# **Java Profilers**

## Overview:

In the development world, It seems that writing and running a code successfully is not enough, basically we need to aim for the best and most efficient code writing skillsets.

But to work with the large and complex application, it's not an easy task to manually perform and monitor how the application is consuming the memory and how the performance is not up to the mark.

To perform this a java developer, need to setup some tools for maintain these activities for all parameters with all permutation and combinations. Those tools basically called here Java Profilers.

Introduction:

Java Profilers and Application Performance Management (APM) Tools are providing all needs for monitoring and debugging performance issues in development and production environments.

Basically, a Java Profiler tool is that monitors Java bytecode constructs and operations at the JVM level.

These code constructs and operations include object creation, iterative executions (including recursive calls), method executions, thread executions, and garbage collections.

## Types:

Basically, there are three different types of java profilers:

**1. Standard JVM profilers**

It can track every detail of the JVM (CPU, thread, memory, garbage collection, etc).

It certainly provides the most data, but not necessarily the most useful information. This depends on the type of debugging task. It is very good for tracking CPU usage and a CPU sampling feature to track and aggregate CPU time by class and method to help zero in on hot spots.

**2. Lightweight profilers**

This kind of profilers basically highlight your application with a bit of abstraction. It takes a different approach at tracking your application by injecting themselves right into the code.

Aspect Profilers use aspect-oriented programming (AOP) to inject code into the start and end of specified methods. Related Products are: XRebel and Stackify

**3. Application performance management (APM)**

These tools are used for monitoring applications live in production environments.

These tools are typically use the Java Agent profiler method but with different instrumentation rules to allow them to run without affecting performance in productions.

The trick with these profilers is to provide the right information in a smart way to not take up CPU cycles.

Examples:

JProfiler, YourKit, VisualVM, NetBeans Profiler, XRebel, JProbe etc.

## Graphical user interface Description automatically generatedSurvey:

Calendar

Description automatically generated

## 1. VisualVM

Java is providing an excellent build-in monitoring tool to do this job very easily.

VisualVM is a powerful tool that provides a visual interface to see deep and detailed information about local and remote Java applications while they are running on a Java Virtual Machine (JVM).

Java VisualVM is Java’s built-in monitoring utility packaged inside JDK/bin directory.

* GUI tool shipped with standard JDK (no installation/setup process)
* Useful for monitoring CPU sampling, memory sampling, run garbage collections, analyze heap errors, take snapshots and more using GUI
* Can develop new functionalities as plugins
* Supports local and remote profiling (does not provide support for ssh tunneling, need to configure JMX ports for remote profiling)
* Can take snapshots of profiling sessions for later analysis.

Steps to setup VisualVM in eclipse/STS:

Step-1: First we need to check in c:-> ProgramFiles->java->jdk<version> -> bin -> jvisualvm.exe is present or not.

A picture containing text

Description automatically generated

Step-2: Now we need to open eclipse/sts and from Menu Browse Help-> Install New Software -> Browse the folder where we have extracted VisualVM eclipse plugins.

![Graphical user interface, application

Description automatically generated]()

Step-3: Now we need to select VisualVM Launcher Feature, which will show after extracting Uncategorized.

![Graphical user interface, application

Description automatically generated]()

Step-4: Now we need to configure it to launch the VisualVM application Under Windows-> Preferences -> Run/Debug -> Launching -> VisualVM Configuration.

Step-5: Now we need to select VisualVM Executable file which will be present in JAVA\_HOME/bin directory. And to just ensure that Jdk Home field is showing the current JAVA\_HOME directory.

![Graphical user interface, text, application, email

Description automatically generated]()

Now we will write a sample java program where we can monitor Memory Leak, Heap Data usages, Garbage collector and CPU profiling.

Here I have created one main class AssignmentApplication where I have creating new Employee object inside the loop and that we are not going to use any more, but this is unused object, and this is not eligible for garbage collector, and this will cause harm in our project. So, this point we are going to Monitor through VisualVm.

* ![Graphical user interface, application

  Description automatically generated]()Debugging the application using VisualVM and will watch step by step.

![Graphical user interface, application

Description automatically generated]()

* First, we will see and monitor when our application got started.

![Graphical user interface, text, application

Description automatically generated]()

* First, we will see The Process-Id where the application is running, in overview section it will show Host, Port, Main class.

Graphical user interface, text, application, email

Description automatically generated

* Graphical user interface, application, table, Excel

  Description automatically generatedIn Monitor section where we can see Garbage Collector activity, Heap Data and Meta Space, Class Loader Details, and the last one will show the number of threads.
* In Heap Section the Blue one is showing Heap space and in Heap Dump all classes and objects it will show how much memory it is occupying.

![Graphical user interface, text, application

Description automatically generated]()

* Now we move our debug pointer and see the effect.

Graphical user interface, application, table, Excel

Description automatically generated

* There is one option in Heap section and that called “Perform GC”, that is responsible to perform

Graphical user interface, table

Description automatically generatedGarbage collector activity. Which will remove unused created object.

* Based on this we can monitor and see each activity that is going to harm our application by using CPU monitoring, Heap Memory, Meta Space, Threads etc.

Chart, bar chart

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated