



Linux DevOps Lab

JAVA PLATFORM

04_Garbage Collection. Memory Dumps.

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

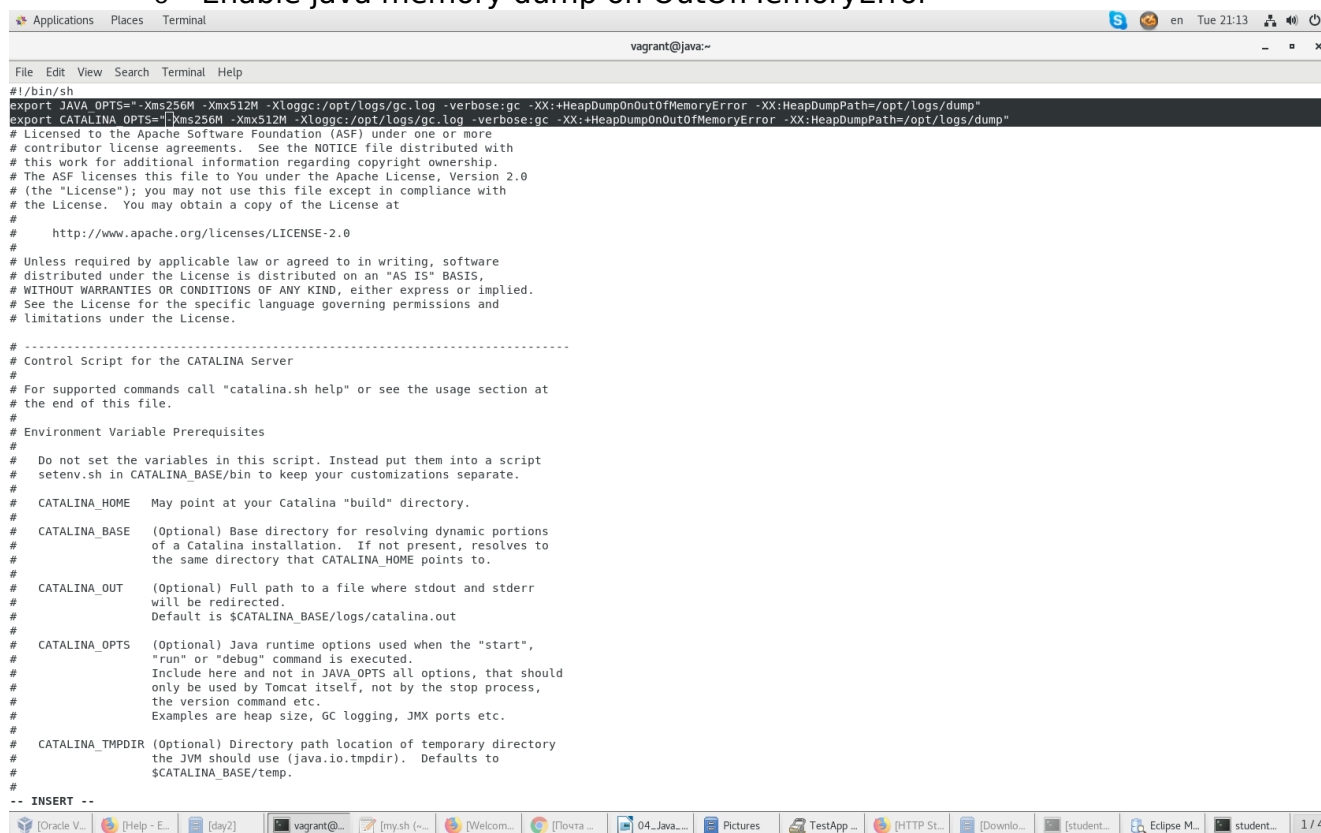
Oracle JDK 1.8 installed

GOAL

Understand principles of jvm heap dump analysis and memory leak detection

TASK

- 1) Deploy TestApp.war with next parameters:
 - o Initial heap size 256 mb
 - o Max heap size 512 mb
 - o Enable GC logs: -verbose:gc
 - o Setup path to file with GC logs: -Xloggc:***path to file***
 - o Enable java memory dump on OutOfMemoryError



```

#!bin/sh
export JAVA_OPTS="-Xms256M -Xmx512M -Xloggc:/opt/logs/gc.log -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/logs/dump"
export CATALINA_OPTS="-Xms256M -Xmx512M -Xloggc:/opt/logs/gc.log -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/logs/dump"
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# -----
# Control Script for the CATALINA Server
#
# For supported commands call "catalina.sh help" or see the usage section at
# the end of this file.
#
# Environment Variable Prerequisites
#
# Do not set the variables in this script. Instead put them into a script
# setenv.sh in CATALINA_BASE/bin to keep your customizations separate.
#
# CATALINA_HOME May point at your Catalina "build" directory.
#
# CATALINA_BASE (Optional) Base directory for resolving dynamic portions
# of a Catalina installation. If not present, resolves to
# the same directory that CATALINA_HOME points to.
#
# CATALINA_OUT (Optional) Full path to a file where stdout and stderr
# will be redirected.
# Default is $CATALINA_BASE/logs/catalina.out
#
# CATALINA_OPTS (Optional) Java runtime options used when the "start",
# "run" or "debug" command is executed.
# Include here and not in JAVA_OPTS all options, that should
# only be used by Tomcat itself, not by the stop process,
# the version command etc.
# Examples are heap size, GC logging, JMX ports etc.
#
# CATALINA_TMPDIR (Optional) Directory path location of temporary directory
# the JVM should use (java.io.tmpdir). Defaults to
# $CATALINA_BASE/temp.
#
-- INSERT --

```

- 2) Press **Parse employees with memory leak** button and upload JSON-file with employees info several times (2-3 times until OutOfMemoryError)

- 3) Analyze generated heap dump using [Memory Analyzer \(MAT\)](#)
 - o Generate Leak Suspect Report

The screenshot shows the Eclipse Memory Analyzer (MAT) interface. The 'Leak Suspects' tab is selected, displaying a list of memory leaks. The top entry is for 'com.epam.nix.java.testapp.servlet.MemoryLeakServlet' with a retention size of 510,990,496 bytes (98.55%). The 'Shortest Paths To the Accumulation Point' table is visible below the description.

Class Name	Shallow Heap	Retained Heap
java.lang.Object[2734845] @ 0xe0000000	10,939,400	510,990,448
java.util.ArrayList @ 0xeb024658	24	510,990,472
com.epam.nix.java.testapp.servlet.MemoryLeakServlet @ 0xeaff6f38	24	510,990,496
org.apache.tomcat.util.threads.TaskThread @ 0xeaff7758 http-nio-8080-exec-7 Thread	128	35,120
org.apache.tomcat.util.threads.TaskThread @ 0xeaff79d8 http-nio-8080-exec-4 Thread	128	30,968
