Yammer metrics tutorial with JMX to gather metrics

Posted on October 29, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

Note: Yammer metrics has now moved to **metrics-core**.

from com.yammer.metrics:metrics-core:2.2.0 (NOT supported anymore) **to** com.codahale.metrics:metrics-core:3.0.*

from com.codahale.metrics:metrics-core:3.0.* **to** io.dropwizard.metrics:metrics-core:3.1.* since 2014.

When you are running long term applications like web applications, batch jobs, or stand-alone status update jobs, it is good to know some statistics about them, like number of requests served or request duration. You can also gather more generic information like the state of your internal

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all

- Ice Breaker Interview
- Core Java Interview C
  - Java Overview (4)
  - Data types (6)
  - constructors-metho
  - Reserved Key Wor
  - Classes (3)
  - Objects (8)
  - OOP (10)
  - GC (2)
  - Generics (5)
  - FP (8)
  - IO (7)
  - Multithreading (12)
  - Algorithms (5)
  - Annotations (2)
  - Collection and Data
  - Differences Betwee
  - Event Driven Progr
  - Exceptions (2)
  - Java 7 (2)

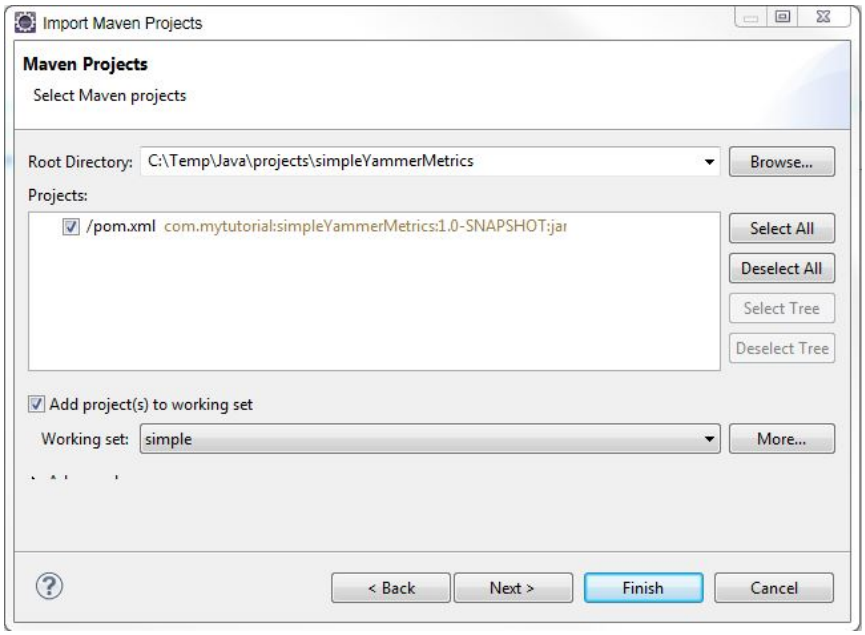
collections, how many times some portion of code is being executed, or health checks like database availability, or any kind of connection to an external system.

All this kind of instrumentation can be achieved by using native JMX or using a modular project like yammer Metrics. Metrics provides a powerful way to measure the behaviour of your critical components and reporting them to a variety of systems like, JConsole, System Console, Ganglia, Graphite, CSV, or making them available through a web services as JSON data.

Step 1: Create a new Maven based Java project.

```
1 mvn archetype:generate -DgroupId=com.mytutorial -
```

Step 2: Import it into eclipse via File –> Import –> Maven –> “Existing Maven Projects”



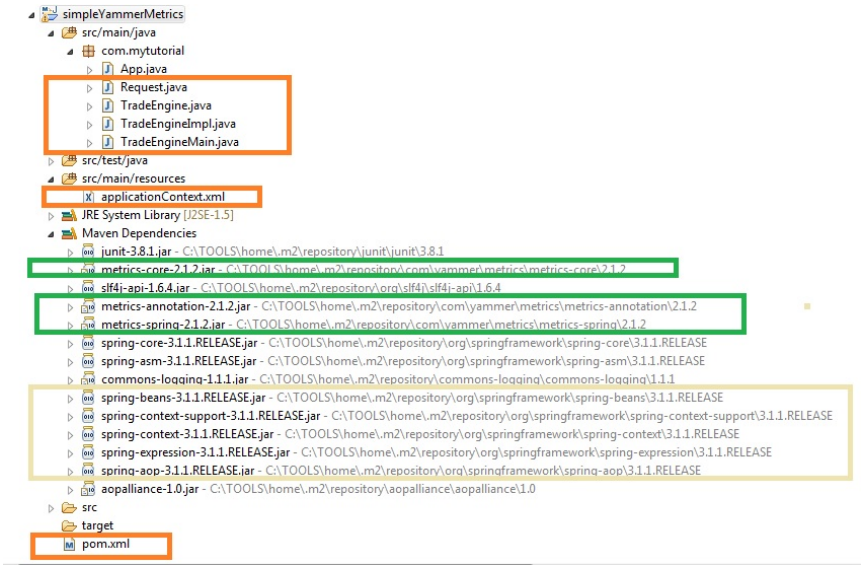
Step 3: It should look like as shown below after completing the steps below.

