**[jvm系列 (一) ---jvm内存区域与溢出](http://www.cnblogs.com/-new/p/7244460.html)**

<http://www.cnblogs.com/-new/p/7244460.html>

# jvm内存区域与溢出

## 目录

* [jvm系列（一）：jvm内存区域与溢出](http://www.cnblogs.com/-new/p/7244460.html)
* [jvm系列（二）：垃圾收集器与内存分配策略](http://www.cnblogs.com/-new/p/7284368.html)

## 为什么学习jvm

* 木板原理，最短的一块板决定一个水的深度，当一个系统垃圾收集成为瓶颈的时候，那么就需要你对jvm的了解掌握。
* 当一个系统出现内存溢出，内存泄露的时候，因为你懂jvm知识，可以更加快速定位错误，可以通过参数去合理设置各内存区域的内存容量。
* 因为你对jvm的认识，写代码的时候会潜意识地让你注意代码质量，可能你会说是那是小小的性能提升，但是量变会导致质变的。

# jvm内存区域

## jvm内存划分

* 方法区
* 虚拟机栈
* 本地方法栈
* 堆
* 程序计数器

## 程序计数器

* 当前线程所执行的字节码的行号指示器。
* 因为cpu的每个核心只能同时运行一个线程，所以当一个线程执行完时间片后切换到另一个线程，切换时为了能恢复到正确的执行位置，所以需要程序计数器(学过计算机组成原理的应该比较熟悉)。
* 如果是线程执行的是一个java方法那么此时计数器记录的是正在执行的虚拟机字节码指令的地址。如果是nativ方法（非java代码），计数器值为空。

## 虚拟机栈

* java执行的内存模型，栈由栈帧组成，线程调用一个java方法时，创建一个栈帧，方法返回时，栈帧弹出。栈帧入栈出栈的过程就是方法开始结束的过程
* 栈帧存储局部变量区（存放基本数据类型数据，对象指针）,操作数栈（存放操作数，比如加法运算的时候操作数栈取值，计算后再压入栈），动态链接（将符号引用（常量池指向方法的符号引用）转换为调用方法的直接引用），方法出口。

## 本地方法栈

* 和虚拟机栈作用类似，只不过服务对象不同，本地方法服务对象是非java方法，虚拟机栈服务对象是java方法

## 方法区

* 存放类的信息、常量、静态变量等
* 运行时常量池是方法区一部分，主要存放字面量（如final修饰的变量）和符号引用量（编译原理方面）

## 堆

* 存放对象，对象要在堆上分配内存
* 堆分为年轻代和年老代
* 年轻代分为伊甸区（Eden space）和幸存者区（Survivor space）
* 幸存区分为from和to空间

## 总结

* jvm的内存区域划分为程序计数器，虚拟机栈，本地方法栈，方法区，堆。
* 程序计数器，虚拟机栈和本地方法栈都是线程独享的，而方法区和堆是线程共享的

# 溢出

## 理清概念

* 操作系统分配给进程的内存是有限的，而jvm的内存区域我们已经知道，当我们设置好堆和方法区的最大容量后，那么剩下的内存将分配给虚拟机栈和本地方法区和程序计数器（占用内存少）。
* 两种常见的溢出异常，一种OutOfMemoryError（OOM）,一种StackOverflowError。

## 各区溢出情况

* 堆溢出，当我们创建的对象占用的内存超过最大堆容量时候，会抛出OOM
* 本地方法栈和虚拟机栈溢出：当请求的栈深度超过虚拟机所允许的深度的时候会抛出栈溢出，这种情况我们在使用递归出错的时候经常遇到；还有就是出现OOM的情况，我们知道分配给这两个栈的内存是有限，和线程数和线程的栈内存有关系，那么当我们其中任意一个过大的话，都有可能造成OOM
* 方法区和运行时常量池溢出：方法区存放类的信息，有时我们用的框架在动态代理的时候会动态生成CLASS，有时候有很多jsp的时候，这个时候有可能会出现方法区的OOM；而常量池溢出我们可以用String的intern方法（如果常量池没有与字符串相等的字符串，就将这个字符串存入方法区，注意在不同的版本是不同的，1.6之前会将字符串实例复制到永久代，而1.7的没有复制实例，只是在常量池记录首次出现的实例引用）进行模拟。

## 设置各区的jvm参数

* 设置堆的最小值-Xms，堆的最大值-Xmx
* 设置永久代（jdk8之前用永久代来实现方法区）的最小值-XX：PermSize ，最大值-XX：MaxPermSize
* 设置栈容量-Xss

# 延伸

## 关于创建字符串

String s2=new String("jiajun");

String s6=new String("jiajun");

System.out.println(s2==s6);

* 结果为false，都存放在堆内存，但是两个地方。

String s6=new String("jiajun");

String s1="jiajun";

System.out.println(s1==s6);

* 结果为false，s6存放在堆当中，而s1存放在常量池当中

String s1="jiajun";

String s7="jiajun";

System.out.println(s1==s7);

* 结果为true，都是存放在常量池

String s4="jia";

String s5=s4+"jun";

String s1="jiajun";

System.out.println(s1==s5);

* 结果为false，变量的值在运行的时候才确定，所以此时s5实际上是new一个对象

String s3="jia"+"jun";

String s1="jiajun";

System.out.println(s1==s3);

* 结果为true，此时s1 s3都是指向常量池一个string

String s1="jiajun";

String s8=new String("jia")+"jun";

System.out.println(s1==s8);

* 结果为false，此时s8同样是new出来一个对象