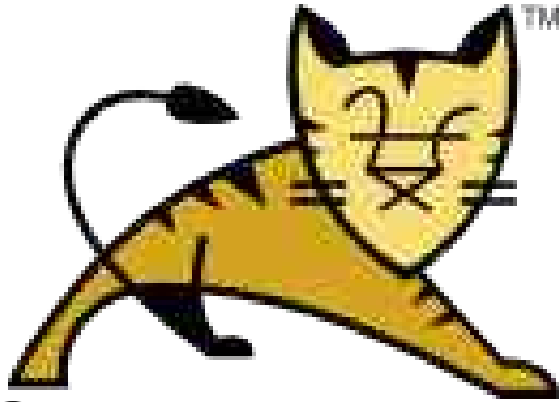


Monitoring Tomcat with JMX



Christopher Schultz

Chief Technology Officer

Total Child Health, Inc.

* Slides available on the Linux Foundation / ApacheCon2015 web site and at [http://people.apache.org/~schultz/ApacheCon NA 2015/Monitoring Apache Tomcat with JMX.odp](http://people.apache.org/~schultz/ApacheCon%20NA%202015/Monitoring%20Apache%20Tomcat%20with%20JMX.odp)

Java Management Extensions

- Protocol and API for managing and monitoring
 - Access data via JMX “Mbeans”
 - Read and write bean attributes
 - Invoke operations
 - Receive notifications
- JVM exposes certain status
- Tomcat exposes certain status

Monitoring JVM

- Heap status
- Total, free, used memory
- Garbage collection
- GC pause times

Monitoring Tomcat

- Status of connector
- Status of request-processor thread pool
- Status of data sources
- Request performance

JMX Tools

- jconsole (JDK)
- VisualVM (JDK, app bundle)
- Most profilers (e.g. YourKit, etc.)
- Custom tools using javax.management API

Monitoring JVM: Heap

The screenshot displays the JMX console interface. On the left, the MBeans tree is expanded to show the `Memory` MBean under the `java.lang` package. The right pane shows the `Attributes` tab for the selected MBean. The `Attribute values` section contains a table with the following data:

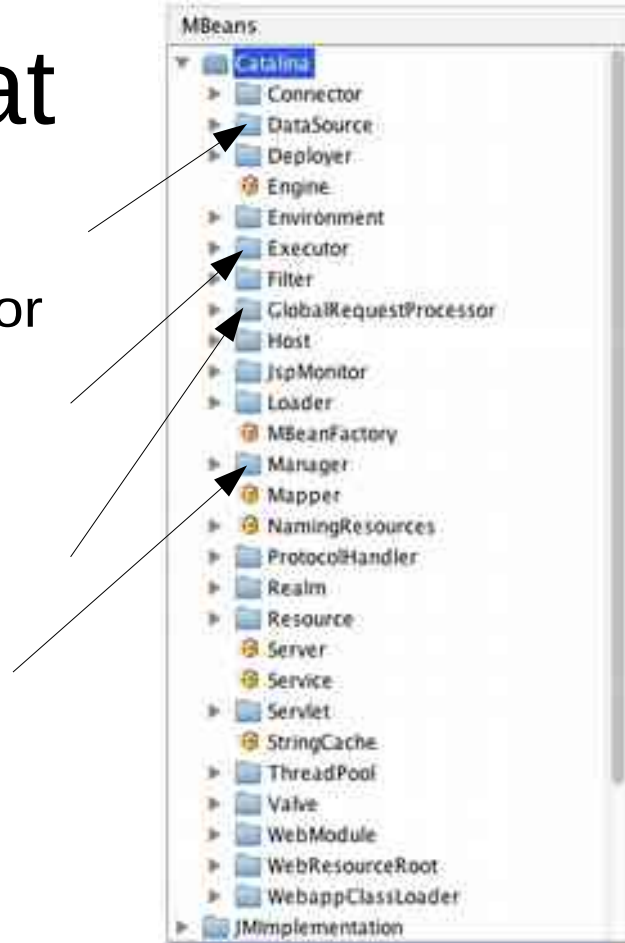
Name	Value
committed	161480704
init	66060288
max	179306496
used	115742312

Below the table, the following attributes are listed:

- `NonHeapMemoryUsage`: `javax.management.openmbean.CompositeDataSupport`
- `ObjectName`: `java.lang:type=Memory`
- `ObjectPendingFinalizationCo...`: `0`
- `Verbose`: `false`

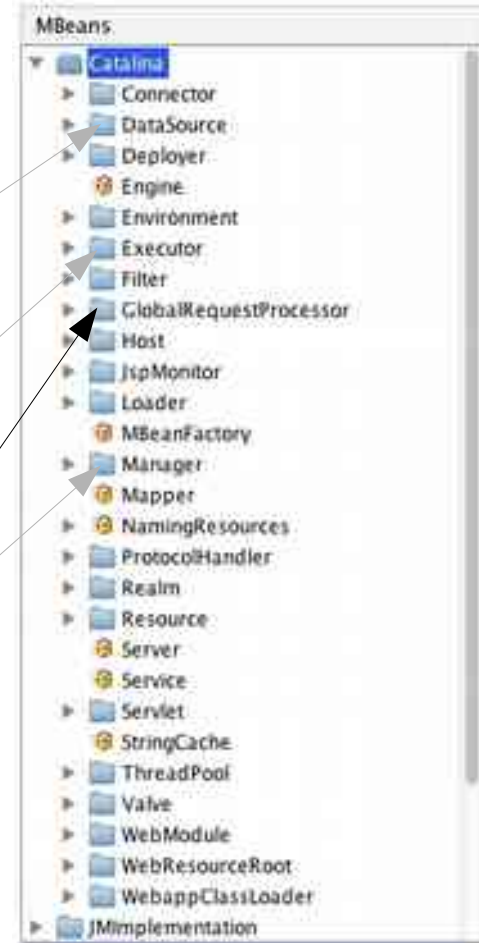
Monitoring Tomcat

- Status of data sources
- Status of request-processor thread pool
- Request performance
- Session information



Monitoring Tomcat

- Status of data sources
- Status of request-processor thread pool
- Request performance
- Session information



Monitoring Tomcat: Requests

The screenshot displays the JMX console interface for monitoring Tomcat. The left pane shows the MBeans tree with the connector `http-nio-127.0.0.1-8217` selected. The right pane shows the attribute values for this connector.

Name	Value
bytesReceived	0
bytesSent	5846954488
errorCount	0
maxTime	824
modelerType	org.apache.coyote.RequestGroupInfo
processingTime	1046463
requestCount	5192453

Monitoring Tomcat: Requests

The screenshot displays the JMX console interface. On the left, the MBeans tree is expanded to show the `GlobalRequestProcessor` sub-tree, with the `http-nio-127.0.0.1-8217` bean selected. On the right, the `Operations` tab is active, showing the `resetCounters` operation being invoked on the selected bean.