- Java 8 (24)
- JVM (6)
- Reactive Programm
- Swing & AWT (2)
- JEE Interview Q&A (3
  - JEE Overview (2)
  - Web basics (8)
  - WebService (11)
  - JPA (2)
  - JTA (1)
  - JDBC (4)
  - JMS (5)
  - JMX (3)
    - 5 JMX and MBe
    - Event Driven Pro
    - Yammer metrics
  - JNDI and LDAP (1)
- Pressed for time? Jav
- SQL, XML, UML, JSC
- Hadoop & BigData Int
- Java Architecture Inte
- Scala Interview Q&As
- Spring, Hibernate, & M
- Testing & Profiling/Sa
- Other Interview Q&A f
- Free Java Interviev

16 Technical Key Areas

open all | close all

- Best Practice (6)
- Coding (26)
- Concurrency (6)
- Design Concepts (7)
- Design Patterns (11)
- Exception Handling (3
- Java Debugging (21)
- Judging Experience In
- Low Latency (7)



Step 4: Update the pom.xml file as shown below

```
<project xmlns="http://maven.apache.org/POM/4.0.
  xsi:schemaLocation="http://maven.apache.org/PO
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mytutorial</groupId>
  <artifactId>simpleYammerMetrics</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>simpleYammerMetrics</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <!-- Yammer metrics dependencies-->
    <dependency>
      <groupId>com.yammer.metrics</groupId>
      <artifactId>metrics-core</artifactId>
      <version>2.1.2</version>
    </dependency>
    <dependency>
      <groupId>com.yammer.metrics</groupId>
      <artifactId>metrics-annotation</artifactId
      <version>2.1.2</version>
    </dependency>

    <dependency>
      <groupId>com.yammer.metrics</groupId>
      <artifactId>metrics-spring</artifactId>
      <version>2.1.2</version>
    </dependency>


  </dependencies>
</project>
```


- Memory Managemen
- Performance (13)
- QoS (8)
- Scalability (4)
- SDLC (6)
- Security (13)
- Transaction Managen

Step 5: The dummy Request class.

```
1 package com.mytutorial;
2
3 public class Request {
4
5 }
```

Step 6: The dummy trading engine interface.

```
1 package com.mytutorial;
2
3 public interface TradeEngine {
4
5     abstract void execute(Request... requests);
6 }
```

Step 7: The dummy trading engine implementation with yammer annotations to gather metrics.

```
1  package com.mytutorial;
2
3  import java.util.concurrent.atomic.AtomicInteger
4
5  import com.yammer.metrics.annotation.ExceptionMe
6  import com.yammer.metrics.annotation.Gauge;
7  import com.yammer.metrics.annotation.Timed;
8
9  public class TradeEngineImpl implements TradeEng
10
11   @Gauge
12   private final AtomicInteger currentRequests =
13
14
15   @Timed
16   @ExceptionMetered
17   public void execute(Request... requests) {
18     executingRequests(requests.length);
19     try {
20       Thread.sleep(2000); //just to emulate some
21     } catch (InterruptedException e) {
22       // TODO Auto-generated catch block
23       e.printStackTrace();
24     }
25   }
26
27   private void executingRequests(int count) {
28     this.currentRequests.addAndGet(count);
29   }
30
31
32 }
```

Step 8: The spring context file to wire up Yammer and Java class defined above.

src/amin/resources/applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/sch
    xmlns:aop="http://www.springframework.org/sc
    xmlns:metrics="http://www.yammer.com/schema/
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    xsi:schemaLocation="http://www.springframewo
                        http://www.springframewo
                        http://www.yammer.com/sc



    <metrics:metrics-registry id="trade-metrics"
    <metrics:health-check-registry id="trade-hea
    <metrics:annotation-driven metrics-registry=
    <metrics:jmx-reporter metrics-registry="trad
    <bean id="tradeEngine" class="com.mytutorial


</beans>

```

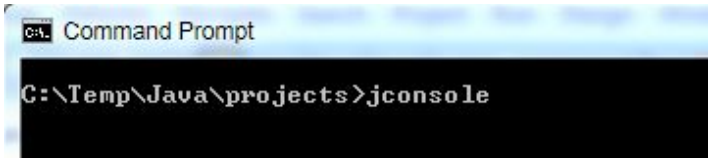
Step 8: The main class that starts the trading engine and fires requests every 5 seconds in a continuos loop.

