[**Java命令行运行参数说明大全(偷来的)**](http://xinklabi.iteye.com/blog/837435)

**博客分类：**

* [Java](http://xinklabi.iteye.com/category/104300)

[Java](http://www.iteye.com/blogs/tag/Java)[虚拟机](http://www.iteye.com/blogs/tag/%E8%99%9A%E6%8B%9F%E6%9C%BA)[JNI](http://www.iteye.com/blogs/tag/JNI)[JVM](http://www.iteye.com/blogs/tag/JVM)[应用服务器](http://www.iteye.com/blogs/tag/%E5%BA%94%E7%94%A8%E6%9C%8D%E5%8A%A1%E5%99%A8)

 Java在运行已编译完成的类时，是通过java虚拟机来装载和执行的，java虚拟机通过操作系统命令JAVA\_HOME"bin"java –option 来启动，-option为虚拟机参数，JAVA\_HOME为JDK安装路径，通过这些参数可对虚拟机的运行状态进行调整，掌握参数的含义可对虚拟机的运行模式有更深入理解。

一、         查看参数列表:  
虚拟机参数分为基本和扩展两类，在命令行中输入JAVA\_HOME"bin"java 就可得到基本参数列表，在命令行输入JAVA\_HOME"bin"java –X 就可得到扩展参数列表。  
二、         基本参数说明：  
1.     -client，-server  
这两个参数用于设置虚拟机使用何种运行模式，client模式启动比较快，但运行时性能和内存管理效率不如server模式，通常用于客户端应用程序。相反，server模式启动比client慢，但可获得更高的运行性能。  
在 windows上，缺省的虚拟机类型为client模式，如果要使用 server模式，就需要在启动虚拟机时加-server参数，以获得更高性能，对服务器端应用，推荐采用server模式，尤其是多个CPU的系统。在 Linux，Solaris上缺省采用server模式。   
2.     -hotspot  
含义与client相同，jdk1.4以前使用的参数，jdk1.4开始不再使用，代之以client。  
3.     -classpath,-cp  
虚拟机在运行一个类时，需要将其装入内存，虚拟机搜索类的方式和顺序如下：  
Bootstrap classes，Extension classes，User classes。  
Bootstrap 中的路径是虚拟机自带的jar或zip文件，虚拟机首先搜索这些包文件，用System.getProperty("sun.boot.class.path")可得到虚拟机搜索的包名。  
Extension是位于jre"lib"ext目录下的jar文件，虚拟机在搜索完Bootstrap后就搜索该目录下的jar文件。用System. getProperty("java.ext.dirs”)可得到虚拟机使用Extension搜索路径。  
User classes搜索顺序为当前目录、环境变量 CLASSPATH、-classpath。  
4.     -classpath  
告知虚拟机搜索目录名、jar文档名、zip文档名，之间用分号;分隔。  
例如当你自己开发了公共类并包装成一个common.jar包，在使用 common.jar中的类时，就需要用-classpath common.jar 告诉虚拟机从common.jar中查找该类，否则虚拟机就会抛出java.lang.NoClassDefFoundError异常，表明未找到类定义。   
在运行时可用System.getProperty(“java.class.path”)得到虚拟机查找类的路径。

使用-classpath后虚拟机将不再使用CLASSPATH中的类搜索路径，如果-classpath和CLASSPATH都没有设置，则虚拟机使用当前路径(.)作为类搜索路径。  
推荐使用-classpath来定义虚拟机要搜索的类路径，而不要使用环境变量 CLASSPATH的搜索路径，以减少多个项目同时使用CLASSPATH时存在的潜在冲突。例如应用1要使用a1.0.jar中的类G，应用2要使用 a2.0.jar中的类G,a2.0.jar是a1.0.jar的升级包，当a1.0.jar，a2.0.jar都在CLASSPATH中，虚拟机搜索到第一个包中的类G时就停止搜索，如果应用1应用2的虚拟机都从CLASSPATH中搜索，就会有一个应用得不到正确版本的类G。  
5.     -D<propertyName>=value  
在虚拟机的系统属性中设置属性名/值对，运行在此虚拟机之上的应用程序可用System.getProperty(“propertyName”)得到value的值。  
如果value中有空格，则需要用双引号将该值括起来，如-Dname=”space string”。  
该参数通常用于设置系统级全局变量值，如配置文件路径，应为该属性在程序中任何地方都可访问。

