

Monitoring the Performance of Your Tomcat Application Server Using JavaMelody

JavaMelody is an opensource (LGPL) application to monitor Java or Java EE application servers in QA and production environments.

JavaMelody is mainly based on statistics of requests and on evolution charts.

(Extract from the Javamelody home page)

- It allows to improve applications in QA and production
- Give facts about the average response times and number of executions
- Make decisions when trends are bad, before problems become too serious
- Optimize based on the more limiting response times
- Find the root causes of response times
- Verify the real improvement after optimization

It includes summary charts showing the evolution over time of the following indicators:

- Number of executions, mean execution times and percentage of errors of http requests, sql requests, jsp pages or methods of business façades (if EJB3, Spring or Guice)
- Java memory
- Java CPU
- Number of user sessions
- Number of jdbc connections

These charts can be viewed on the current day, week, month, year or custom period.
You can even execute garbage collection to free resources, or view / invalidate http sessions.

Setting up Javamelody on your server

*Step 1: *

Download the [javamelody-1.36.0.zip](#) zip file.

Now, unzip, and add copy the files javamelody.jar and jrobin-x.jar into the in the lib directory of Tomcat.

*Step 2: *

Add the following lines in the web.xml file of the conf directory of Tomcat (and not in the WEB-INF/web.xml files of the webapps).

web.xml.snippet

```
<filter>
  <filter-name>monitoring</filter-name>
  <filter-class>net.bull.javamelody.MonitoringFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>monitoring</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<listener>
  <listener-class>net.bull.javamelody.SessionListener</listener-class>
</listener>
```

Once this is done, all you need to do is restart the application. Once the application is up, navigate to <http://<host>/<context>/monitoring> url to view a whole set of interesting data.

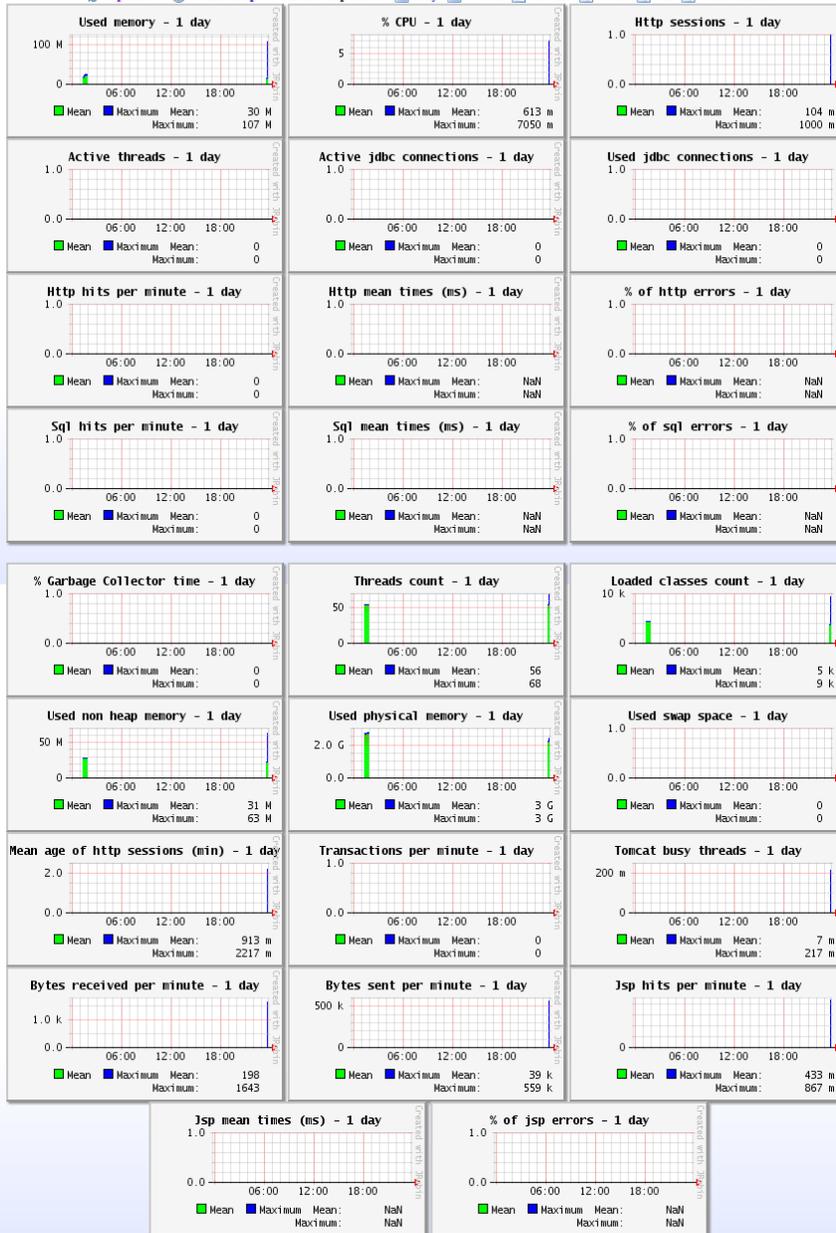
The full instructions are available [here](#).

