JAVA 虚拟机的多线程实现

通过线程轮流切换并分配处理器执行时间的方式。

虚拟机栈是JAVA方法执行的内存模型

每个方法在执行的同时都会创建一个栈帧用于存储局部变量表、操作数栈、动态链接、方法口等信息。

JAVA 内存溢出(Java.lang.outOfMemoryError)的常见情况和处理方式总结

导致OutOfMemoryError异常的常见原因有以下几种：

1. 内存中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. 使用第三方软件中的BUG；
5. 启动参数内存值设定的过小；

此错误常见的错误指示：

1. tomcat:java.lang.OutOfMemoryError:PermGen space
2. tomcat:java.lang.OutOfMemoryError:Java heap space
3. weblogic:Root cause of ServletException java.lang.OutOfMemoryError.
4. resin:java.lang. OutOfMemoryError
5. java:java.lang. OutOfMemoryError

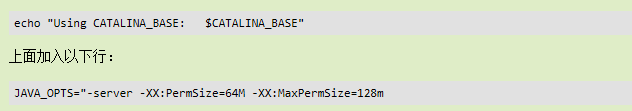
解决java.lang.OutOfMemoryError的方法有如下几种：

1. 增加jvm的内存大小。方法有：1）在执行某个class文件时候，可以使用java -Xmx256M aa.class来设置运行aa.class时jvm所允许占用的最大内存为256M。2）对tomcat容器，可以在启动时对jvm设置内存限度。对tomcat，可以在catalina.bat中添加：



或把%CATALINA\_OPTS%和%JAVA\_OPTS%代替为-Xms128M -Xmx256M。

1. 优化程序，释放垃圾。主要包括避免死循环，应该及时释放种资源：内存, 数据库的各种连接，防止一次载入太多的数据。导致java.lang.OutOfMemoryError的根本原因是程序不健壮。因此，从根本上解决Java内存溢出的唯一方法就是修改程序，及时地释放没用的对象，释放内存空间。
2. Java代码导致OutOfMemoryError错误的解决：
3. 检查代码中是否有死循环或递归调用。
4. 检查是否有大循环重复产生新对象实体。
5. 查询对数据库查询中，是否有一次获得全部数据的查询。
6. 检查List、Map等集合对象是否有使用完成后，未清除的问题。
7. tomcat中java.lang.OutOfMemoryError:PermGen space异常处理
8. 手动设置MaxPermSize大小修改TOMCAT\_HOME/bin/catalina.sh在



JVM垃圾收集算法

1. 标记-清除算法
2. 复制算法
3. 标记-整理算法。
4. 分代收集算法。

JVM垃圾收集器

1. Serial收集器(单线程)
2. ParNew 收集器(多线程)
3. CMS 收集器
4. G1 收集器

CLASS文件内容结构

1. 魔数
2. 次版本
3. 主版本
4. 常量池
5. 访问标志
6. 类索引、父类索引与接口索引
7. 字段表集合
8. 方法表集合
9. 属性表集合

类加载：

1. 加载：查找并加载类的二进制数据

1、通过一个类的全限定名来获取其定义的二进制字节流。

2、将这个字节流所代表的静态存储结构转化为方法区的运行时数据结构。

3、在JAVA堆中生成一个代表这个类的java.lang.Class对象，作为对方法区中这些数据的访问入口。

1. 验证：确保被加载的类的正确性
2. 文件格式验证：验证字节流是否符合Class文件格式的规范。
3. 元数据验证：对字节码描述的信息进行语义分析，以保证其描述的信息符合Java语言规范的要求。
4. 字节码验证：用过数据流和控制流分析，确定程序语义是否合法的，符合逻辑的。
5. 符号引用验证：确保解析动作能正确执行。
6. 准备：类的静态变量分配内存，并将其初始化为默认值
7. 内存分配的仅包括类变量(static)，而不包括实例变量，实例变量会在对象实例化时随着对象一块分配在Java堆中。
8. 初始化数据类型默认的零值(0、0L、null、false等)，而不是被在Java代码中被显示地赋予的值。
9. 解析：类中的符号引用转换为直接引用
10. 主要针对类或接口、字段、类方法、接口方法、方法类型、方法句柄和调用点限定符7类符号引用进行。
11. 符号引用就是一组符号来描述目标，可以是任何字面量。
12. 直接引用就是直接指向目标的指针、相对偏移量或一个间接定位到目标的句柄。
13. 初始化：类的静态变量赋值正确的初始值，JVM负责对类进行初始化，主要对类变量进行初始化
14. 声明类变量是指定初始值。
15. 使用静态代码块为类变量指定初始值。
16. 使用
17. 卸载