org.apache.catalina.core.ApplicationFilterChain @ 0xe98ce868	32	88
org.apache.catalina.core.StandardWrapper @ 0xeab04638	240	1,656
org.apache.catalina.core.ApplicationFilterChain @ 0xeaff6f88	32	88
Total: 5 entries		

сообщает нам, что объект из com.epam.nix.java.testapp.servlet.MemoryLeakServlet занимает 98 процентов памяти.

The screenshot shows the Eclipse Memory Analyzer (MAT) interface with the 'Dominator Tree' tab selected. The tree shows the hierarchy of objects in memory, with 'com.epam.nix.java.testapp.servlet.MemoryLeakServlet' at the top, accounting for 98.55% of the retained heap.

Class Name	Shallow Heap	Retained Heap	Percentage
com.epam.nix.java.testapp.servlet.MemoryLeakServlet @ 0xeaff6f38	24	510,990,496	98.55%
java.util.ArrayList @ 0xeb024658	24	510,990,472	98.55%
java.lang.Object[2734845] @ 0xe0000000	10,939,400	510,990,448	98.55%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf50c0	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5238	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf53b0	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5528	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf56a0	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5818	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5990	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5b08	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5c80	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5df8	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf5f70	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf60e8	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf6260	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf63d8	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf6550	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf66c8	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf6840	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf69b8	32	192	0.00%
com.epam.nix.java.testapp.entity.Employee @ 0xe0cf6b30	32	192	0.00%

- o Using Dominator Tree find why objects was not collected by garbage collector

Здесь видим что объект который мы передаем программе а именно Employee, сохраняется в объект ArrayList который постоянно разрастается. В итоге на протяжении всего цикла работа программы в com.epam.nix.java.testapp.servlet.MemoryLeakServlet существует ссылка на ArrayList, от которого есть ссылки на каждый объект Employee. В результате объекты GC не вычищаются, а накапливаются до OutofMemory. Для фикса этой проблемы можно попробовать убрать ArrayList или в конце удалять объекты entity, или поместить его в локальную переменную, чтобы он удалялся после выполнения метода.

- 4) Analyze GC logs using [GCViewer](#) [Describe in few words result.](#)

Наблюдаем, что наш хип заполнен. GC со временем вообще ничего не вычищает. Продолжительность чистки увеличивается. Видим что выполняется в основном только FullGC. 50% времени работы программы занимает чистка, еще 50 Full GC - Allocation Failure, что говорит нам о том, что приложение не работает корректно. Что соответствует заключению сделанному в предыдущем пункте.

