Eclipse崩溃，错误提示：  
MyEclipse has detected that less than 5% of the 64MB of Perm   
Gen (Non-heap memory) space remains. It is strongly recommended  
that you exit and restart MyEclipse with new virtual machine memory  
paramters to increase this memory.   Failure to do so can result in  
data loss. The recommended Eclipse memory parameters are:   
eclipse.exe -vmargs -Xms128M -Xmx512M -XX:PermSize=64M -XX:MaxPermSize=128M  
   
1.参数的含义  
**-vmargs -Xms128M -Xmx512M -XX:PermSize=64M -XX:MaxPermSize=128M  
-vmargs 说明后面是VM的参数，所以后面的其实都是JVM的参数了  
-Xms128m JVM初始分配的堆内存  
-Xmx512m JVM最大允许分配的堆内存，按需分配  
-XX:PermSize=64M JVM初始分配的非堆内存  
-XX:MaxPermSize=128M JVM最大允许分配的非堆内存，按需分配**

我们首先了解一下JVM内存管理的机制，然后再解释每个参数代表的含义。

**1)堆(Heap)和非堆(Non-heap)内存**

 按照官方的说法：“Java 虚拟机具有一个堆，堆是运行时数据区域，所有类实例和数组的内存均从此处分配。堆是在 Java 虚拟机启动时创建的。”“在JVM中堆之外的内存称为非堆内存(Non-heap memory)”。  
 可以看出JVM主要管理两种类型的内存：堆和非堆。简单来说堆就是Java代码可及的内存，是留给开发人员使用的；非堆就是JVM留给自己用的，  
 所以方法区、JVM内部处理或优化所需的内存(如JIT编译后的代码缓存)、每个类结构(如运行时常数池、字段和方法数据)以及方法和构造方法的代码都在非堆内存中。

**堆内存分配**

 JVM初始分配的堆内存由-Xms指定，默认是物理内存的1/64；JVM最大分配的堆内存由-Xmx指定，默认是物理内存的1/4。默认空余堆内存小于40%时，JVM就会增大堆直到-Xmx的最大限制；  
 空余堆内存大于70%时，JVM会减少堆直到-Xms的最小限制。因此服务器一般设置-Xms、-Xmx 相等以避免在每次GC 后调整堆的大小。  
 说明：如果-Xmx 不指定或者指定偏小，应用可能会导致java.lang.OutOfMemory错误，此错误来自JVM，不是Throwable的，无法用try...catch捕捉。

**非堆内存分配**

 JVM使用-XX:PermSize设置非堆内存初始值，默认是物理内存的1/64；由XX:MaxPermSize设置最大非堆内存的大小，默认是物理内存的1/4。（还有一说：MaxPermSize缺省值和-server -client选项相关，  
 -server选项下默认MaxPermSize为64m，-client选项下默认MaxPermSize为32m。这个我没有实验。）  
 上面错误信息中的PermGen space的全称是Permanent Generation space，是指内存的永久保存区域。还没有弄明白PermGen space是属于非堆内存，还是就是非堆内存，但至少是属于了。  
XX:MaxPermSize设置过小会导致java.lang.OutOfMemoryError: PermGen space 就是内存益出。   
说说为什么会内存益出：   
（1）这一部分内存用于存放Class和Meta的信息，Class在被 Load的时候被放入PermGen space区域，它和存放Instance的Heap区域不同。   
（2）GC(Garbage Collection)不会在主程序运行期对PermGen space进行清理，所以如果你的APP会LOAD很多CLASS 的话,就很可能出现PermGen space错误。  
  这种错误常见在web服务器对JSP进行pre compile的时候。

**2)JVM内存限制(最大值)**

 首先JVM内存限制于实际的最大物理内存，假设物理内存无限大的话，JVM内存的最大值跟操作系统有很大的关系。简单的说就32位处理器虽然可控内存空间有4GB,但是具体的操作系统会给一个限制，  
 这个限制一般是2GB-3GB（一般来说Windows系统下为1.5G-2G，Linux系统下为2G-3G），而64bit以上的处理器就不会有限制了。

