JAVA对象转为JavaString的几种常用方法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/167/2021_2022_JAVA_E5_AF_B9_E8_B1_A1_c104_167005.htm 在java项目的实际开发和应用中，常常需要用到将对象转为string这一基本功能。本文将对常用的转换方法进行一个总结。常用的方法有object#tostring()，（string）要转换的对象，string.valueof(object)等。下面对这些方法一一进行分析。 方法1：采用 object#tostring()方法 请看下面的例子： object object = getobject(). system.out.println(object.tostring()). 在这种使用方法中，因为java.lang.object类里已有public方法.tostring()，所以对任何严格意义上的java对象都可以调用此方法。但在使用时要注意，必须保证object不是null值，否则将抛出nullpointerexception异常。采用这种方法时，通常派生类会覆盖object里的tostring（）方法。 方法2：采用类型转换（string）object方法 这是标准的类型转换，将object转成string类型的值。使用这种方法时，需要注意的是类型必须能转成string类型。因此最好用instanceof做个类型检查，以判断是否可以转换。否则容易抛出calsscastexception异常。此外，需特别小心的是因定义为object 类型的对象在转成string时语法检查并不会报错，这将可能导致潜在的错误存在。这时要格外小心。如： object obj = new integer(100). string strval = (string)obj. 在运行时将会出错，因为将integer类型强制转换为string类型，无法通过。但是， integer obj = new integer(100). string strval = (string)obj. 如是格式代码，将会报语法错误。 此外，因null值可以强制转换为任何java类类型，(string)null也是

合法的。 方法3：采用string.valueof(object) string.valueof(object)的基础是object#tostring()。但它与object#tostring()又有所不同。在前面方法1的分析中提到，使用后者时需保证不为null。但采用第三种方法时，将不用担心object是否为null值这一问题。为了便于说明问题，我们来分析一下相关的源代码。jdk里string# valueof(object)源码如下： /** * returns the string representation of the object argument. * * @param obj an object. * @return if the argument is null, then a string equal to * null. otherwise, the value of * obj.tostring() is returned. * @see java.lang.object#tostring() */ public static string valueof(object obj) { return (obj == null) ? null : obj.tostring(). } 从上面的源码可以很清晰的看出null值不用担心的理由。但是，这也恰恰给了我们隐患。我们应当注意到，当object为null时，string.valueof（object）的值是字符串 ” null ” ，而不是null ！ ！ ！在使用过程中切记要注意。试想一下，如果我们用 if(string.valueof（object）==null){system.out.println(“ 传入的值是null ！ ”).} 这样的语句将可能会发生什么问题。再想一下，向控制台输出时，在视觉上如下语句在执行的结果上有什么不同： system.out.println(string.valueof(null)). system.out.println(null).