MBeans

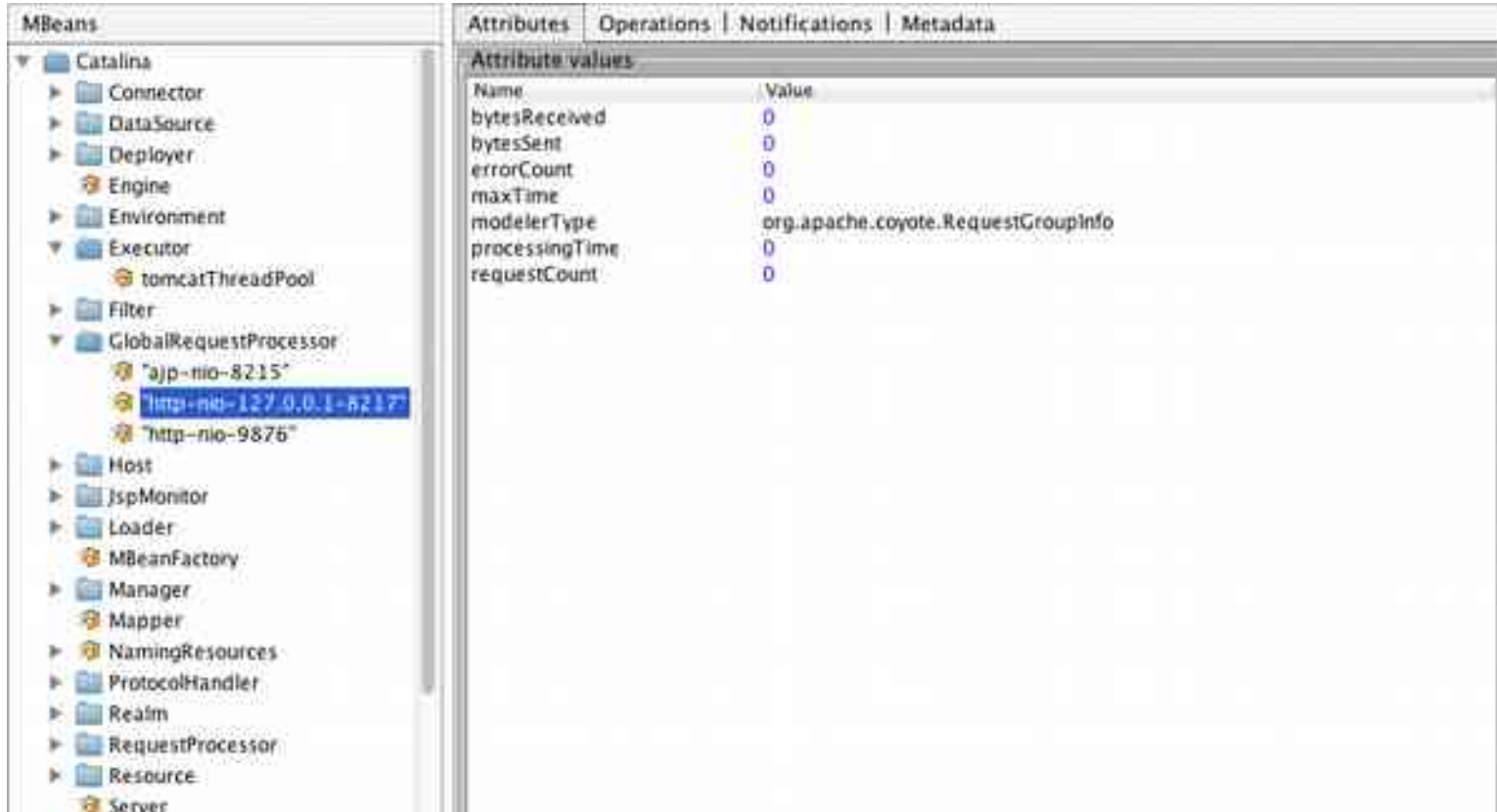
- ▼ Catalina
 - ▶ Connector
 - ▶ DataSource
 - ▶ Deployer
 - ▶ Engine
 - ▶ Environment
 - ▼ Executor
 - ▶ tomcatThreadPool
 - ▶ Filter
 - ▼ GlobalRequestProcessor
 - ▶ "ajp-nio-8215"
 - ▶ **"http-nio-127.0.0.1-8217"**
 - ▶ "http-nio-9876"
 - ▶ Host
 - ▶ JspMonitor
 - ▶ Loader
 - ▶ MBeanFactory
 - ▶ Manager
 - ▶ Mapper
 - ▶ NamingResources
 - ▶ ProtocolHandler
 - ▶ Realm
 - ▶ RequestProcessor
 - ▶ Resource
 - ▶ Server

Attributes | **Operations** | Notifications | Metadata

Operation invocation

```
void resetCounters ()
```

Monitoring Tomcat: Requests



The screenshot displays the JMX console interface for monitoring Tomcat. The left pane shows the MBeans tree with the following structure:

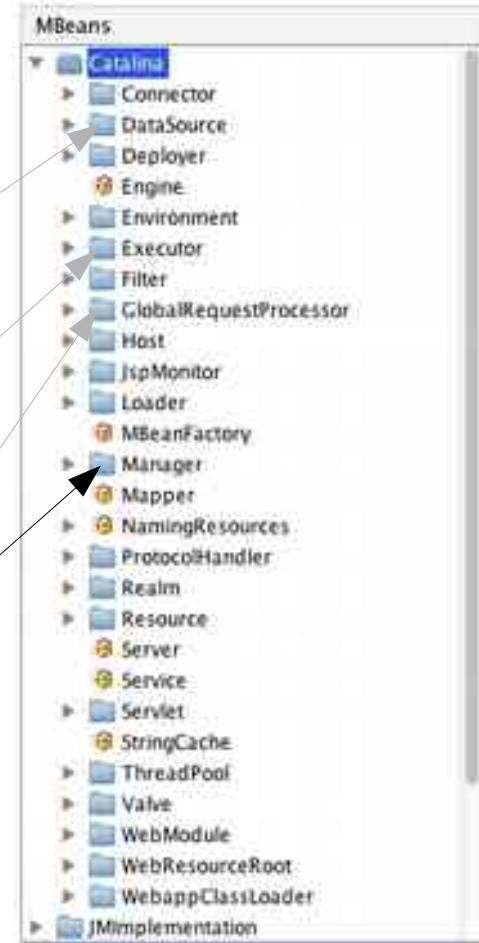
- Catalina
 - Connector
 - DataSource
 - Deployer
 - Engine
 - Environment
 - Executor
 - tomcatThreadPool
 - Filter
 - GlobalRequestProcessor
 - ajp-nio-8215
 - http-nio-127.0.0.1-8217** (selected)
 - http-nio-9876
 - Host
 - JspMonitor
 - Loader
 - MBeanFactory
 - Manager
 - Mapper
 - NamingResources
 - ProtocolHandler
 - Realm
 - RequestProcessor
 - Resource
 - Server

The right pane shows the selected MBean's attribute values:

