstanceBuilder$$anonfun$15.apply(Extraction.scala:546)
 at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
 at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
 at scala.collection.mutable.ResizableArray$class.foreach(ResizableArray.scala:59)
 at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:48)
 at scala.collection.TraversableLike$class.map(TraversableLike.scala:234)
 at scala.collection.AbstractTraversable.map(Traversable.scala:104)
 at org.json4s.Extraction$ClassInstanceBuilder.instantiate(Extraction.scala:546)
 at org.json4s.Extraction$ClassInstanceBuilder.result(Extraction.scala:597)
 at org.json4s.Extraction$$anonfun$extract$6.apply(Extraction.scala:400)
 at org.json4s.Extraction$$anonfun$extract$6.apply(Extraction.scala:392)
 at org.json4s.Extraction$.customOrElse(Extraction.scala:606)
 at org.json4s.Extraction$.extract(Extraction.scala:392)
 at org.json4s.Extraction$.extract(Extraction.scala:39)
 at com.iteblog.Test$.main(Test.scala:32)
 at com.iteblog.Test.main(Test.scala)
Caused by: org.json4s.package$MappingException: Parsed JSON values do not match with clas
s constructor
args=
arg types=
executable=Executable(Constructor(public java.sql.Date(int,int,int)))
cause=wrong number of arguments
types comparison result=MISSING(int),MISSING(int),MISSING(int)
 at org.json4s.reflect.package$.fail(package.scala:95)
 at org.json4s.Extraction$ClassInstanceBuilder.instantiate(Extraction.scala:573)
 at org.json4s.Extraction$ClassInstanceBuilder.result(Extraction.scala:597)
 at org.json4s.Extraction$$anonfun$extract$6.apply(Extraction.scala:400)
 at org.json4s.Extraction$$anonfun$extract$6.apply(Extraction.scala:392)
 at org.json4s.Extraction$.customOrElse(Extraction.scala:606)
 at org.json4s.Extraction$.extract(Extraction.scala:392)
 at org.json4s.Extraction$ClassInstanceBuilder.org$json4s$Extraction$ClassInstanceBuilder$$
buildCtorArg(Extraction.scala:514)
 ... 17 more
```

可以看出，出现无法解析出 birthday 字段，因为 json4s 并没有提供将字符串解析到 java.sql.Date。那怎么办呢？json4s 为我们提供了自定义解析类型的方法，那就是 CustomSerializer，我们只需要继承这个类，并实现对自定义类型的序列化和反序列化的方法即可。那对我们的例子可以实现如下：

```
package com.iteblog

import java.sql.Date

import org.json4s.JsonAST.{JNull, JString}
import org.json4s.{CustomSerializer, DefaultFormats, Extraction, Formats}
import org.json4s.jackson.JsonMethods.render
import org.json4s.jackson.JsonMethods.pretty
import org.json4s.jackson.JsonMethods.parse

object Iteblog {

  case class Person(name: String, age: Int, birthday: Date)

  case object DateSerializer extends CustomSerializer[Date](_ => ( {
    case JString(s) => Date.valueOf(s)
    case JNull => null
  }, {
    case d: Date => JString(d.toString)
  }))

  def main(args: Array[String]): Unit = {

    implicit val formats: Formats = DefaultFormats + DateSerializer

    val person = Person("iteblog", 110, Date.valueOf("2019-07-01"))
    val jvalue = Extraction.decompose(person)

    println(pretty(render(jvalue)))

    val jvalue1 = parse("""{"name" : "iteblog", "age" : 110, "birthday" : "2019-07-01"}""")
    val r = Extraction.extract[Person](jvalue1)
    println(r)


  }
}
```

输出

```
{
  "name" : "iteblog",
  "age" : 110,
  "birthday" : "2019-07-01"
}
Person(iteblog,110,2019-07-01)
```

可见，通过自定义的 DateSerializer 我们可以解析 java.sql.Date 类型了。