# 2. Asynchronous processing in Java real life examples – part-2

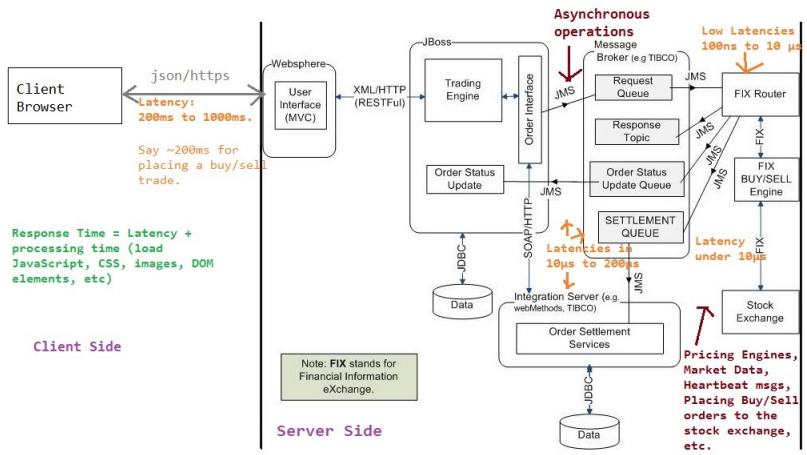
Posted on October 7, 2015 by Arulkumaran Kumaraswamipillai

Extends: 1. Asynchronous processing in Java real life examples – part-1.

**Example 3:** Trade execution reports are received & processed asynchronously

If you are working in an online trading application, you may want the functionality to queue trades and process them when the stock market opens. You also asynchronously receive the execution report statuses like partially-filled, rejected, fully filled, etc from the stock market. The message oriented middle-wares provide features like guaranteed delivery with store-and-forward mechanism, no duplicates, and transaction management for enterprise level program-to-

program communications by sending and receiving messages asynchronously (or synchronously). The diagram below gives a big picture.




Trade execution reports are received and processed asynchronously

## Example 4: Asynchronous logging

You may have a requirement for stringent logging, auditing or performance metrics gathering. Processing these non-functional requirements asynchronously and non-intrusively can make your system perform and scale better. For example, you could send the log messages to a queue to be processed later asynchronously by a separate process running on the same machine or a separate machine. The performance metrics can be processed asynchronously as well. here is a working example with relevant code snippets.

For example, a trading application may have a number of synchronous and asynchronous moving parts and metrics needs to be recorded for various operations like placing a trade on to a queue, receiving asynchronous responses from the stock market, correlating order ids, linking similar order ids, etc. A custom metrics gathering solution can be accomplished by logging the relevant metrics to a database and then running relevant aggregate queries or writing to a file system and then running PERL based text searches to aggregate the results to a “csv” based file to be opened and analyzed in a spreadsheet with graphs. In my view, writing to


⊞ Free Java Interviev

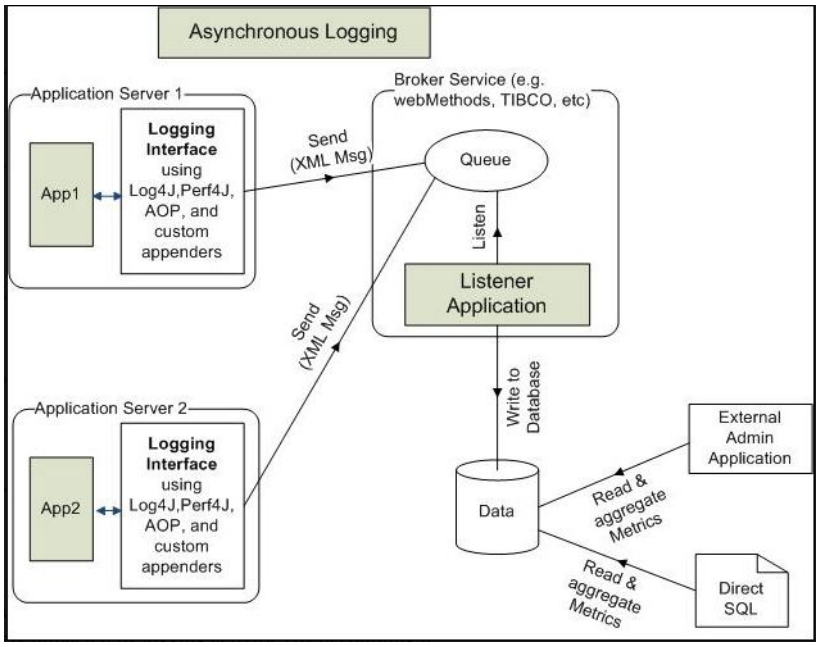
## 16 Technical Key Areas

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

## 80+ step by step Java Tutorials

open all | close all
⊞ Setting up Tutorial (6)
⊞ Tutorial - Diagnosis (2
⊞ Akka Tutorial (9)
⊞ Core Java Tutorials (2
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

a database provides a greater flexibility. For example, in Java, the following approach can be used.




Asynchronous logging

**Step 1:** Use log4j JMS appender or a custom JMS appender to send log messages to a queue. This ensures that your application's performance is not adversely impacted by logging activities by decoupling it.

**Step 2:** Use this appender in your application via Aspect Oriented Programming (AOP – e.g Spring AOP, AspectJ, etc) or dynamic proxy classes to non-intrusively log relevant metrics to a queue. It is worth looking at Perf4j and context based logging with MDC (Mapped Diagnostic Contexts) or NDC (Nested Diagnostic Contexts) to log on a per thread basis to correlate or link relevant operations. Perf4J is a great framework for performance logging. It's non-intrusive and really fills the need for accurate performance logging. The Perf4j provides features like a command line tool to generate aggregated results and graphs from raw log files, ability to expose performance statistics as JMX attributes and to send notifications when statistics exceed specified thresholds, custom log4j appenders, and AOP aspects that allow non obtrusive statements when used with Spring AOP.

