Monitoring

Permalink to this page: https://cwiki.apache.org/confluence/x/eColBg

Table of Contents

Monitoring Tomcat

Monitoring of a running Tomcat instance can be done in several ways, but observing a Tomcat instance via JMX beans will give you the best information available through standard interfaces (i.e. JMX). You can find information about connecting to Tomcat via JMX in the Tomcat Users' Guide. Rather than repeating that information here (which is mostly about configuration, connection, etc.), please go read the official documentation.

This page is intended to be a community-curated collection of useful JMX beans that may be useful to you *after* you have made your JMX connection and want to observe interesting data through Tomcat's (and other) JMX beans.

JVM Information

Heap and other Memory Information

You will certainly want to inspect your JVM's memory usage. Here are some JMX beans and attributes that can be used to do so.

- JMX Bean: `java.lang:type=Memory`
- Attribute: `HeapMemoryUsage`

The attribute value is a `javax.management.openmbean.CompositeData` which contains 4 keys: committed, init, max, and used. The 'used' key is probably the most useful (or a combination of 'used' / 'max' to get a memory-usage metric as a ratio).

- JMX Bean: `java.lang:type=MemoryPool,name=CMS Perm Gen`
- Attribute: `Usage`

Similar to the `HeapMemoryUsage` MXBean described above, this one will give you information about the "`PermGen`" heap generation. Depending upon your garbage collection and other memory settings, you might have different MXBeans under `java.lang:type=MemoryPool` with different names. You should inspect each one to determine if they would be useful for you to inspect.

Tomcat Information

Version Warning

These JMX bean names are accurate for the current version of Tomcat 7 (7.0.37 at the time of this writing). If you are using a different version of Tomcat, you may have to adjust the names of the beans identified on this page.

Thread Usage

- JMX Bean: `Catalina:type=Executor,name=[executor name]`
- Attributes: `poolSize`, `activeCount`

This is the number of threads currently in the executor's thread pool. Obviously, this is only useful if you are using an <Executor> (which you *are* using, of course, right?).

Not using an Executor

- JMX Bean: `Catalina: type=ThreadPool,name="[depends]"`
- Attributes: `maxThreads`, `connectionCount`

This information is largely useless in Tomcat 7, as an Executor is always used and the data can be found there, while the ThreadPool has only initial configuration information: the real-time data is available from the Executor's MBean.

Request Throughput

- JMX Bean: `Catalina:type=GlobalRequestProcessor,name="[depends]"`
- Attributes: `bytesSent`, `bytesReceived`, `errorCount`, `maxTime`, `requestCount`
- Operations: `resetCounters`

Sessions

- JMX Bean: `Catalina:type=Manager,context=[context name],host=[hostname]`
- Attributes: `activeSessions`, `sessionCounter`, `expiredSessions`

JNDI DataSource

- JMX Bean: `Catalina:type=DataSource,context=[context name],host=[hostname],class=javax.sql.DataSource,name="[JNDI name]"`
- Attributes: `numActive`, `numIdle`

External Monitoring Tools

Another way to watch a Tomcat application is to use an external monitoring tool.

MoSKito, is an open source, multi-purpose, non-invasive, interval-based monitoring system kit that collects, stores and provides instant analysis of a Tomcat application's performance and behavior data.

JavaMelody can monitor your JavaEE/Tomcat application from dev to production. It is open-source and easy: get the first view of your application in about 2 minutes from now.

Other plug-in-based monitoring software like Nagios or Icinga may need some help interacting with Tomcat's JMXProxyServlet. tools/check_jmxproxy.pl is a Perl script that can be used with these tools to monitor Tomcat via the JMXProxyServlet.

Exposing Tomcat application internals using JMX

It is possible to write custom MBeans that expose your Tomcat application internals through JMX. User-defined MBeans need to be registered with Tomcat MBean server. Once registered, user can access MBean and observe user-defined attributes and call user-defined operations, e.g.

- JMX Bean: `Counter:type=CounterManager`
- Attributes: `currentTime`, `currentCount`
- Operations: `resetCounter`, `resetCounter(int)`

Example Application Exposing Internals Using JMX