**java基础之断言细谈**

断言是JDK1.4中引入的一个新的关键字，是一种错误处理机制，是在程序的开发和测试阶段使用的工具。在实现中，assertion就是在程序中的一条语句，它对一个boolean表达式进行检查，一个正确程序必须保证这个boolean表达式的值为true；（程序员认为这个状态是true。）如果该值为false，说明程序已经处于不正确的状态下，系统将给出警告并且退出。

assertion检查有助于增强代码的健壮性，比如如果在程序中出现了某种错误，可以更方便地调试程序。这样做要比程序在某处执行失败造成不良后果来发现错误要好得多。当知道程序失败是由于它违反了假设而引起的时候，跟踪失败的原因要简单得多。要记住断言是用来对那些不应该出现的情况进行实际的“健全性检查”，因此不应该使用它们来替代常规的错误检查。

一般来说，assertion用于保证程序最基本、关键的正确性。assertion检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion检查通常是关闭的。通常，在整个开发阶段都会启用断言。一旦完全测试了系统并将它移送到产品环境时，则希望禁用断言，因为这样做会略微改善性能。使用assert，只需一行代码，并且不必从发布的代码中删除assert语句。不管怎样，为了编制文档的目的，断言也应保留在代码中。这样，当以后更改代码时，会提醒程序员要保持所有假设都是有效的，并且这也是可测试的。

n 记住两点：

1. 断言失败是致命的、不可恢复的错误。

2. 断言检查仅仅用在程序开发和测试阶段。

n 因此，断言仅仅应该在测试阶段用来定位程序内部错误。

（一） 使用方式

断言可以有两种形式

1.assert Expression1

2.assert Expression1:Expression2

其中Expression1是一个布尔表达式，Expression2可以是任何对象或原始类型（包括null）。

在运行时，如果关闭了assertion功能，这些语句将不起任何作用。如果打开了assertion功能，那么expression1的值将被计算，如果它的值为false，该语句强抛出一个AssertionError对象。如果assertion语句包括expression2参数，程序将计算出expression2的结果，然后将这个结果作为AssertionError的构造函数的参数，来创建AssertionError对象，并抛出该对象，expression2可以在发生断言失败时获得更多的细节信息。如果expression1值为true，expression2将不被计算。AssertionError继承于Error对象，而Error继承于Throwable，Error是和Exception并列的一个错误对象，通常用于表达系统级运行错误，也就是说是一个不可控制异常（unchecked Exception)。AssertionError由于是错误，所以可以不捕获，但不推荐这样做，因为那样会使你的系统进入不稳定状态。

注意：（1）一种特殊情况是，如果在计算表达式时，表达式本身抛出Exception，那么assert将停止运行，而抛出这个Exception。

（2）AssertionError对象并不存储表达式Expression2的值，因此你不可能在以后获取它。

例如，如果要进行如下的计算时：

? double y=Math.sqrt(x);

sqrt(x)是一个开平方运算，x必须为正才不会出错。为了检查传入的参数是否为正，可以使用如下的断言语句：

? assert x>=0;

? double y=Math.sqrt(x);

或者

? assert x>=0：“ x>=0”;//将“ x>=0”传给AssertionError对象，从而可在出错时显示出来

? double y=Math.sqrt(x);

当x为负值时， assert语句将抛出AssertionError异常，你就可以根据异常信息对程序的其它部分进行检查。

（二）编译运行

（1）在编译时必须使用-source 1.4选项，-source 1.4表示使用JDK 1.4版本的方式来编译源代码，否则编译就不能通过，因为缺省的Javac编译器使用JDK1.3的语法规则。

例如：

javac -source 1.4 Myclass.class

（2）在使用断言以前，你需要先开启断言功能，因为Java Runtime Environment(JRE)默认关闭断言功能。在运行时启用用户类的断言需要使用 -ea参数 。要在系统类（不通过类装载器而由JVM直接装载的类）中启用和禁用断言可以使用 -esa 和 -dsa参数。打开或者关闭断言是类装载器的功能。当断言功能被关闭时，类装载器会跳过那些和断言相关的代码，因此不会降低程序运行速度，即它们没有任何副作用。

例如：java –ea

oo.bar... -da

oo.bar.old Myclass

意思是开启对于包foo.bar和其子包的断言，包foo.bar.old除外。

1. 参数 -esa 和 -dsa：

它们含义为开启(关闭)系统类的assertion功能。由于新版本的Java的系统类中，也使了assertion语句，因此如果用户需要观察它们的运行情况，就需要打开系统类的assertion功能，我们可使用-esa参数打开，使用 -dsa参数关闭。

-esa和-dsa的全名为-enablesystemassertions和-disenablesystemassertions，全名和缩写名有同样的功能。

2. 参数 -ea和-da：

它们含义为开启(关闭)用户类的assertion功能：通过这个参数，用户可以打开某些类或包的assertion功能，同样用户也可以关闭某些类和包的assertion功能。打开assertion功能参数为-ea；如果不带任何参数，表示打开所有用户类；如果带有包名称或者类名称，表示打开这些类或包；如果包名称后面跟有三个点，代表这个包及其子包；如果只有三个点，代表无名包。关闭assertion功能参数为-da，使用方法与-ea类似。-ea和-da的全名为-enableassertions和-disenableassertions，全名和缩写名有同样的功能。

（三）如何检测断言是否被开启的程序。Sun提供了这个小技巧：

static {

boolean assertsEnabled = false;

// here's the trick

assert assertsEnabled = true;

if( !assertsEnabled )

throw new RuntimeException(

"Asserts must be enabled!");

}

你可以在任何使用断言的代码中加入这个代码块。如果断言功能被关闭，那assert不会被执行，异常会被抛出。用次代码块保证断言开启

（四）警告：

Assert最好不要滥用，原因是assert并不一定都是enable的，断言语句不是永远会执行，可以屏蔽也可以启用

（1）不要让断言语句更改代码中的状态/值。否则当最终关闭断言时，代码的行为方式将不同于启用断言时代码的行为。也就是说，不要使用断言语句去修改变量和改变方法的返回值

例如，不要创建如下的断言：

assert (++i > 10); // BAD: i changes only with assertions enabled!

（2）不要再public的方法里面检查参数是不是为null之类的操作,也就是说不要使用断言作为公共方法的参数检查，因为公共方法的参数永远都要执行，这样就不能将assert设置为enable了

例如public int get(String s)

{

assert s != null;

}

假如需要检查也最好通过if s = null 抛出NullPointerException来检查

（3）一般来说，动作(actions)(方法、操作等)不应该被作为assert的第一个参数所使用，除非有特别的需要，因为它们可能不会被执行。

可以简言之：不要用assert来检查方法操作的返回值来判定方法操作的结果