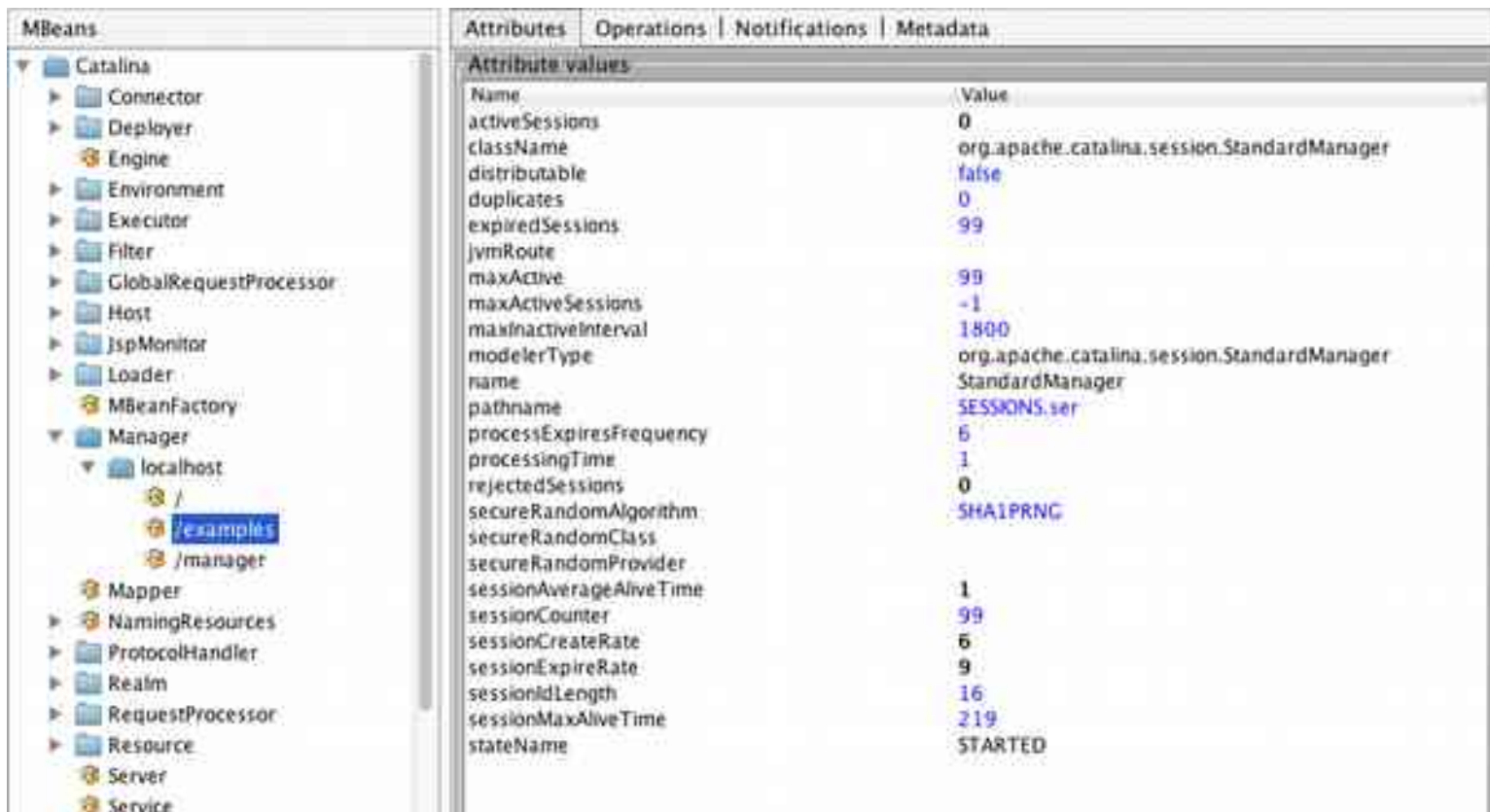
Name	Value
bytesReceived	0
bytesSent	0
errorCount	0
maxTime	0
modelerType	org.apache.coyote.RequestGroupInfo
processingTime	0
requestCount	0

Monitoring Tomcat

- Status of data sources
- Status of request-processor thread pool
- Request performance
- **Session information**



Monitoring Tomcat: Sessions



The screenshot displays the JMX console interface. On the left, the MBean tree shows the hierarchy: Catalina > Manager > localhost > /examples. The 'examples' MBean is selected. On the right, the 'Attribute values' table provides the following data:

Name	Value
activeSessions	0
className	org.apache.catalina.session.StandardManager
distributable	false
duplicates	0
expiredSessions	99
jvmRoute	
maxActive	99
maxActiveSessions	-1
maxInactiveInterval	1800
modelerType	org.apache.catalina.session.StandardManager
name	StandardManager
pathname	SESSIONS.ser
processExpiresFrequency	6
processingTime	1
rejectedSessions	0
secureRandomAlgorithm	SHA1PRNG
secureRandomClass	
secureRandomProvider	
sessionAverageAliveTime	1
sessionCounter	99
sessionCreateRate	6
sessionExpireRate	9
sessionIdLength	16
sessionMaxAliveTime	219
stateName	STARTED

Monitoring Tomcat

- Status of data sources
- Status of request-processor thread pool
- Request performance
- Session information



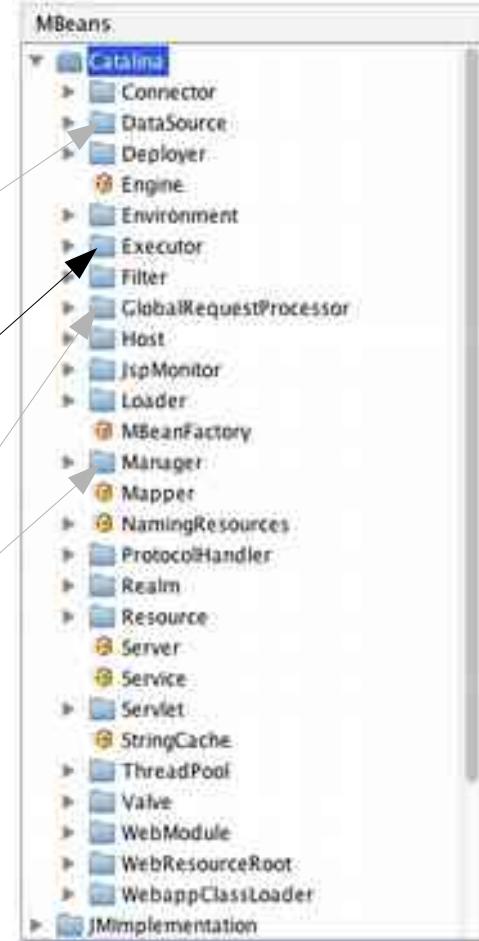
Monitoring Tomcat: DataSources

The screenshot displays the JMX console interface. On the left, the 'MBeans' tree is expanded to show the path: Catalina > DataSource > localhost > /examples > javax.sql.DataSource > jdbc/MyDataSource. The selected MBean is highlighted in blue. On the right, the 'Attributes' tab is active, showing a table of attribute values for the selected DataSource.

