调试排错 - Java OOM 分析

Java 堆内存溢出

在 Java 堆中只要不断的创建对象，并且 GC-Roots 到对象之间存在引用链，这样 JVM 就不会回收对象。

只要将-Xms(最小堆),-Xmx(最大堆) 设置为一样禁止自动扩展堆内存。

当使用一个 while(true) 循环来不断创建对象就会发生 OutOfMemory，还可以使用 -XX:+HeapDumpOutofMemoryErorr 当发生 OOM 时会自动 dump 堆栈到文件中。

伪代码:

```
    public static void main(String[] args) {
        List<String> list = new ArrayList<>(10) ;
        while (true){
            list.add("1") ;
        }
    }
```

当出现 OOM 时可以通过工具来分析 GC-Roots 引用链 (opens new window)，查看对象和 GC-Roots 是如何进行关联的，是否存在对象的生命周期过长，或者是这些对象确实改存在的，那就要考虑将堆内存调大了。

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
    at java.util.Arrays.copyOf(Arrays.java:3210)
    at java.util.Arrays.copyOf(Arrays.java:3181)
    at java.util.ArrayList.grow(ArrayList.java:261)
    at java.util.ArrayList.ensureExplicitCapacity(ArrayList.java:235)
    at java.util.ArrayList.ensureCapacityInternal(ArrayList.java:227)
    at java.util.ArrayList.add(ArrayList.java:458)
    at com.crossoverjie.oom.HeapOOM.main(HeapOOM.java:18)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at com.intellij.rt.execution.application.AppMain.main(AppMain.java:147)

Process finished with exit code 1
```

java.lang.OutOfMemoryError: Java heap space表示堆内存溢出。

MetaSpace (元数据) 内存溢出

> JDK8 中将永久代移除，使用 MetaSpace 来保存类加载之后的类信息，字符串常量池也被移动到 Java 堆。

PermSize 和 MaxPermSize 已经不能使用了，在 JDK8 中配置这两个参数将会发出警告。

JDK 8 中将类信息移到到了本地堆内存(Native Heap)中，将原有的永久代移动到了本地堆中成为 MetaSpace ,如果不指定该区域的大小，JVM 将会动态的调整。

可以使用 -XX:MaxMetaspaceSize=10M 来限制最大元数据。这样当不停的创建类时将会占满该区域并出现 OOM。

```
    public static void main(String[] args) {
        while (true){
            Enhancer  enhancer = new Enhancer() ;
            enhancer.setSuperclass(HeapOOM.class);
            enhancer.setUseCache(false) ;
            enhancer.setCallback(new MethodInterceptor() {
                @Override
                public Object intercept(Object o, Method method, Object[] objects, MethodProxy
methodProxy) throws Throwable {
                    return methodProxy.invoke(o,objects) ;
                }
            });
            enhancer.create() ;

        }
    }
```

使用 cglib 不停的创建新类，最终会抛出:

```
Caused by: java.lang.reflect.InvocationTargetException
    at sun.reflect.GeneratedMethodAccessor1.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at net.sf.cglib.core.ReflectUtils.defineClass(ReflectUtils.java:459)
    at net.sf.cglib.core.AbstractClassGenerator.generate(AbstractClassGenerator.java:336)
    ... 11 more
Caused by: java.lang.OutOfMemoryError: Metaspace
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:763)
    ... 16 more
```

注意: 这里的 OOM 伴随的是 java.lang.OutOfMemoryError: Metaspace 也就是元数据溢出。