6.    -verbose[:class|gc|jni]  
在输出设备上显示虚拟机运行信息。  
verbose和verbose:class含义相同，输出虚拟机装入的类的信息，显示的信息格式如下：  
[Loaded java.io.FilePermission$1 from shared objects file]  
当虚拟机报告类找不到或类冲突时可用此参数来诊断来查看虚拟机从装入类的情况。  
7.     -verbose:gc  
在虚拟机发生内存回收时在输出设备显示信息，格式如下：  
[Full GC 268K->168K(1984K), 0.0187390 secs]  
该参数用来监视虚拟机内存回收的情况。  
8.     -verbose:jni  
在虚拟机调用native方法时输出设备显示信息，格式如下：  
[Dynamic-linking native method HelloNative.sum ... JNI]  
该参数用来监视虚拟机调用本地方法的情况，在发生jni错误时可为诊断提供便利。  
9.     -version  
显示可运行的虚拟机版本信息然后退出。一台机器上装有不同版本的JDK时  
10.-showversion  
显示版本信息以及帮助信息。 内容来自dedecms   
11.-ea[:<packagename>...|:<classname>]  
12.-enableassertions[:<packagename>...|:<classname>]  
从JDK1.4开始，java可支持断言机制，用于诊断运行时问题。通常在测试阶段使断言有效，在正式运行时不需要运行断言。断言后的表达式的值是一个逻辑值，为true时断言不运行，为false时断言运行，抛出java.lang.AssertionError错误。  
上述参数就用来设置虚拟机是否启动断言机制，缺省时虚拟机关闭断言机制，用-ea 可打开断言机制，不加<packagename>和classname时运行所有包和类中的断言，如果希望只运行某些包或类中的断言，可将包名或类名加到-ea之后。例如要启动包com.foo.util中的断言，可用命令 –ea:com.foo.util 。  
13.-da[:<packagename>...|:<classname>]  
14.-disableassertions[:<packagename>...|:<classname>]  
用来设置虚拟机关闭断言处理，packagename和classname的使用方法和-ea相同。  
15.-esa | -enablesystemassertions  
设置虚拟机显示系统类的断言。  
16.-dsa | -disablesystemassertions  
设置虚拟机关闭系统类的断言。 dedecms.com   
17.-agentlib:<libname>[=<options>]  
该参数是JDK5新引入的，用于虚拟机装载本地代理库。  
Libname 为本地代理库文件名，虚拟机的搜索路径为环境变量PATH中的路径，options为传给本地库启动时的参数，多个参数之间用逗号分隔。在Windows 平台上虚拟机搜索本地库名为libname.dll的文件，在Unix上虚拟机搜索本地库名为libname.so的文件，搜索路径环境变量在不同系统上有所不同，Linux、SunOS、IRIX上为LD\_LIBRARY\_PATH，AIX上为LIBPATH，HP-UX上为SHLIB\_PATH。  
例如可使用-agentlib:hprof来获取虚拟机的运行情况，包括CPU、内存、线程等的运行数据，并可输出到指定文件中，可用-agentlib:hprof=help来得到使用帮助列表。在jre"bin目录下可发现hprof.dll文件。  
18. -agentpath:<pathname>[=<options>]  
设置虚拟机按全路径装载本地库，不再搜索PATH中的路径。其他功能和agentlib相同。  
19.-javaagent:<jarpath>[=<options>]  
虚拟机启动时装入java语言设备代理。Jarpath文件中的mainfest 文件必须有Agent-Class属性。代理类要实现public static void premain(String agentArgs, Instrumentation inst)方法。当虚拟机初始化时，将按代理类的说明顺序调用premain方法。 copyright dedecms   
参见：java.lang.instrument  
三、         扩展参数说明  
1.     -Xmixed  
设置-client模式虚拟机对使用频率高的方式进行Just-In-Time编译和执行，对其他方法使用解释方式执行。该方式是虚拟机缺省模式。  
2.     -Xint  
设置-client模式下运行的虚拟机以解释方式执行类的字节码，不将字节码编译为本机码。  
3.     -Xbootclasspath:path  
4.     -Xbootclasspath/a:path  
5.     -Xbootclasspath/p:path  
改变虚拟机装载缺省系统运行包rt.jar而从-Xbootclasspath中设定的搜索路径中装载系统运行类。除非你自己能写一个运行时，否则不会用到该参数。  
/a:将在缺省搜索路径后加上path 中的搜索路径。  
/p:在缺省搜索路径前先搜索path中的搜索路径。  
6.     -Xnoclassgc  
关闭虚拟机对class的垃圾回收功能。  
7.     -Xincgc  
启动增量垃圾收集器，缺省是关闭的。增量垃圾收集器能减少偶然发生的长时间的垃圾回收造成的暂停时间。但增量垃圾收集器和应用程序并发执行，因此会占用部分CPU在应用程序上的功能。   
8.     -Xloggc:<file>  
将虚拟机每次垃圾回收的信息写到日志文件中，文件名由file指定，文件格式是平文件，内容和-verbose:gc输出内容相同。  
9.     -Xbatch  
虚拟机的缺省运行方式是在后台编译类代码，然后在前台执行代码，使用-Xbatch参数将关闭虚拟机后台编译，在前台编译完成后再执行。  
10.-Xms<size>  
设置虚拟机可用内存堆的初始大小，缺省单位为字节，该大小为1024的整数倍并且要大于1MB，可用k(K)或m(M)为单位来设置较大的内存数。初始堆大小为2MB。  
例如：-Xms6400K，-Xms256M  
11.-Xmx<size>  
设置虚拟机内存堆的最大可用大小，缺省单位为字节。该值必须为1024整数倍，并且要大于2MB。可用k(K)或m(M)为单位来设置较大的内存数。缺省堆最大值为64MB。  
例如：-Xmx81920K，-Xmx80M  
当应用程序申请了大内存运行时虚拟机抛出java.lang.OutOfMemoryError: Java heap space错误，就需要使用-Xmx设媒洗蟮目捎媚诖娑选?BR>  
12.-Xss<size>  
设置线程栈的大小，缺省单位为字节。与-Xmx类似，也可用K或M来设置较大的值。通常操作系统分配给线程栈的缺省大小为1MB。   
另外也可在java中创建线程对象时设置栈的大小，构造函数原型为Thread(ThreadGroup group, Runnable target, String name, long stackSize)。  
13.-Xprof  
输出CPU运行时的诊断信息。  
14.-Xfuture  
对类文件进行严格格式检查，以保证类代码符合类代码规范。为保持向后兼容，虚拟机缺省不进行严格的格式检查。  
15.-Xrs  
减少虚拟机中操作系统的信号（singals）的使用。该参数通常用在虚拟机以后台服务方式运行时使用（如Servlet）。  
16.-Xcheck:jni