The Perf4J is for

```
1 System.currentTimeMillis( );
2
```

as Log4J is for

```
1 System.out.println(.....);
2
```

**Step 3:** A stand-alone listener application needs to be developed to dequeue the performance metrics messages from the queue and write to a database or a file system for further analysis and reporting purpose. This listener could be written in Java as a JMX service using JMS or via broker service like webMethods, TIBCO, etc.

**Step 4:** Finally, relevant SQL or regular expression based queries can be written to aggregate and report relevant metrics in a customized way.

The data captured could be an Event object containing

```
1  private final String environment;           // s
2  private final Long timestamp;               // s
3  private final Long duration;                // s
4  private final String thread;                // L
5  private final String component;             // c
6  private final Stage stage;                  // e
7  private final String source;                // L
8  private final String context;               // L
9  private final String process;               // w
10 private final String item;                  // w
11 private final String level;                 // L
12 private final String message;               // L
13 private final String exceptionMessage;      // L
14 private final String exceptionStacktrace;   // L
15
16
```

An XML based library like Xtream can be used to serialize

and deserialize the messages. Here is some pseudo code in Java.

### Let's look at some sample code:

**STEP-1:** Define the custom appender in Log4j. In Log4J, appenders are responsible for handling the output and transport of logging data ('output destinations'). There are appenders for writing log data to files, for logging over the network or to write the logging output to the terminal, just to name a few.

```
1  package com;
2
3  public class EventJmsXmlAppender extends Appende
4    ...
5    public boolean requiresLayout( )
6    {
7      return true;
8    }
9
10   protected void append(LoggingEvent event)
11   {
12     try
13      {
14        //construct and the send the log metrics
15        JMSServiceFactory jmsService = JMSService
16        QueueConnectionFactory qFactory = (QueueC
17        QueueConnection qConnection = qFactory.cr
18        qSession qSession = qConnection.createqSe
19        Queue q =  (Queue)jmsService.getTargetJMS
20        QueueSender qSender = qSession.createSend
21        TextMessage msg =  qSession.createTextMes
22        msg.setText(getLayout().format(event));
23        qSender.send(msg);
24
25      }
26      catch(ServiceNotFoundException snfe)
27      {
28        //handle Exception
29      }
30      catch(JMSException jmse)
31      {
32       //handle Exception
33   }
34      }
35      ...
36 }
37
38
```

In the **log4j.xml** file define the custom appender.

```
...
 <appender name="events" class="com.EventJmsXmlA
     <errorHandler class="org.jboss.logging.util
     <param name="Target" value="System.out"/>
     <param name="Threshold" value="INFO"/>

     <layout class="org.apache.log4j.PatternLayo
        <!-- The default pattern: Date Priority
        <param name="ConversionPattern" value="%
     </layout>
 </appender>


 <logger name="com.LoggingInterceptor" additivit
      <level value="info" />
      <appender-ref ref="events" />
 </logger>

 ...

```

**Step 2:** Define the interceptor that logs metrics using the custom JMS appender. The interceptor will be invoked on actual method invocation to log the time. The interceptor will forward the request to actual method call via invocation.proceed( ) method call. After invoking the actual method, the time duration for the method execution is calculated. The LOG.info method call will be using the previously defined JMS appender.

```
package com;

public class LoggingInterceptor implements Metho

    private static final Logger LOG = Logger.get

    @Override
    public Object invoke(MethodInvocation invoca
        long begin = System.currentTimeMillis( )
        ...
        try {
          ...
          Object result = invocation.proceed( );
          ...
          LOG.info("Duration = " + System.curren
          return result;
        } catch (ResponseUnavailableException e)
            ....
            throw e;
        } catch (Throwable e) {
            throw e;
        } finally {
           ...
        }
    }
   ..
}
```

```
28
29
```

In Spring config file wire up the required service(s) and interceptor(s). The TradingService will be making use of the interceptor(s) to log performance metrics.

```
1  <bean id="loggingInterceptor" class="com.Logging
2
3      <bean id="abstractService" abstract="true"
4          <property name="interceptorNames">
5              <list>
6                  <value>loggingInterceptor</value
7              </list>
8          </property>
9      </bean>
10     <bean id="tradingService" parent="abstractSe
11         <property name="proxyInterfaces">
12             <value>com.TradingService</value>
13         </property>
14         <property name="target">
15             <bean class="com..TradingServiceImpl
16                 <constructor-arg><ref bean="..."
17                 <constructor-arg><ref bean=".." 
18             </bean>
19         </property>
20     </bean>
21
22
```

**STEP 3:** Write a stand-alone JMS listener to retrieve the XML event massages from the queue “jms/EventsQueue” and write to a database table or a file for further analysis and reporting. This can be done via a MDB (a Message Driven Bean in the EJB specification), a Spring based JMS listener, or a listener configured via a Message Oriented Broker like web Methods or TIBCO broker.

# Popular Member Posts

♦ 11 Spring boot interview questions & answers

**850 views**

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

**768 views**

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

**399 views**

18 Java scenarios based interview Questions and Answers

**387 views**

♦ 7 Java debugging interview questions & answers

**308 views**

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

**305 views**

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

**297 views**

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

**293 views**

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

**246 views**

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

**204 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**

‹ Unit Test Hibernate – DAO Layer with JPA, Spring & HSQLDB

Debugging java.security.cert.CertificateException: Certificates do not conform to algorithm constraints ›

**Posted in** Java Architecture Interview Q&A, member-paid

# 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





↑