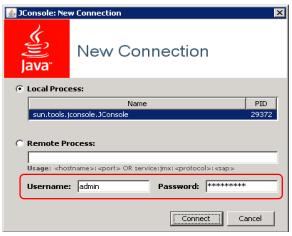
How to use Jconsole

by John
2015-01-14

1. Starting JConsole

The jconsole executable can be found in JDK_HOME/bin, where JDK_HOME is the directory in which the Java Development Kit (JDK) is installed

2. Creating a Connection to a Local Process



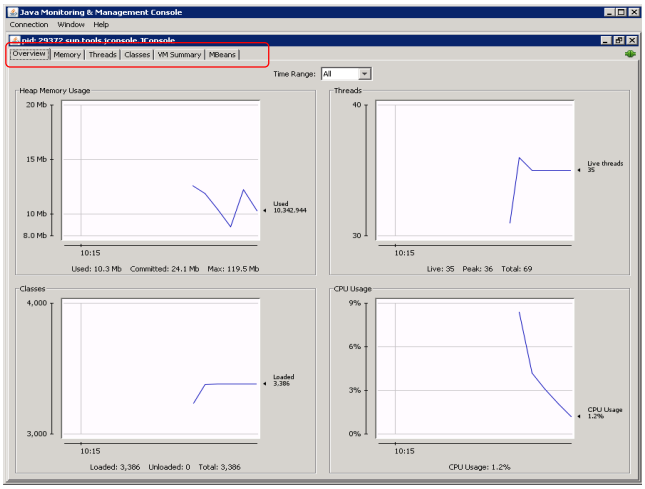
Username and password could be added in conf/tomcat-users.xml in your installation, if you've never setup Tomcat user.
For example, to add the manager-gui role to a user named tomcat with a password of s3cret, add the following to the conf/tomcat-users.xml.

<role rolename="manager-gui"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>

Note: Please restart Tomcat after adding user.

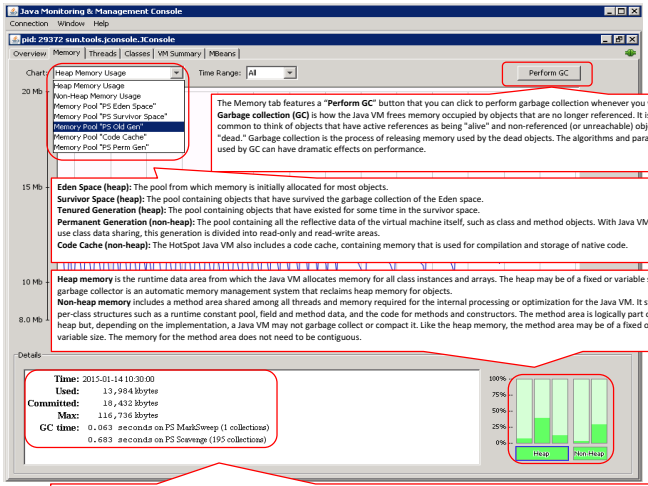
3. Viewing Overview Information

The Overview tab displays graphical monitoring information about CPU usage, memory usage, thread counts, and the classes loaded in the Java VM, all in a single screen.



3.1 Monitoring Memory Consumption

The Memory tab provides information about memory consumption and memory pools.



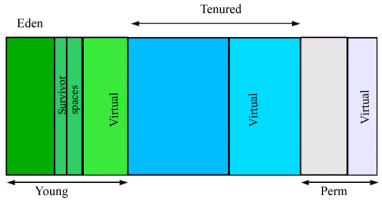
The Memory tab features a "**Perform GC**" button that you can click to perform garbage collection whenever you want.
Garbage collection (GC) is how the Java VM frees memory occupied by objects that are no longer referenced. It is common to think of objects that have active references as being "alive" and non-referenced (or unreachable) objects as "dead." Garbage collection is the process of releasing memory used by the dead objects. The algorithms and parameters used by GC can have dramatic effects on performance.

