如何得知某个web请求所消耗的内存。

答案很简单，请求前，将内存给导出来，记录下大小。

请求后再把内存给导出来，记录下大小。

两者的差值即为这次请求消耗的内存大小。

本文描述了具体操作过程。

1.

名词缩写

A:不可回收对象所用的内存大小。不可回收对象指存在引用指向这个对象,暂时不能被垃圾回收。

B:可回收对象所用的内存大小。可回收对象指没有任何引用指向这个对象，可被垃圾回收，还没被回收，还留在内存中。

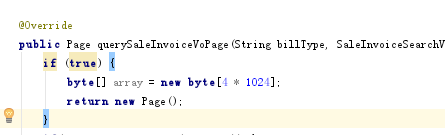
验证过程：

1. 找一个方法，分配4K内存后返回，计算访问前访问后消耗的内存差值，赋值为M1。
2. 修改这个方法，分配1M内在后返回，计算访问前访问后消耗的内存差值，赋值为M2。

如果 M2-M1 刚好为 1M 左右，那么说明这种计算过程可行。

(注:不能用debug启动相应进程，debug误差很大)

举例说明:



1. 在SaleInvoiceService这个方法中,分配了4K内存后直接返回。
2. 运行命令 jmap -dump:format=b,file=begin\_0.bin [pid],导内存，得到文件 begin0.bin
3. 浏览器访问对应url后，使得其调用这个Service方法
4. 运行命令 jmap -dump:format=b,file=begin\_0\_end.bin [pid],导内存，得到文件 begin\_0\_end.bin

重复执行6次，可得到文件列表：

2018/04/27 13:42 302,489,380 begin\_0.bin

2018/04/27 13:42 302,812,514 begin\_0\_end.bin

2018/04/27 13:42 302,813,163 begin\_1.bin

2018/04/27 13:42 303,132,658 begin\_1\_end.bin

2018/04/27 13:42 303,133,541 begin\_2.bin

2018/04/27 13:42 303,453,231 begin\_2\_end.bin

2018/04/27 13:42 303,455,322 begin\_3.bin

2018/04/27 13:42 303,775,158 begin\_3\_end.bin

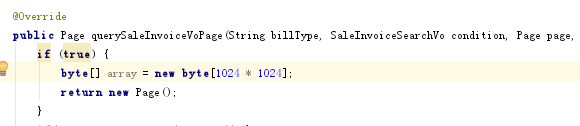
2018/04/27 13:42 303,776,333 begin\_4.bin

2018/04/27 13:42 304,096,501 begin\_4\_end.bin

2018/04/27 13:42 304,097,676 begin\_5.bin

2018/04/27 13:42 304,418,392 begin\_5\_end.bin

接下来，修改service方法，使得其分配1M后返回，代码如下：



重新访问前，访问后导内存，得到文件列表：

2018/04/27 11:40 560,179,944 begin\_0.bin

2018/04/27 11:40 561,495,128 begin\_0\_end.bin

2018/04/27 11:40 561,497,643 begin\_1.bin

2018/04/27 11:40 562,813,076 begin\_1\_end.bin

2018/04/27 11:41 562,817,374 begin\_2.bin

2018/04/27 11:41 564,135,626 begin\_2\_end.bin

2018/04/27 11:41 565,443,483 begin\_3.bin

2018/04/27 11:41 566,761,709 begin\_3\_end.bin

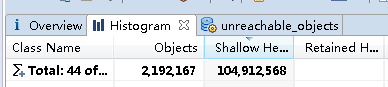
2018/04/27 11:41 566,768,902 begin\_4.bin

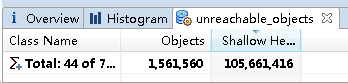
2018/04/27 11:41 568,084,470 begin\_4\_end.bin

2018/04/27 11:42 568,086,488 begin\_5.bin

2018/04/27 11:42 569,403,324 begin\_5\_end.bin

接下来，用eclipse memory mat 分析这些文件，计算每个文件的访问前后的内存差值。





104912568 + 105661416 = 210573984

当分配4K内存时，可得图表1。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 访问前A+B(单位B) | 访问后A+B(单位B) | 差值(单位m) |
| 1 | 210573984 | 210884256 | 0.295898438 |
| 2 | 210884520 | 211194704 | 0.295814514 |
| 3 | 211195000 | 211505288 | 0.295913696 |
| 4 | 211505944 | 211816312 | 0.29598999 |
| 5 | 211816768 | 212127016 | 0.295875549 |
| 6 | 212127472 | 212437840 | 0.29598999 |

当分配1M内存时，可得以下图表2。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 访问前A+B(单位B) | 访问后A+B(单位B) | 差值(m) |
| 1 | 444307280 | 445612256 | 1.244522095 |
| 2 | 445613480 | 446918576 | 1.244636536 |
| 3 | 446920592 | 448225648 | 1.244598389 |
| 4 | 449530648 | 450835680 | 1.2445755 |
| 5 | 450838960 | 452143816 | 1.244407654 |
| 6 | 452144432 | 453449328 | 1.244445801 |

即分配4K内存时，每次访问消耗了大约 0.29 M内存。

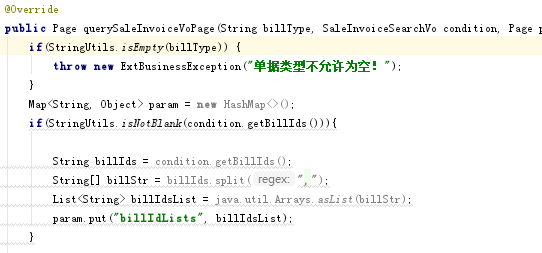
分配1M内存时，每次访问消耗了大约 1.24 M内存。

差额大约在1M左右。

即这种计算内存差额的方法，还算可靠。

运用这种方法，即可得出访问销售发票列表这个方法所消耗的内存。

访问下面这个方法前后，导一下内存数据，



可以得到文件列表：

2018/04/27 13:50 323,422,581 begin\_0.bin

2018/04/27 13:50 328,247,151 begin\_0\_end.bin

2018/04/27 13:50 328,886,527 begin\_1.bin

2018/04/27 13:50 332,705,513 begin\_1\_end.bin

2018/04/27 13:50 332,707,324 begin\_2.bin

2018/04/27 13:50 337,601,667 begin\_2\_end.bin

2018/04/27 13:50 337,602,915 begin\_3.bin

2018/04/27 13:51 341,426,853 begin\_3\_end.bin

2018/04/27 13:51 341,427,882 begin\_4.bin

2018/04/27 13:51 345,245,682 begin\_4\_end.bin

2018/04/27 13:51 345,246,565 begin\_5.bin

2018/04/27 13:51 349,072,248 begin\_5\_end.bin

内存分析后，可得到以下图表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 访问前A+B(单位B) | 访问后A+B(单位B) | 差值(m) |
| 1 | 228979680 | 232998688 | 3.832824707 |
| 2 | 233635512 | 236757200 | 2.977073669 |
| 3 | 236757704 | 240941992 | 3.990447998 |
| 4 | 240942488 | 244072048 | 2.984580994 |
| 5 | 244072424 | 247195408 | 2.978309631 |
| 6 | 247195704 | 250318976 | 2.97858429 |

即每次请求销售发票列表，消耗了 2.9M~3.9M左右的内存。