**一、运行class文件**   
执行带main方法的class文件，命令行为：  
java <CLASS文件名>  
注意：CLASS文件名不要带文件后缀.class  
例如：  
java Test  
如果执行的class文件是带包的，即在类文件中使用了：  
package <包名>  
那应该在包的基路径下执行，命令行为：  
java <包名>.CLASS文件名  
例如：  
PackageTest.java中，其包名为：com.ee2ee.test，对应的语句为：  
package com.ee2ee.test;  
PackageTest.java及编译后的class文件PackageTest.class的存放目录如下：  
classes  
  |\_\_com  
      |\_\_ee2ee  
           |\_\_test  
                |\_\_PackageTest.java  
                |\_\_PackageTest.class  
  
要运行PackageTest.class，应在classes目录下执行：  
java com.ee2ee.test.PackageTest   
**二、运行jar文件中的class**   
原理和运行class文件一样，只需加上参数-cp <jar文件名>即可。  
例如：执行test.jar中的类com.ee2ee.test.PackageTest，命令行如下：  
java -cp test.jar com.ee2ee.test.PackageTest   
**三、显示jdk版本信息**   
当一台机器上有多个jdk版本时，需要知道当前使用的是那个版本的jdk，使用参数-version即可知道其版本，命令行为：  
java -version   
**四、增加虚拟机可以使用的最大内存**   
java虚拟机可使用的最大内存是有限制的，缺省值通常为64MB或128MB。  
如果一个应用程序为了提高性能而把数据加载内存中而占用较大的内存，比如超过了默认的最大值128MB，需要加大java虚拟机可使用的最大内存，否则会出现Out of Memory（系统内存不足）的异常。启动java时，需要使用如下两个参数：  
-Xms java虚拟机初始化时使用的内存大小  
-Xmx java虚拟机可以使用的最大内存  
以上两个参数中设置的size，可以带单位，例如：256m表示256MB  
举例说明：  
java -Xms128m -Xmx256m ...  
表示java虚拟机初始化时使用的内存为128MB，可使用的最大内存为256MB。  
对于tomcat，可以修改其脚本catalina.sh（unix平台）或catalina.bat（windows平台），设置变量JAVA\_OPTS即可，例如：  
JAVA\_OPTS='-Xms128m -Xmx256m'

在控制台输出信息中，有个-X（注意是大写）的命令，这个正是查看JVM配置参数的命令。

**其次，用java -X 命令查看JVM的配置说明：**

运行后如下结果，这些就是配置JVM参数的秘密武器，这些信息都是英文的，为了方便阅读，我根据自己的理解翻译成中文了（不准确的地方还请各位博友斧正）

-----------------------------------------------------------------------

D:\j2sdk15\bin>java -X

    -Xmixed           mixed mode execution (default)