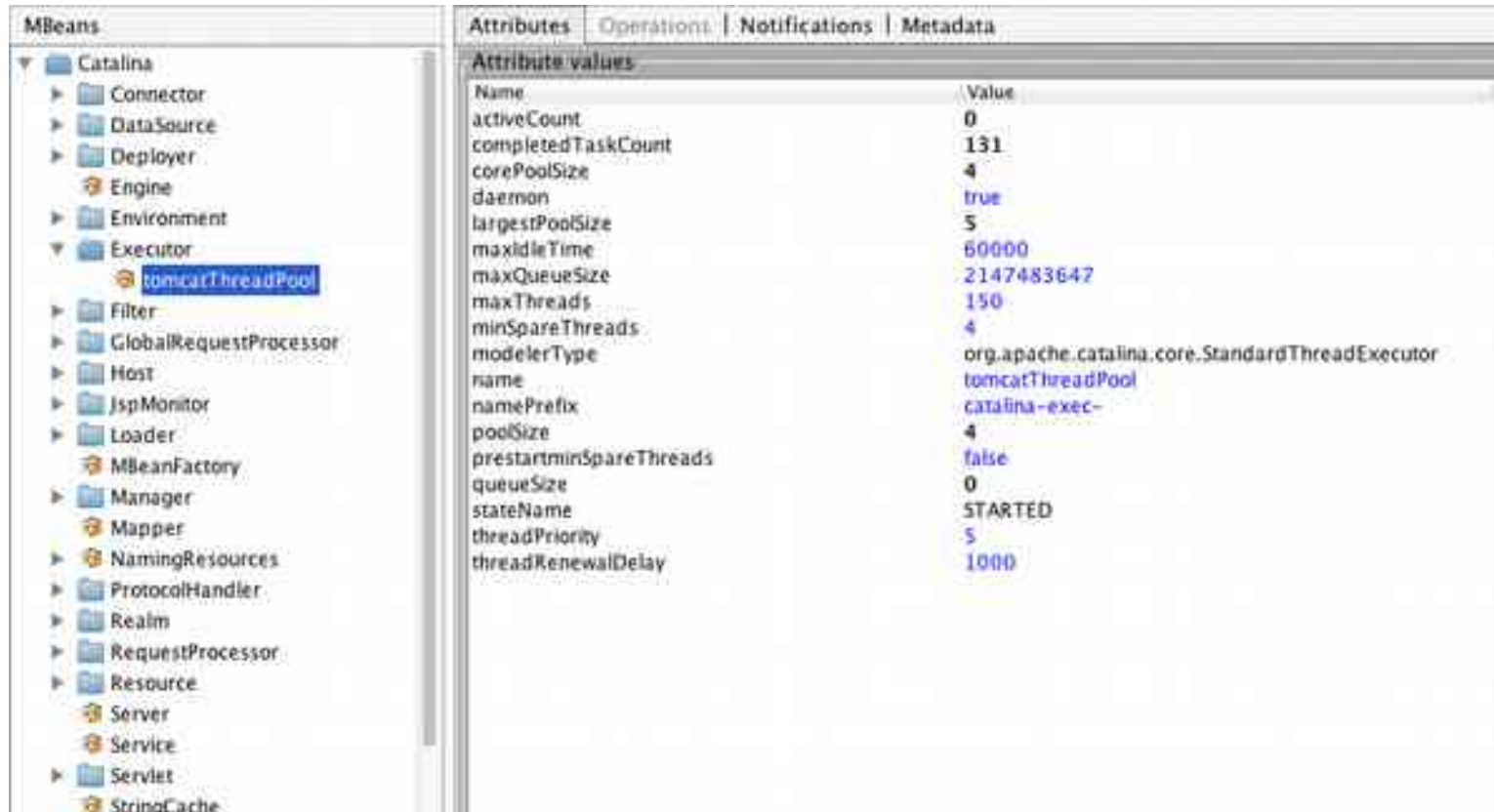
Name	Value
defaultTransactionIsolation	-1
driverClassName	com.mysql.jdbc.Driver
enableAutoCommitOnReturn	true
evictionPolicyClassName	org.apache.tomcat.dbcp.pool2.impl.DefaultEvicti...
initialSize	1
jmxName	Catalina?type=DataSource,host=localhost,context...
lifo	true
logAbandoned	true
loginTimeout	Unavailable
maxConnLifetimeMillis	-1
maxIdle	1
maxOpenPreparedStatements	-1
maxTotal	1
maxWaitMillis	10000
minEvictableIdleTimeMillis	1800000
minIdle	0
modelerType	org.apache.tomcat.dbcp.dbcp2.BasicDataSource
numActive	0
numIdle	1
numTestsPerEvictionRun	3
password	
poolPreparedStatements	false
removeAbandonedOnBorrow	false
removeAbandonedOnMaintenance	false
removeAbandonedTimeout	30
rollbackOnReturn	true
softMinEvictableIdleTimeMillis	-1

Monitoring Tomcat

- Status of data sources
- Status of request-processor thread pool
- Request performance
- Session information



Monitoring Tomcat: Threads



The screenshot displays the JMX console interface. On the left, the 'MBeans' tree shows the hierarchy: Catalina > Executor > tomcatThreadPool. The right pane shows the 'Attributes' tab for the selected MBean, listing various attribute values.