2. 为什么有的机器我将-Xmx和-XX:MaxPermSize都设置为512M之后Eclipse可以启动，而有些机器无法启动？  
 通过上面对JVM内存管理的介绍我们已经了解到JVM内存包含两种：堆内存和非堆内存，另外JVM最大内存首先取决于实际的物理内存和操作系统。所以说设置VM参数导致程序无法启动主要有以下几种原因：  
1) 参数中-Xms的值大于-Xmx，或者-XX:PermSize的值大于-XX:MaxPermSize；  
2) -Xmx的值和-XX:MaxPermSize的总和超过了JVM内存的最大限制，比如当前操作系统最大内存限制，或者实际的物理内存等等。说到实际物理内存这里需要说明一点的是，  
 如果你的内存是1024MB，但实际系统中用到的并不可能是1024MB，因为有一部分被硬件占用了。

3. 为何将上面的参数写入到eclipse.ini文件Eclipse没有执行对应的设置？  
 那为什么同样的参数在快捷方式或者命令行中有效而在eclipse.ini文件中是无效的呢？这是因为我们没有遵守eclipse.ini文件的设置规则：  
参数形如“项 值”这种形式，中间有空格的需要换行书写，如果值中有空格的需要用双引号包括起来。比如我们使用-vm C:/Java/jre1.6.0/bin/javaw.exe参数设置虚拟机，  
在eclipse.ini文件中要写成这样：  
-vm   
C:/Java/jre1.6.0/bin/javaw.exe   
-vmargs   
-Xms128M   
-Xmx512M   
-XX:PermSize=64M   
-XX:MaxPermSize=128M   
实际运行的结果可以通过Eclipse中“Help”-“About Eclipse SDK”窗口里面的“Configuration Details”按钮进行查看。  
另外需要说明的是，Eclipse压缩包中自带的eclipse.ini文件内容是这样的：  
-showsplash   
org.eclipse.platform   
--launcher.XXMaxPermSize   
256m   
-vmargs   
-Xms40m   
-Xmx256m   
其中–launcher.XXMaxPermSize（注意最前面是两个连接线）跟-XX:MaxPermSize参数的含义基本是一样的，我觉得唯一的区别就是前者是eclipse.exe启动的时候设置的参数，  
而后者是eclipse所使用的JVM中的参数。其实二者设置一个就可以了，所以这里可以把–launcher.XXMaxPermSize和下一行使用#注释掉。

4. 其他的启动参数。 如果你有一个双核的CPU，也许可以尝试这个参数:  
-XX:+UseParallelGC  
让GC可以更快的执行。（只是JDK 5里对GC新增加的参数）

补充：  
　　如果你的WEB APP下都用了大量的第三方jar，其大小超过了服务器jvm默认的大小，那么就会产生内存益出问题了。  
解决方法： 设置MaxPermSize大小   
可以在myelipse里选中相应的服务器比如tomcat5，展开里面的JDK子项页面，来增加服务器启动的JVM参数设置：  
-Xms128m   
-Xmx256m   
-XX:PermSize=128M   
-XX:MaxNewSize=256m   
-XX:MaxPermSize=256m  
或者手动设置MaxPermSize大小,比如tomcat，  
修改TOMCAT\_HOME/bin/catalina.bat，在echo "Using CATALINA\_BASE: $CATALINA\_BASE"上面加入以下行：   
JAVA\_OPTS="-server -XX:PermSize=64M -XX:MaxPermSize=128m

建议：将相同的第三方jar文件移置到tomcat/shared/lib目录下，这样可以减少jar 文档重复占用内存

引自：<http://www.cnblogs.com/mingforyou/archive/2012/03/03/2378143.html>