



Linux DevOps Lab

JAVA PLATFORM

05_Extras. Spring Boot

Home task

Legal Notice: This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

CONFIDENTIAL | Effective Date: 25-Jan-19

PREREQUISITES

Oracle JDK 1.8 installed

GOAL

Understand principals of jmx and rmi

TASK

- 1) Deploy TestApp.war with next parameters: - Done
 - o jmx ports 12345 12346 - Done
 - o connect via jconsole and retrieve heap and thread count information - Done

Java Monitoring & Management Console

Connection: 192.168.56.29:12345

Overview | Memory | Threads | Classes | **VM Summary** | MBeans

VM Summary
Thursday, February 13, 2020 4:18:59 PM MSK

Connection name: 192.168.56.29:12345	Uptime: 1 minute
Virtual Machine: Java HotSpot(TM) 64-Bit Server VM version 25.241-b07	Process CPU time: 2.810 seconds
Vendor: Oracle Corporation	JIT compiler: HotSpot 64-Bit Tiered Compilers
Name: 4889@kazak-java	Total compile time: 1.741 seconds

Live threads: 33	Current classes loaded: 3,432
Peak: 33	Total classes loaded: 3,432
Daemon threads: 32	Total classes unloaded: 0
Total threads started: 36	

Current heap size: 57,183 kbytes	Committed memory: 251,392 kbytes
Maximum heap size: 466,432 kbytes	Pending finalization: 0 objects
Garbage collector: Name = 'PS MarkSweep', Collections = 0, Total time spent = 0.000 seconds	
Garbage collector: Name = 'PS Scavenge', Collections = 1, Total time spent = 0.010 seconds	

Operating System: Linux 3.10.0-957.27.2.el7.x86_64	Total physical memory: 1,882,132 kbytes
Architecture: amd64	Free physical memory: 703,660 kbytes
Number of processors: 2	Total swap space: 2,097,148 kbytes
Committed virtual memory: 3,055,296 kbytes	Free swap space: 2,097,148 kbytes

VM arguments: -Djava.util.logging.config.file=/opt/tomcat/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=12345 -Dcom.sun.management.jmxremote.rmi.port=12346 -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Djava.rmi.server.hostname=192.168.56.29 -Djava.awt.headless=true -Djdk.tls.ephemeralDHKeySize=2048 -Djava.protocol.handler.pkgs=org.apache.catalina.webresources -Dorg.apache.catalina.security.SecurityListener.UMASK=0027 -Xms256m -Xmx512m -Dignore.endorsed.dirs= -Dcatalina.base=/opt/tomcat -Dcatalina.home=/opt/tomcat -Djava.io.tmpdir=/opt/tomcat/temp

Class path: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar

Library path: /usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib

192.168.56.29:12345

- o write simple java application to gather Tomcat heap information and thread count - Done

- 2) Press **Parse employees with memory leak** button and upload JSON-file with employees info - Done
- 3) Run application before and after parsing, describe what you see. - Done

```
student@localhost ~/devopstraining/java/java301/appl <master*>
$ java GatherInfo
Number of Threads:      31
Init Heap Memory:      256M
Used Heap Memory:      41M
Committed Heap Memory: 245M
Max Heap Memory:       455M
```

Information is gathered correctly corresponding to the output of jconsole. 41M of heap memory used is ok.

```
student@localhost ~/devopstraining/java/java301/appl <master*>
$ java GatherInfo
Number of Threads:      30
Init Heap Memory:      256M
Used Heap Memory:      464M
Committed Heap Memory: 509M
Max Heap Memory:       509M
```

Once MemoryLeak Submit button is pressed, we could see Heap Memory usage is coming to it's highest value and the GC isn't managing to clean everything just in time. 464M of heap memory used is definitely NOT ok. Heap Memory usage will grow up to the maximum value and then application will throw an OutOfMemoryError.

Application do works!