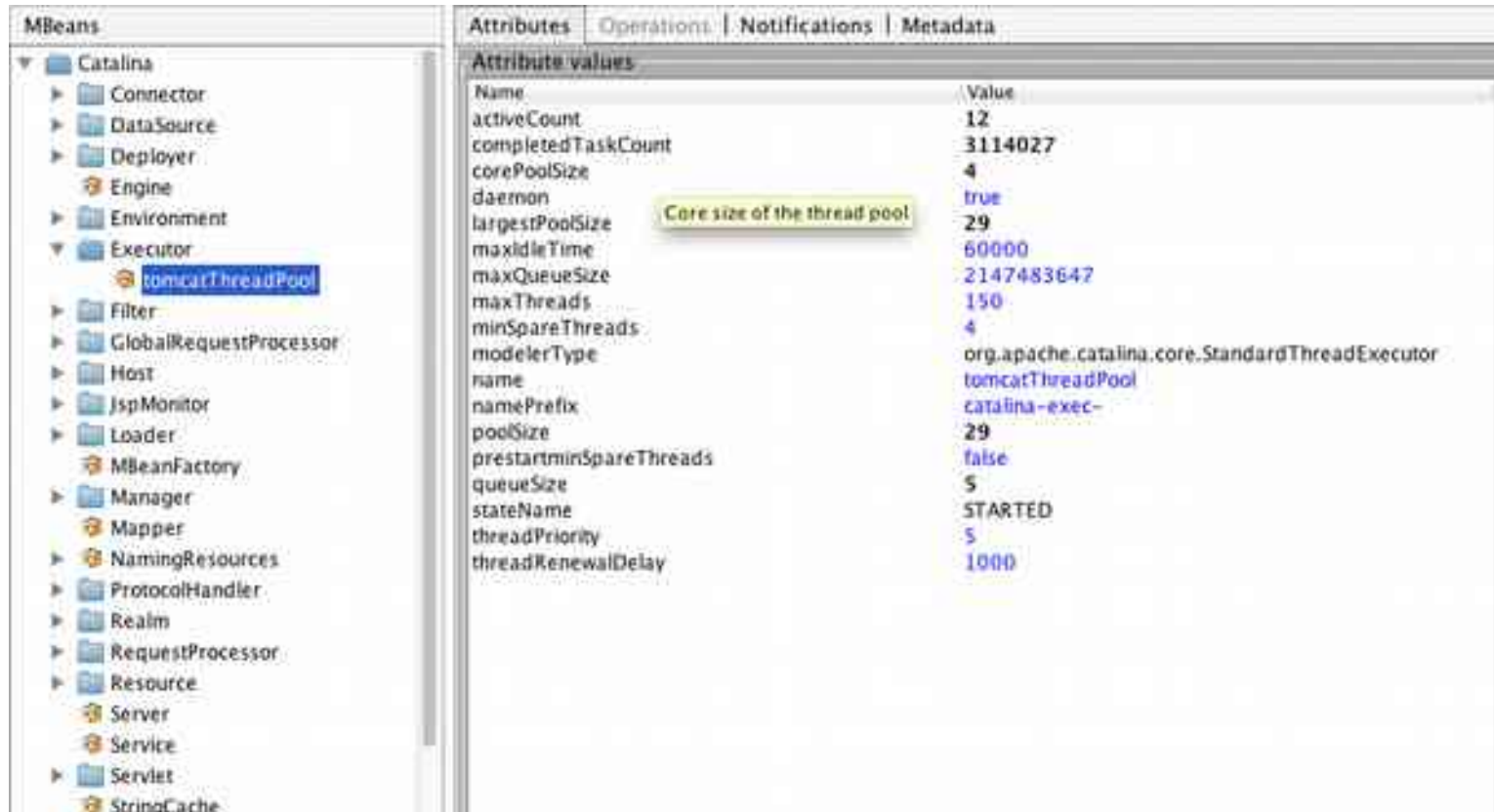
Name	Value
activeCount	0
completedTaskCount	131
corePoolSize	4
daemon	true
largestPoolSize	5
maxIdleTime	60000
maxQueueSize	2147483647
maxThreads	150
minSpareThreads	4
modelerType	org.apache.catalina.core.StandardThreadExecutor
name	tomcatThreadPool
namePrefix	catalina-exec-
poolSize	4
prestartminSpareThreads	false
queueSize	0
stateName	STARTED
threadPriority	5
threadRenewalDelay	1000

Monitoring Tomcat: Threads

The screenshot displays the JBoss JMX console interface. On the left, a tree view under 'Catalina' shows the 'tomcatThreadPool' MBean selected. The right pane shows the 'Attributes' tab with a table of attribute values.

Name	Value
activeCount	6
completedTaskCount	725534
corePoolSize	4
daemon	true
largestPoolSize	21
maxIdleTime	60000
maxQueueSize	2147483647
maxThreads	150
minSpareThreads	4
modelerType	org.apache.catalina.core.StandardThreadExecutor
name	tomcatThreadPool
namePrefix	catalina-exec-
poolSize	21
prestartminSpareThreads	false
queueSize	0
stateName	STARTED
threadPriority	5
threadRenewalDelay	1000

Monitoring Tomcat: Threads



The screenshot displays the JBoss console interface. On the left, the 'MBeans' tree shows the hierarchy: Catalina > Executor > tomcatThreadPool. The right pane shows the 'Attributes' tab for the selected MBean, listing various configuration parameters and their current values.

Name	Value
activeCount	12
completedTaskCount	3114027
corePoolSize	4
daemon	true
largestPoolSize	29
maxIdleTime	60000
maxQueueSize	2147483647
maxThreads	150
minSpareThreads	4
modelerType	org.apache.catalina.core.StandardThreadExecutor
name	tomcatThreadPool
namePrefix	catalina-exec-
poolSize	29
prestartminSpareThreads	false
queueSize	5
stateName	STARTED
threadPriority	5
threadRenewalDelay	1000

A yellow callout box highlights the 'largestPoolSize' attribute with the text 'Core size of the thread pool'.


Monitoring Tomcat: Threads

MBeans

- Catalina
 - Connector
 - DataSource
 - Deployer
 - Engine
 - Environment
 - Executor
 - tomcatThreadPool**
 - Filter
 - GlobalRequestProcessor
 - Host
 - JspMonitor
 - Loader
 - MBeanFactory
 - Manager
 - Mapper
 - NamingResources
 - ProtocolHandler
 - Realm
 - RequestProcessor
 - Resource
 - Server
 - Service
 - Servlet
 - StringCache

Attributes | Operations | Notifications | Metadata

Attribute values

Name	Value
activeCount	
completedTaskCount	3114027
corePoolSize	4
daemon	true
largestPoolSize	29
maxIdleTime	60000
maxQueueSize	2147483647
maxThreads	150
minSpareThreads	4
modelerType	org.apache.catalina.core.StandardThreadExecutor
name	tomcatThreadPool
namePrefix	catalina-exec-
poolSize	29
prestartminSpareThreads	false

Monitoring Your Application

- Monitor Application Processes
- Performance Metrics
- On-the-fly re-configuration

Monitoring Your Application

- Write an MBean
 - Create an Interface: FooMBean
 - Create an Implementation: Foo
 - Create an XML MBean descriptor
- Deploy package to Tomcat
 - Publish the MBean to the MBean server
- Query / invoke as necessary

* Example code will be available at

[http://people.apache.org/~schultz/ApacheCon NA 2015/Tomcat Monitoring/](http://people.apache.org/~schultz/ApacheCon%20NA%202015/Tomcat%20Monitoring/)

Example MBean

- Servlet Filter that captures total request processing time
 - Timestamp prior to request
 - Timestamp after request
 - Add the delta to a JMX-accessible counter:
RequestStats

RequestStats MBean

- Write an MBean

```
public interface RequestStatsMBean {
    public long getProcessingTime();
    public long getRequestCount();
    public void resetCounters();
}

public class RequestStats
    implements RequestStatsMBean {
    [...]
    public void updateStats(long
timestamp, ServletRequest request, long
elapsed) {

    _totalElapsedTime.addAndGet(elapsed);

    _requestCount.incrementAndGet();
    }
}
```

```
    public long getProcessingTime(){
        return _totalElapsedTime.get();
    }
    public long getRequestCount() {
        return _requestCount.get();
    }
    public void resetCounters() {
        _totalElapsedTime.set(0l);
        _requestCount.set(0l);
    }
}
```

RequestStats MBean

- Write an MBean descriptor

```
<mbeans-descriptors>
  <mbean name="RequestStats" ...>
    <operation name="getProcessingTime"
      description="Gets the total number of
milliseconds spent processing requests."
      impact="INFO"
      returnType="long" />
    <operation name="getRequestCount"
      description="Gets the total number
of requests processed."
      impact="INFO"
      returnType="long" />
    <operation
      name="resetCounters"
      description="Resets all
counters."
      impact="ACTION"
      returnType="void" />
  </mbean>
</mbeans-descriptors>
```

RequestStats MBean

- Create JAR
 - Java interface
 - Java implementation
 - mbeans-descriptors.xml
- Put JAR into \$CATALINA_BASE/lib