Eden Space (heap): The pool from which memory is initially allocated for most objects.
Survivor Space (heap): The pool containing objects that have survived the garbage collection of the Eden space.
Tenured Generation (heap): The pool containing objects that have existed for some time in the survivor space.
Permanent Generation (non-heap): The pool containing all the reflective data of the virtual machine itself, such as class and method objects. With Java VMs that use class data sharing, this generation is divided into read-only and read-write areas.
Code Cache (non-heap): The HotSpot Java VM also includes a code cache, containing memory that is used for compilation and storage of native code.

Heap memory is the runtime data area from which the Java VM allocates memory for all class instances and arrays. The heap may be of a fixed or variable size. The garbage collector is an automatic memory management system that reclaims heap memory for objects.
Non-heap memory includes a method area shared among all threads and memory required for the internal processing or optimization for the Java VM. It stores per-class structures such as a runtime constant pool, field and method data, and the code for methods and constructors. The method area is logically part of the heap but, depending on the implementation, a Java VM may not garbage collect or compact it. Like the heap memory, the method area may be of a fixed or variable size. The memory for the method area does not need to be contiguous.

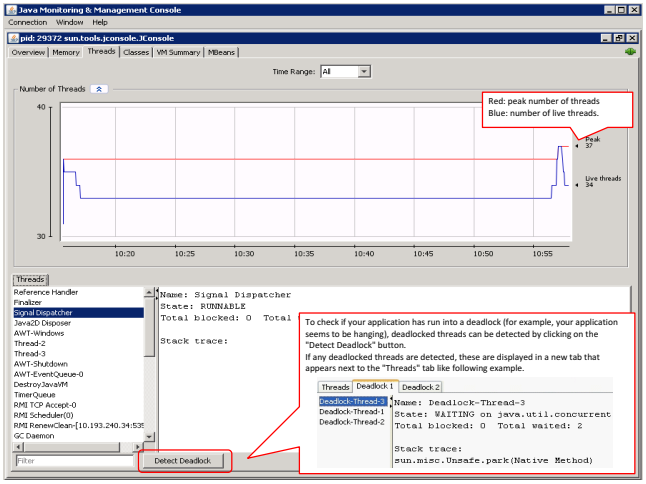
Used: the amount of memory currently used, including the memory occupied by all objects, both reachable and unreachable.
Committed: the amount of memory guaranteed to be available for use by the Java VM. The amount of committed memory may change over time. The Java virtual machine may release memory to the system and the amount of committed memory could be less than the amount of memory initially allocated at start up. The amount of committed memory will always be greater than or equal to the amount of used memory.
Max: the maximum amount of memory that can be used for memory management. Its value may change or be undefined. A memory allocation may fail if the Java VM attempts to increase the used memory to be greater than committed memory, even if the amount used is less than or equal to max (for example, when the system is low on virtual memory).
GC time: the cumulative time spent on garbage collection and the total number of invocations. It may have multiple rows, each of which represents one garbage collector algorithm used in the Java VM.

3.1.1 Generations of Data in Garbage Collection

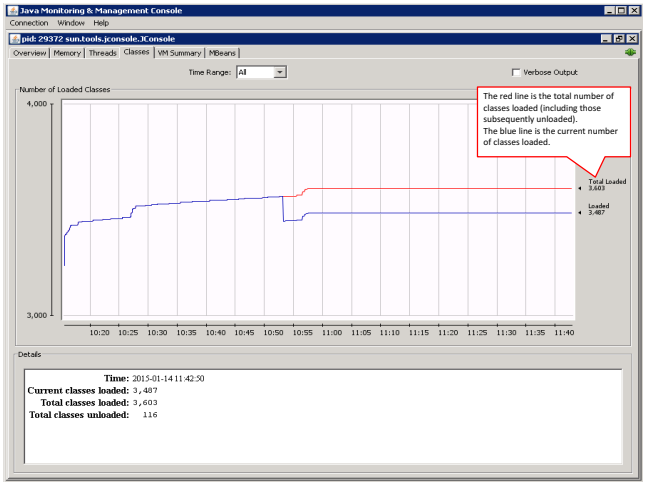


3.2 Monitoring Class Loading

The Classes tab displays information about class loading.



3.3 Monitoring Class Loading



3.4 Viewing VM Information

The VM Summary tab provides information about the Java VM.