```
package com.mytutorial;

import org.springframework.context.ApplicationCo
import org.springframework.context.support.Class

public class TradeEngineMain {

  public static void main(String[] args) {

    ApplicationContext context = new ClassPathXm
        new String[] { "applicationContext.xml"

    TradeEngine tradeEngine = (TradeEngine) cont

    try {
      while (true) {
        Request[] requests = new Request[2];
        requests[0] = new Request();
        requests[1] = new Request();
        tradeEngine.execute(requests);

        Thread.sleep(5000);
      }
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
```

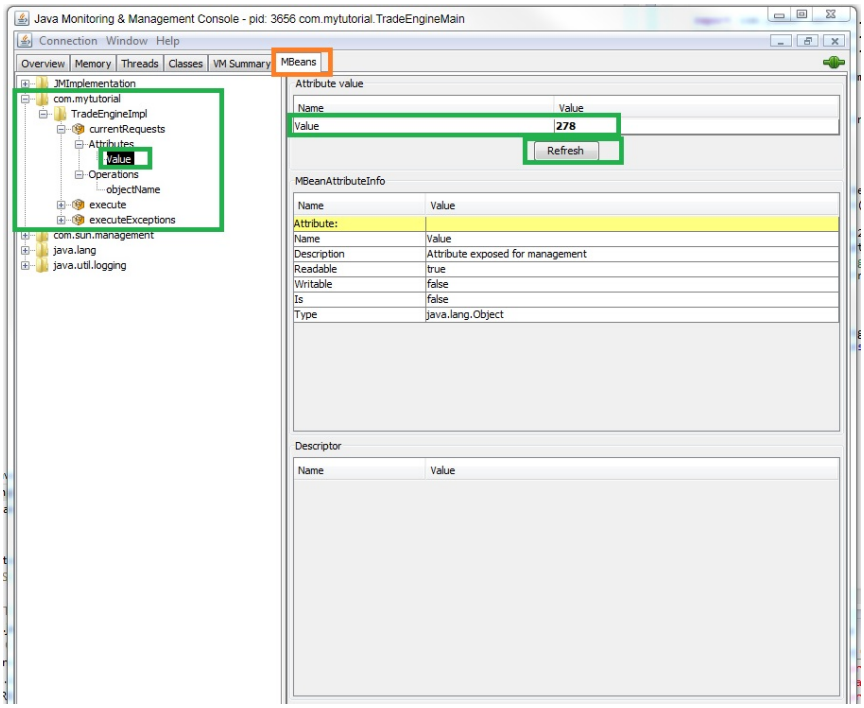
```
27
28    }
29
30 }
```

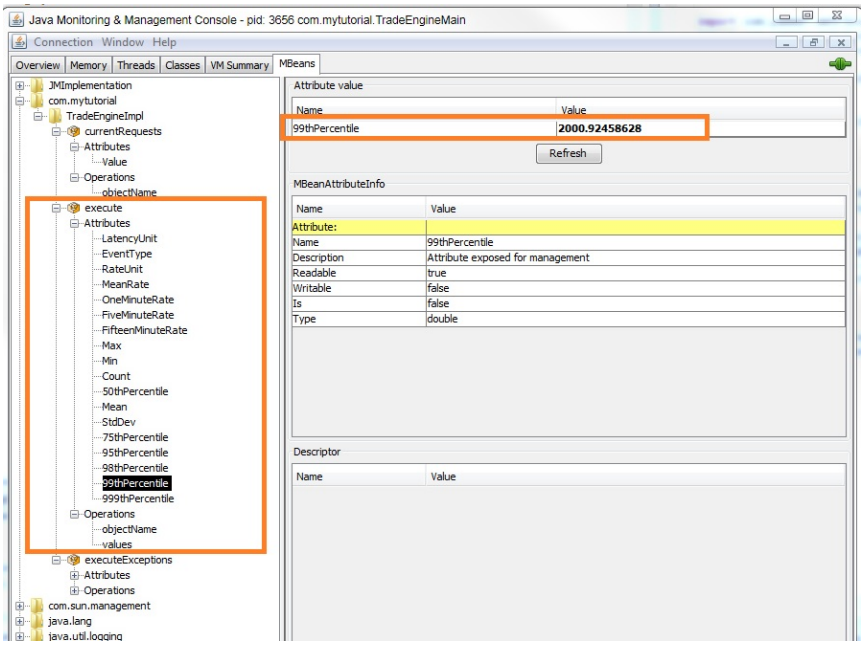
Step 9: Run the "**TradeEngineMain**", which starts the trading engine in a continous loop.

Step 10: Whilst the engine is running, from a DOS prompt invoke the JMX window by typing "jconsole". You can type "jps" to find the process id of the trading engine that is running.



Step 11: You can see the metrics in the jconsole JMX window as shown below. Yammer supports more outputs including RESTful web service to report stats.





Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

766 views

18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ FIX and Java tutorial with QuickFIXj

Exposing a Java class as a MXBean (or MBean) tutorial with jconsole ›

Posted in JMX, Metrics Tutorial

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

© Disclaimer

 ↑