What does the Javamelody monitoring page look like?

Attached below is a screenshot of my Javamelody monitoring page. Note that this screenshot is of a development environment with no data and just a single user.

Statistics of JavaMelody monitoring taken at 4/14/12 11:43 PM on _HP-PC (Welcome to Tomcat)

Update Online help Choice of period: Day Week Month Year All Customized



Other charts

Statistics http - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	% of system error	Mean size (Kb)
http global	100	1	1	1	0	0	0	0.00	21
http warning	0	0	-1	0	-1	-1	-1	0.00	0
http severe	0	0	-1	0	-1	-1	-1	0.00	0

0 hits/min on 1 requests [Details](#)

Statistics sql - 1 day

None

Statistics jsp - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	% of system error
jsp global	100	109	463	10,360	1,265	100	148	0.00
jsp warning	5	1	2,992	2,992	0	0	0	0.00
jsp severe	39	3	6,615	10,360	3,482	0	0	0.00

16 hits/min on 52 requests [Details](#)

Statistics http system errors - 1 day

None

Statistics system errors logs - 1 day

None

Current requests

None

System information

[Execute the garbage collector](#)
[Generate a heap dump](#)
[View memory histogram](#)
[Invalidate http sessions](#)
[View http sessions](#)
[View deployment descriptor](#)
[MBeans](#)
[View OS processes](#)
[JNDI tree](#)

Host: **HP-PC@111.223.155.252**
 Java memory used: **81 Mb / 247 Mb**
 Nb of http sessions: **1**
 Nb of active threads (current http requests): **0**
 Nb of active jdbc connections: **0**
 Nb of used jdbc connections (opened if no datasource): **0** [Details](#)

OS: **Windows 7, x86_32 (4 cores)**
 Java: **Java(TM) SE Runtime Environment, 1.6.0_29-b11**
 JVM: **Java HotSpot(TM) Client VM, 20.4-b02, mixed mode**
 PID of process: **2128**
 Server: **Apache Tomcat/6.0.35**
 Webapp context:
 Start: **4/14/12 11:36 PM**
 JVM arguments:
 -Djava.util.logging.config.file=E:\apache-tomcat-6.0.35\apache-tomcat-6.0.35\conf\logging.properties
 -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
 -Djava.endorsed.dirs=E:\apache-tomcat-6.0.35\apache-tomcat-6.0.35\endorsed
 -Dcatalina.base=E:\apache-tomcat-6.0.35\apache-tomcat-6.0.35
 -Dcatalina.home=E:\apache-tomcat-6.0.35\apache-tomcat-6.0.35
 -Djava.io.tmpdir=E:\apache-tomcat-6.0.35\apache-tomcat-6.0.35\temp

Mean age of http sessions (min): **3**
 Tomcat http-8080: **Busy threads = 1 / 200**
 Bytes received = 5,174
 Bytes sent = 1,059,648
 Request count = 229
 Error count = 0
 Sum of processing times (ms) = 3,516
 Max processing time (ms) = 1,568

Memory: **Non heap memory = 59 Mb (Perm Gen, Code Cache),
 Loaded classes = 9,504,
 Garbage collection time = 2,593 ms,
 Process cpu time = 56,581 ms,
 Committed virtual memory = 249 Mb,
 Free physical memory = 1,740 Mb,
 Total physical memory = 4,043 Mb,
 Free swap space = 4,095 Mb,
 Total swap space = 4,095 Mb**

Perm Gen memory: **49 Mb / 64 Mb**
 Free disk space: **194,384 Mb**

Threads

Threads on HP-PC@111.223.155.252: Number = 68, Maximum = 71, Total started = 92 [Details](#)

Last collect time: 309 ms
 Display time: 51 ms
 Memory overhead estimate: < 1 Mb
 JavaMelody 1.36.0

Disclaimer : I was only able to try out Javamelody on a development environment. Therefore, implementers are advised to test out Javamelody on a trial environment before moving it into the production server/s.