参考：

[JVM运行时内存区域（Red Code）]

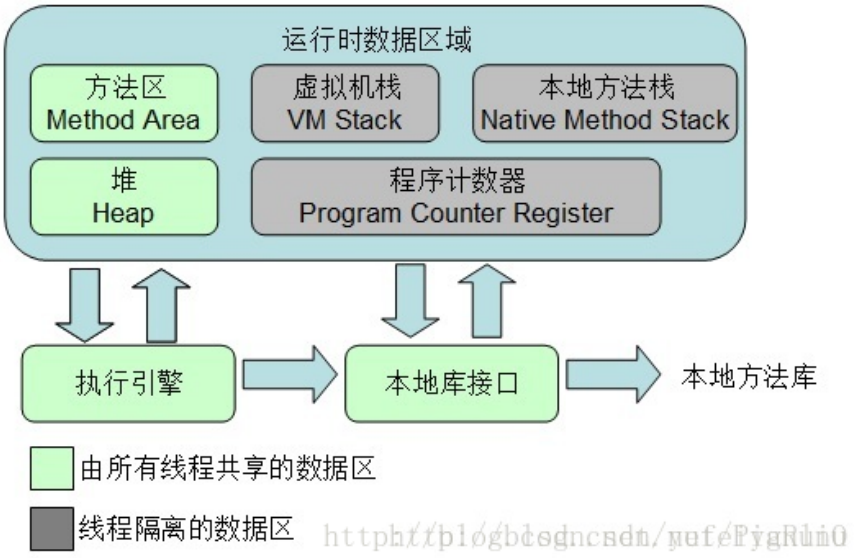
[JVM运行时内存区域（yufeiyanliu）]

[JVM内存区域的划分（内存结构或者内存模型）（ Meng）]

前言

JVM是学Java绕不开的话题，都知道它是在内存空间单独开辟出一个空间来的，所以能够跨平台。那么，这个空间里面到底有啥呢。

JVM在运行时会把内存划分成几个不同的数据区域。而这样的划分我们可以用这张图概括下。



程序计数器，虚拟机栈，本地方法栈，方法区，堆。一共五个区域。

详细介绍

1.程序计数器（线程私有）

程序计数器相当于JVM所执行的字节码（jvm指令）的“行号指示器”，通过程序计数器的“值”找到吓一跳需要执行的字节码指令。因

为线程私有，每个线程都有自己的程序计数器，此内存区域是唯一一个没有规定 “OutOfMemoryError”的区域。

2.虚拟机栈（线程私有）

虚拟机栈是一个“栈结构”的内存区域（先进后出）。里面存的是一个个“栈帧”。每个栈帧存储了局部变量表，操作数栈，动态连接，方法返回地址等。不过通常说的栈只指局部变量表，局部变量表所需内存会在编译期间完成分配。

举个例子。

如：fun A(){ fun B()}

```
public void A(){
     B();
     }
```

其栈帧的执行顺序就是：A栈帧先入栈，B栈帧再入栈，等方法B执行完后，B 栈帧再出栈，A栈帧再出栈。其整体过程也符合栈结构的“先进后出”。

该区域有两种异常：当请求的“栈深度”大于JVM虚拟机所允许的深度时，抛出“StackOverFlowError”。当内存不够时，抛出“OutOfMemmory”。

3.本地方法栈（线程私有）

本地方法栈类似于虚拟机栈，不同处在于，虚拟机栈为虚拟机运行的Java方法服务，本地方法栈为虚拟机使用的native方法服务。

4.堆（线程共享）

堆是JVM管理的最大的一块内存，是用来存放对象实例和数组的。同时堆也是垃圾回收器的主要管理区域。堆的大小是可以指定的，通过-Xmx和-Xms来控制。内存不够时会抛出“OutOfMemmory”。

5. 方法区（线程共享）

方法区用于存放“类的信息”，包括类中的：

类的全路径名、类的直接超类的全限定名、类的修饰符、类的“常量池”、类的“域信息”、类的“方法信息”、类的final常量，类的“所有static静态变量”。

常量池：

每个类都有自己的常量池。常量池是“同一个类所用常量的集合”。如：

```
Integer a = 1;
Integer b = 1;
//a和b指向Integer类的常量池中的同一个内存空间（因为他们的常量都是1）。

//同理
String s1 = "hello";
String s2 = "hello";
System.out.print(s1==s2);
//输出的是“true”，因为s1和s2指向String类的常量池中同一个内存空间。

//以下不是常量池存储
String s3 = new String("hello");
String s4 = new String("hello");
System.out.print(s1==s2);
//输出的是“false”，因为s3和s4是对象实例，存与堆中不同的空间。
```

域信息：

jvm必须在方法区中保存类型的所有域的相关信息以及域的声明顺序，　域的相关信息包括：　域名、域类型、域修饰符。

方法信息：

每一个类中都含有各种方法，所以方法区中会存储这些类中方法的相关信息，包括：

方法名、方法的返回类型、方法参数和类型、方法的修饰符。

类中的static静态变量：

static变量实际上属于类（并不属于某一个对象），所以自然这些static变量也属于“类的相关信息”。