    -Xint             interpreted mode execution only

    -Xbootclasspath:<directories and zip/jar files separated by ;>

                      set search path for bootstrap classes and resources

    -Xbootclasspath/a:<directories and zip/jar files separated by ;>

                      append to end of bootstrap class path

    -Xbootclasspath/p:<directories and zip/jar files separated by ;>

                      prepend in front of bootstrap class path

    -Xnoclassgc       disable class garbage collection

    -Xincgc           enable incremental garbage collection

    -Xloggc:<file>    log GC status to a file with time stamps

    -Xbatch           disable background compilation

    -Xms<size>        set initial Java heap size

    -Xmx<size>        set maximum Java heap size

    -Xss<size>        set java thread stack size

    -Xprof            output cpu profiling data

    -Xfuture          enable strictest checks, anticipating future default

    -Xrs              reduce use of OS signals by Java/VM (see documentation)

    -Xcheck:jni       perform. additional checks for JNI functions

    -Xshare:off       do not attempt to use shared class data

    -Xshare:auto      use shared class data if possible (default)

    -Xshare:on        require using shared class data, otherwise fail.

The -X options are non-standard and subject to change without notice.

-----------------------------------------------------------------------

JVM配置参数中文说明：

-----------------------------------------------------------------------

1、-Xmixed           mixed mode execution (default)

 混合模式执行

2、-Xint             interpreted mode execution only

 解释模式执行

3、-Xbootclasspath:<directories and zip/jar files separated by ;>

      set search path for bootstrap classes and resources

 设置zip/jar资源或者类（.class文件）存放目录路径

3、-Xbootclasspath/a:<directories and zip/jar files separated by ;>

      append to end of bootstrap class path

 追加zip/jar资源或者类（.class文件）存放目录路径

4、-Xbootclasspath/p:<directories and zip/jar files separated by ;>

      prepend in front of bootstrap class path

 预先加载zip/jar资源或者类（.class文件）存放目录路径

5、-Xnoclassgc       disable class garbage collection

 关闭类垃圾回收功能

6、-Xincgc           enable incremental garbage collection

 开启类的垃圾回收功能

7、-Xloggc:<file>    log GC status to a file with time stamps

 记录垃圾回日志到一个文件。

8、-Xbatch           disable background compilation

 关闭后台编译

9、-Xms<size>        set initial Java heap size

 设置JVM初始化堆内存大小

10、-Xmx<size>        set maximum Java heap size

 设置JVM最大的堆内存大小

11、-Xss<size>        set java thread stack size

 设置JVM栈内存大小

12、-Xprof            output cpu profiling data

 输入CPU概要表数据

13、-Xfuture          enable strictest checks, anticipating future default

 执行严格的代码检查，预测可能出现的情况

14、-Xrs              reduce use of OS signals by Java/VM (see documentation)

 通过JVM还原操作系统信号

15、-Xcheck:jni       perform. additional checks for JNI functions

 对JNI函数执行检查

16、-Xshare:off       do not attempt to use shared class data

 尽可能不去使用共享类的数据

17、-Xshare:auto      use shared class data if possible (default)

 尽可能的使用共享类的数据

18、-Xshare:on       require using shared class data, otherwise fail.

 尽可能的使用共享类的数据，否则运行失败

The -X options are non-standard and subject to change without notice.

-----------------------------------------------------------------------

怎么用这这些参数呢？其实所有的命令行都是这么一用，下面我就给出一个最简单的HelloWorl的例子来演示这个参数的用法，非常的简单。

HelloWorld.java

-----------------------------------------------

public class  HelloWorld

{

 public static void main(String[] args)

 {

  System.out.println("Hello World!");

 }

}

编译并运行：

D:\j2sdk15\bin>javac HelloWorld.java

D:\j2sdk15\bin>java -Xms256M -Xmx512M HelloWorld

Hello World!

呵呵，这下满足了吧！

**实践：在大型系统或者应用中配置JVM参数**

比如你配置IDE工具的参数，常见的有IDEA、Eclipse，这个是在一个配置文件中指定即可。

如果你要在J2EE环境中配置这些参数，那么你需要在J2EE应用服务器或者Servlet容器相关启动参数设置处指定，其启动文件中来配置，Tomcat是在catalina.bat中配置，weblogic和websphere是在其他地方，具体我就说了，相信玩过的这些大型服务器的人都知道，没玩过的看看这篇文章，玩玩就知道了，呵呵。

另外常常有人问到jdk的一些相关命令用法，其实，当你看到这里的时候，你应该知道如何获取这些命令的用法了。如果你还不会，那么，建议你去学学DOS，我是没辙了。如果你会这些，还是没有看明白，那么你赶紧学学英语吧，这样你就能看懂了。