RequestStats MBean

- Write the Filter

```
public void init(FilterConfig config) {
    MBeanServer server = getServer();
    server.registerMBean(_stats, new
ObjectName("Example:RequestStats=RequestStats,name=" + filterName;));
}
public void doFilter(...) {
    timestamp = elapsed = System.currentTimeMillis();
    chain.doFilter(request, response);
    elapsed = System.currentTimeMillis() - elapsed;

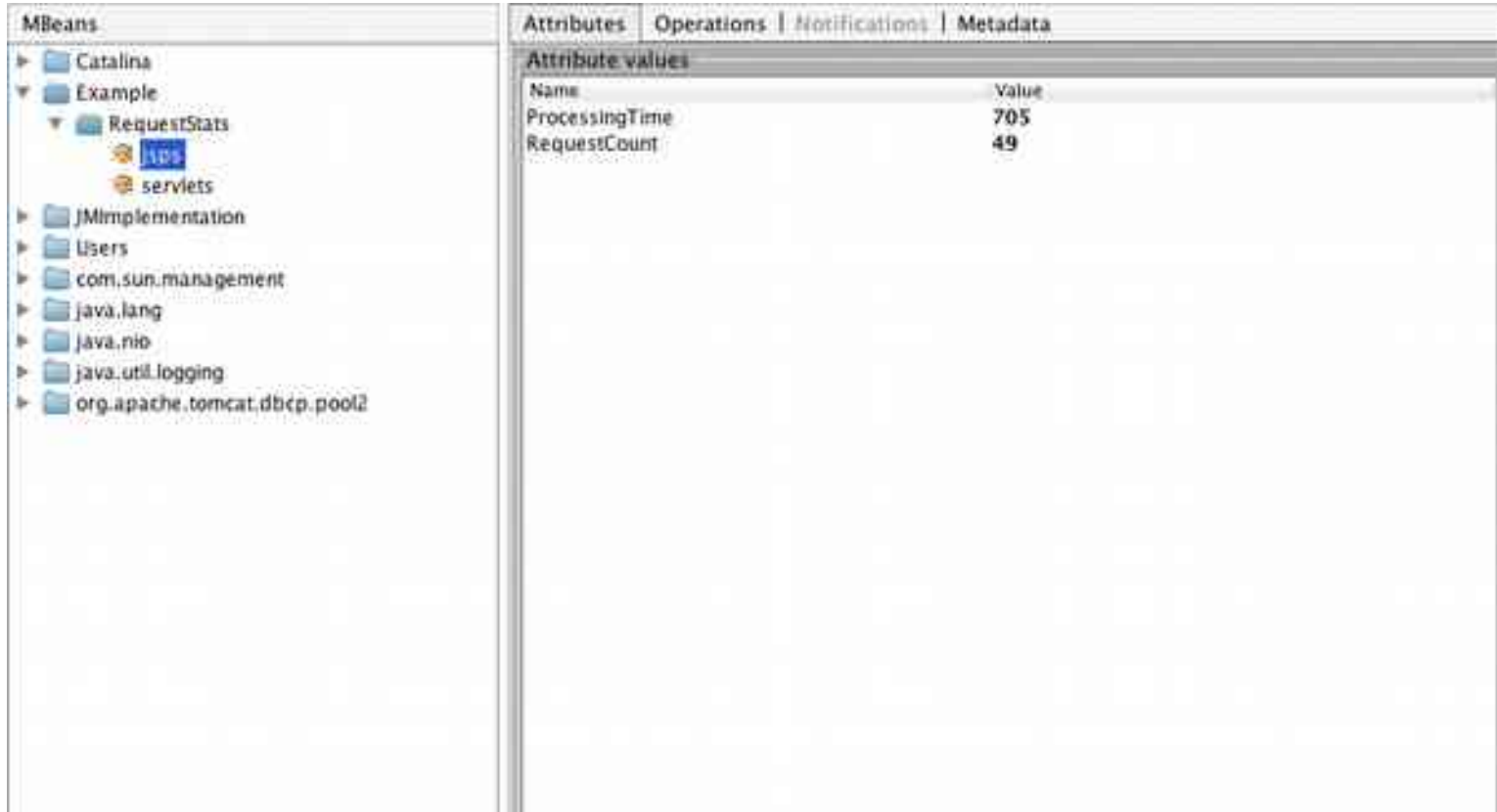
    _stats.updateStats(timestamp, request, elapsed);
}
```

RequestStats MBean

- Map the Filter

```
<filter>
  <filter-name>servlet-request-stats</filter-name>
  <filter-class>filters.RequestStatsFilter</filter-class>
  <init-param>
    <param-name>name</param-name>
    <param-value>servlets</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>servlet-request-stats</filter-name>
  <url-pattern>/servlets/*</url-pattern>
</filter-mapping>
<filter><filter-name>jsp-request-stats</filter-name><filter-
class>filters.RequestStatsFilter</filter-class><init-param><param-name>name</param-
name><param-value>jsp</param-value></init-param></filter>
  <filter-mapping><filter-name>jsp-request-stats</filter-name><url-pattern>/jsp/*</url-
pattern></filter-mapping>
```

RequestStats MBean



The screenshot displays the JMX console interface. On the left, the 'MBeans' tree shows a hierarchy: Catalina > Example > RequestStats. The 'RequestStats' MBean is selected. On the right, the 'Attributes' tab is active, showing a table of attribute values:

Name	Value
ProcessingTime	705
RequestCount	49

RequestStats MBean

The screenshot displays the JBoss JMX console interface. On the left, the 'MBeans' tree shows a hierarchy: Catalina > Example > RequestStats. Under 'RequestStats', there are sub-nodes for 'MBeans', 'servlets', and 'servlets'. The right pane shows the 'Operations' tab for the selected MBean. The 'Operation invocation' section displays a single operation: 'void resetCounters ()'. The 'resetCounters' method name is highlighted with a blue selection box.

Attributes	Operations	Notifications	Metadata
Operation invocation			
void resetCounters ()			

Automated Monitoring

- Remote Access
- Large Scale
- Constant

Automated Monitoring

- Remote *Access*
- *Large Scale*
- *Constant*

Need more tools!

Automated Monitoring

- Nagios
 - Simple
 - Flexible
 - Well-deployed
 - No-cost community version available

