# Java虚拟机栈与本地方法栈

栈的大小控制参数是 -Xss。

Java虚拟机在栈中定义了两种异常，StackOverFlowError和OutOfMemoryError。当请求栈的深度大于java虚拟机所允许的最大深度则抛出StackOverFlowError；如果java虚拟机在栈扩展时，没有申请到足够的空间时，则抛出OutOfMemoryError。

StackOverFlowError：java虚拟机在运行中，调用方法时，都要创建栈帧，当栈的空间不够时就会产生StackOverFlowError。那么对应的解决方法就只能是调节-Xss参数，或者减少方法的调用，减小栈帧的大小两种方法。

OutOfMemoryError：在栈上出现OOM一般是多线程的情形。首先解析一下栈使用的空间可以有多大，拿32位操作系统来举例，最大内存2G -Xmx（最大堆容量） -MaxPermSize（最大方法区容量） -虚拟机本身耗费的内存和程序计数器使用的内存。剩下的内存就是栈可以使用的 空间，当Xss配置的参数一定时，那么在不断的创建线程的过程中，遇到不能申请到栈空间的时候 就会抛出OOM，那么对应的解决方式就是，调节-Xss参数降低栈大小，或者调节-Xmx以及MaxPermSize的大小扩大留给栈的空间。

# Java内存溢出定位和解决方案

引起内存溢出的原因 有很多种，列举一下常见的几种：

1. 内存种加载的数据量过于庞大，如一次从数据库取出过多数据
2. 集合类中有对对象的引用，使用完后未清空，使得JVM不能回收
3. 代码中存在死循环或循环产生过多重复的对象实体
4. 使用的第三方软件中的bug
5. 启动参数内存值设定的过小

# 内存溢出的解决方案

第一步，修改JVM启动参数，直接增加内存（-Xms，-Xms参数一定不要忘记加）

第二步，检查错误日志，查看“OutOfmemory”错误前是否有其他异常或错误

第三步，对代码进行走查和分析，找出可能发生内存溢出的位置

重点排查以下几点：

1. 检查对数据库查询中，是否有一次获得全部数据的查询。一般来说，如果一次取10万条记录到内存，就可能引起内存溢出。这个问题比较隐蔽，在上线前，数据库中的数据较少，不容易出问题，上线后，数据库中数据多了，一次查询就有可能引起内存溢出。因此对于数据库的查询尽量采用分页的方式查询。
2. 检查代码中是否有死循环或递归调用
3. 检查是否有大量循环重复产生新对象实体。
4. 检查List、Map等集合对象是否有使用完后未清除的问题。List、Map等集合对象会始终存有对对象的引用，使得这些对象不能被GC回收

第四步，使用内存查看工具动态查看内存的使用情况。