深入学习jvm

重要的点：运行时数据区、类加载器

问题：

1》将.java文件编译成.class文件，是谁执行的 jvm还是jdk执行的？

答：jdk执行的，javac

正文：

1.常有加载器：Bootstrap加载器、AppClassLoader加载器

2.运行时数据区组成部分：堆、栈、方法区、程序计数器、本地方法栈。

3.最关键的问题就是优化：堆内存空间。

4.Jvm调优至关重要的参数设置：

1.-Xms:设置初始化的内存分配大小。

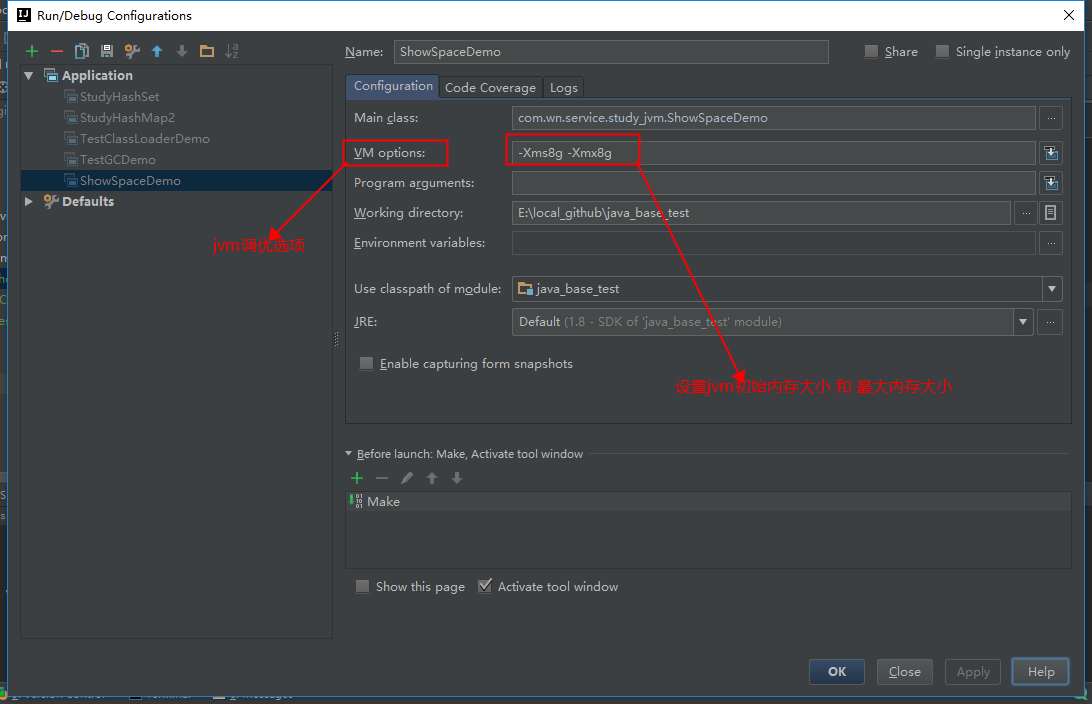
2.-Xmx:设置最大的可用内存空间。

3.在idea里面如何配置对内存：

A.以前是用：-Xms8g -Xmx8g -XX:+PrintGCDetails

B.现在是用(GC算法用G1的版本，jdk1.9的版本及之后)： -Xms8g -Xmx8g -Xlog:gc\*

参数配置的合适，可以减少对内存的收缩处理操作。



1. 年轻代和老年代：

年轻代：

所以在这个环节里面就需要考虑两个技术名词：BTP、TLAB

· BTP：在伊甸园区采用栈的形式将最晚创建的对象保存在栈顶。

· TLAB：分块保存，适合于多线程的处理操作上。

-Xmn：设置年轻代的空间大小，默认采用的时物理内存的“1 / 64”.

-Xss：设置每一个线程所占用的栈的线程

-X:SurviorRatio：设置伊甸园区与两个存活区之间的内存分配比，默认“8 : 1 : 1”。

老年代：

与年轻代比率：-XX:NewRatio

当对象很大的时候往往不在年轻代进行保存，而是直接晋级到老年代，利用“-XX:PretenureSizeThreshold”。

【分水岭】JDK1.8 之后取消了所谓的永久代，而变为了元空间（不在堆内存里面保存，而是直接利用物理内存保存。）

6.虚拟机的监控工具(在cmd可调用)：

Jconsole

jmap可以查看虚拟机相关信息。

二、GC 算法（主流：G1、未来：ZGC）

GC 算法的选择直接决定了你最终程序的执行性能。

传统意义上进行的回收处理操作，只是认为简单的有垃圾产生了，而后自动进行 GC 操作（MinorGC、MijorGC）或者手工利用“System.gc()”操作（MijorGC、FullGC）。