Automated Monitoring

Host	Service	Status	Last Check	Duration	Attempt	Status Information
abi.apache.org	 SSH	OK	2014-03-18 15:12:04	1d 16h 22m 57s	1/10	SSH OK - OpenSSH_5.8p2_hp13v11 FreeBSD-20110503 (protocol 2.0)
aegis.apache.org	 HTTP - Buildbot	OK	2014-03-18 15:13:43	9d 12h 56m 18s	1/10	HTTP OK HTTP/1.1 200 OK - 22230 bytes in 0.642 seconds
	HTTPS - Jenkins	OK	2014-03-18 15:14:43	0d 23h 25m 18s	1/10	HTTP OK HTTP/1.1 200 OK - 22230 bytes in 0.830 seconds
	SSH	OK	2014-03-18 15:14:14	9d 0m 25m 47s	1/10	SSH OK - OpenSSH_5.9p1 Debian-Subuntu1.1 (protocol 2.0)
analysis-vn.apache.org	 SSH	OK	2014-03-18 15:12:43	4d 12h 32m 18s	1/10	SSH OK - OpenSSH_5.9p1 Debian-Subuntu1.1 (protocol 2.0)
any.no-ip.com	DNS	OK	2014-03-18 15:11:42	24d 19h 28m 22s	1/10	DNS OK - 0.023 seconds response time: www.apache.org returns 140.211.11.131,192.87.106.229
arc4s.apache.org	 SSH	OK	2014-03-18 15:14:48	19d 7h 40m 14s	1/10	SSH OK - OpenSSH_5.9p1 Debian-20120710asf3 (protocol 2.0)
athena.apache.org	 DNS	OK	2014-03-18 15:14:23	9d 0m 25m 38s	1/10	DNS OK - 0.172 seconds response time: svn.geo.apache.org returns 160.45.251.2
	GEODNS	OK	2014-03-18 15:14:14	12d 17h 10m 47s	1/10	OK DNS server 140.211.11.138 geo.apache.org is in sync with the zone file held in SVN (SERIAL in SVN: [2013101200] // SERIAL on 140.211.11.138 geo.apache.org [2013101200])
	SMTP	OK	2014-03-18 15:10:14	0d 13h 14m 47s	1/10	SMTP OK - 0.651 sec. response time
	SSH	OK	2014-03-18 15:14:23	20d 15h 50m 38s	1/10	SSH OK - OpenSSH_5.8p2_hp13v11 FreeBSD-20110503 (protocol 2.0)
aurora.apache.org	 HTTP - WWW EU	OK	2014-03-18 15:13:19	0d 23h 41m 42s	1/10	HTTP OK HTTP/1.1 200 OK - 40315 bytes in 0.418 seconds

Nagios Monitoring

- Plug-in architecture (i.e. arbitrary scripts)
- Freely-available JMX plug-in: `check_jmx`

```
$ ./check_jmx -U
service:jmx:rmi:///jndi/rmi://localhost:1100/jmxrmi\
  -O java.lang:type=Memory -A NonHeapMemoryUsage -K used\
  -w 290000000 -c 300000000
JMX WARNING NonHeapMemoryUsage.used=29050880
```

Nagios Monitoring

- Problems with check_jmx
 - Complex configuration for remote JMX
 - JVM launch for every check
 - Course-grained authentication options






Nagios Monitoring

- Alternative Option: Tomcat's JMXProxyServlet
 - JMX data available via HTTP
 - Can use Tomcat's authentication tools

```
$ ./check_jmxproxy -U 'http://localhost/manager/jmxproxy?
get=java.lang:type=Memory&att=HeapMemoryUsage&key=used' \
-w 290000000 -c 300000000
JMX CRITICAL: OK - Attribute get 'java.lang:type=Memory' -
HeapMemoryUsage - key 'used' = 100875248
```

* check_jmxproxy can be found at
http://wiki.apache.org/tomcat/tools/check_jmxproxy.pl

Nagios Monitoring

 JVM:Heap	OK	03-18-2014 15:17:04	8d 9h 56m 14s	1/4	JMX OK: OK - Attribute get 'java.lang.type=Memory' - HeapMemoryUsage - key 'used' = 126743888
 JVM:Sessions	OK	03-18-2014 15:15:05	8d 9h 53m 13s	1/4	JMX OK: OK - Attribute get 'Catalina:type=Manager,context=/...host=localhost' - activeSessions = 0
 JVM:Heap	OK	03-18-2014 15:16:08	0d 0h 42m 10s	1/4	JMX OK: OK - Attribute get 'java.lang.type=Memory' - HeapMemoryUsage - key 'used' = 253538440
 JVM:Sessions	OK	03-18-2014 15:15:08	8d 10h 13m 10s	1/4	JMX OK: OK - Attribute get 'Catalina:type=Manager,context=/...host=localhost' - activeSessions = 180
 JVM:Heap-OOME ?	OK	03-06-2014 15:58:13	11d 23h 20m 5s	1/1	OK

JMX Command-line Tricks

- Show all logged-in usernames

```
for sessionid in `wget -O - 'http://user:pwd@host/manager/jmxproxy?
invoke=Catalina:type=Manager,context=/myapp,host=localhost&op=listS
essionIds' \
    | sed -e "s/ /\\n/g"
    | grep '^([0-9A-Za-z]\\+\\(\\.\\.\\.*)\\)?$' ;\
do wget -O - "http://user:pwd@host/manager/jmxproxy?
invoke=Catalina:type=Manager,context=/myapp,host=localhost&op=getSe
ssionAttribute&ps=$sessionid,user" ; done 2>/dev/null \
    | grep User
```


Tracking Values Over Time

- Some metrics are best observed as deltas
 - Session count
 - Request error count
- Requires that you have a history of data
- Requires that you consult the history of that data
- `check_jmxproxy` provides such capabilities

Tracking Values Over Time

```
$ ./check_jmxproxy -U 'http://localhost/manager/jmxproxy?  
get=java.lang:type=Memory&att=HeapMemoryUsage&key=used' -w 33554432 -c 50331648 --write  
number.out --compare number.out
```

```
JMX OK: OK - Attribute get 'java.lang:type=Memory' - HeapMemoryUsage - key 'used' = 102278904,  
delta=[...]
```

```
$ ./check_jmxproxy -U 'http://localhost/manager/jmxproxy?  
get=java.lang:type=Memory&att=HeapMemoryUsage&key=used' -w 33554432 -c 50331648 --write  
number.out --compare number.out
```

```
JMX OK: OK - Attribute get 'java.lang:type=Memory' - HeapMemoryUsage - key 'used' = 113806144,  
delta=11527240
```

```
$ ./check_jmxproxy -U 'http://localhost/manager/jmxproxy?  
get=java.lang:type=Memory&att=HeapMemoryUsage&key=used' -w 33554432 -c 50331648 --write  
number.out --compare number.out
```

```
JMX OK: OK - Attribute get 'java.lang:type=Memory' - HeapMemoryUsage - key 'used' = 109264056,  
delta=-4542088
```

Tracking Values Over Time

- Session count
 - Tomcat actually provides this already via Manager's `sessionCreateRate` attribute
- Request errors

```
$ ./check_jmxproxy -U 'http://localhost/manager/jmxproxy?
get=Catalina:type=RequestProcessor,worker="http-nio-127.0.0.1-
8217",name=HttpRequest1&att=errorCount' -w 1 -c 10 --write
errors.txt --compare errors.txt
```

```
JMX OK: OK - Attribute get
```

```
'Catalina:type=RequestProcessor,worker="http-nio-127.0.0.1-
8217",name=HttpRequest1' - errorCount = 0, delta=0
```

Detecting OutOfMemory

- Many sources of OOME
 - Heap exhaustion
 - PermGen exhaustion
 - Hit thread limit
 - Hit file descriptor limit

Detecting OutOfMemory

- Two types of heap OOME
 - One thread generates lots of local references
 - All threads collaborate to generate globally-reachable objects (e.g. session data)
- Former is recoverable, latter is not
- You want to be notified in any case

Memory Pool Thresholds

The screenshot displays the MBeans console with the following structure:

- MBeans
 - Catalina
 - Example
 - JMImplementation
 - Users
 - com.sun.management
 - java.lang
 - ClassLoading
 - Compilation
 - GarbageCollector
 - Memory
 - MemoryManager
 - MemoryPool
 - Code Cache
 - PS Eden Space
 - PS Old Gen
 - PS Perm Gen**
 - PS Survivor Space
 - OperatingSystem
 - Runtime
 - Threading
 - java.nio
 - java.util.logging
 - org.apache.tomcat.dbcp.pool2

The right-hand pane shows the configuration for the selected MBean:

Attributes	Operations	Notifications	Metadata
Attribute values			
Name			Value
CollectionUsage			javax.management.openmbean.CompositeData...
CollectionUsageThreshold			0
CollectionUsageThresholdCount			0
CollectionUsageThresholdExceeded			false
CollectionUsageThresholdSupported			true
MemoryManagerNames			java.lang.String[1]
Name			PS Perm Gen
ObjectName			java.lang.type=MemoryPool,name=PS Perm Gen
PeakUsage			javax.management.openmbean.CompositeData...
Type			NON_HEAP
Usage			javax.management.openmbean.CompositeData...
UsageThreshold			0
UsageThresholdCount			0
UsageThresholdExceeded			false
UsageThresholdSupported			true
Valid			true

Memory Pool Thresholds

The screenshot shows the JMX console interface. On the left is a tree view of MBeans, with 'java.lang.MemoryPool.PS Old Gen' selected. The right pane shows the configuration for this MBean, including attribute values and usage statistics.

MBeans

- Catalina
- Example
- JMImplementation
- Users
- com.sun.management
- java.lang
 - ClassLoader
 - Compilation
 - GarbageCollector
 - Memory
 - MemoryManager
 - MemoryPool
 - Code Cache
 - PS Eden Space
 - PS Old Gen**
 - PS Perm Gen
 - PS Survivor Space
 - OperatingSystem
 - Runtime
 - Threading
- java.nio
- java.util.logging
- org.apache.tomcat.dbcp.pool2

Attributes | Operations | Notifications | Metadata

Attribute values

Name	Value
CollectionUsageThresholdSupported	true
MemoryManagerNames	java.lang.String[1]
Name	PS Old Gen
ObjectName	java.lang:type=MemoryPool,name=PS Old Gen
PeakUsage	javax.management.openmbean.CompositeDat... HEAP
Type	HEAP

Usage

Name	Value
commit...	119537664
init	44040192
max	134217728
used	112171368

UsageThreshold: 120000000

UsageThresholdCount: 0

UsageThresholdExceeded: false

UsageThresholdSupported: true

Valid: true

Refresh

Memory Pool Thresholds

The screenshot displays the JMX console interface. On the left, the 'MBeans' tree is expanded to show the 'java.lang' package, specifically the 'MemoryPool' sub-package, where 'PS Old Gen' is selected. The main panel shows the 'Attributes' tab for this MBean. It lists various attributes such as 'CollectionUsageThresholdSupported', 'MemoryManagerNames', 'Name', 'ObjectName', 'PeakUsage', and 'Type'. Below this, there are navigation buttons for 'Tabular Navigation' and 'Composite Navigation'. A 'Usage' table is also present, showing memory usage statistics. At the bottom, the 'UsageThreshold' attribute is highlighted in blue, indicating its value is 120000000. A 'Refresh' button is located at the very bottom of the console.

Name	Value
CollectionUsageThresholdSupported	true
MemoryManagerNames	java.lang.String[1]
Name	PS Old Gen
ObjectName	java.lang:type=MemoryPool,name=PS Old Gen
PeakUsage	javax.management.openmbean.CompositeData...
Type	HEAP

Name	Value
commit...	128974848
init	44040192
max	134217728
used	114510568

UsageThreshold	120000000
UsageThresholdCount	2
UsageThresholdExceeded	false
UsageThresholdSupported	true
Valid	true

Memory Pool Thresholds

The screenshot displays a Java monitoring interface. On the left, a tree view of MBeans is shown, with the 'Memory' sub-tree expanded. On the right, a 'Notification buffer' table shows a single notification event.

MBeans:

- Catalina
- Example
- JMImplementation
- Users
- com.sun.management
- java.lang
 - ClassLoading
 - Compilation
 - GarbageCollector
 - Memory**
 - MemoryManager
 - MemoryPool
 - Code Cache
 - PS Eden Space
 - PS Old Gen
 - PS Perm Gen
 - PS Survivor Space
 - OperatingSystem
 - Runtime
 - Threading
- java.nio
- java.util.logging
- org.apache.tomcat.dbcp.pool2

Notification buffer:

TimeStamp	Type	UserData	SeqNum	Message	Event	Source
15:59:04:...	java.managem...	javax.manage...	2	Memory ...	javax.mana...	java.lang.ty...

Buttons: [Subscribe](#) [Unsubscribe](#) [Clear](#)

Memory Pool Thresholds

- Choice of how to detect exceeded-threshold conditions
 - Polling using `check_jmxproxy`
 - Register a notification listener from Java
 - Have that listener take some action

Detect OutOfMemory

- Monitoring Memory Thresholds
 - Set threshold on startup
 - Register a notification listener (callback)
 - Watch “exceeded” count (poll)
 - Report to monitoring software (Nagios)
 - Repeat for each memory pool you want to watch
 - Hope the JVM does not fail during notification
 - This is getting ridiculous

Detecting OutOfMemory

- JVM has an easier way
- Use `-XX:OnOutOfMemoryError` to run a command on *first* OOME detected by the JVM
- Need a command to notify Nagios

Notify Nagios on OOME

- Script that wraps curl

```
$ curl -si \  
  --data-urlencode 'cmd_typ=30' \  
  --data-urlencode 'cmd_mod=2' \  
  --data-urlencode "host=myhost" \  
  --data-urlencode "service=JVM:Heap:OOME" \  
  --data-urlencode "plugin_state=2" \  
  --data-urlencode "plugin_output=OOME CRITICAL" \  
  'https://monitoring-host/nagios/cgi-bin/cmd.cgi'
```

Script can be found at <http://wiki.apache.org/tomcat/tools/nagios-send-passive-check.sh>

Monitoring Tomcat with JMX

- JMX Provides Monitoring and Management of JVMs
- Tomcat exposes a great amount of information via JMX
- Applications can expose anything to JMX via MBeans
- JRE ships with tools for light JMX interaction
- Practical use of JMX requires some additional tools

Resources

- Presentation Slides

[http://people.apache.org/~schultz/ApacheCon NA 2015/Monitoring Apache Tomcat with JMX.odp](http://people.apache.org/~schultz/ApacheCon%20NA%202015/Monitoring%20Apache%20Tomcat%20with%20JMX.odp)

- Nagios passive-check script

<http://wiki.apache.org/tomcat/tools/nagios-send-passive-check.sh>

- `check_jmxproxy`

http://wiki.apache.org/tomcat/tools/check_jmxproxy.pl

- Special thanks to Christopher Blunck (MBeans info)

<http://oss.wxnet.org/mbeans.html>