Whats the best way to process an asynchronous queue continuously in Java?

Asked 15 years, 10 months ago Modified 7 years, 7 months ago

Viewed 11k times

8

I'm having a hard time figuring out how to architect the final piece of my system. Currently I'm running a Tomcat server that has a servlet that responds to client requests. Each request in turn adds a processing message to an asynchronous queue (I'll probably be using JMS via Spring or more likely Amazon SQS).

The sequence of events is this:

Sending side:
1. Take a client request
2. Add some data into a DB related to this request with a unique ID
3. Add a message object representing this request to the message queue

Receiving side:
1. Pull a new message object from the queue
2. Unwrap the object and grab some information from a web site based on information contained in the msg object.
3. Send an email alert

4. update my DB row (same unique ID) with the information that operation was completed for this request.

I'm having a hard figuring out how to properly deal with the receiving side. On one hand I can probably create a simple java program that I kick off from the command line that picks each item in the queue and processes it. Is that safe? Does it make more sense to have that program running as another thread inside the Tomcat container? I will not want to do this serially, meaning the receiving end should be able to process several objects at a time -- using multiple threads. I want this to be always running, 24 hours a day.

What are some options for building the receiving side?

java asynchronous queue jms

Share

Improve this question

Follow

edited Apr 25, 2017 at 17:28

Cœur

38.6k ● 26 ● 202 ● 276

asked Feb 5, 2009 at 2:33

Ish

1,895 ● 5 ● 24 ● 32

In case anyone is interested in what I finally ended up doing. I used Amazon's SQS and have a java client (utilizes spring framework) that polls the queue. When it finds a msg it processes it and goes back into wait state. I might add

Quartz threading, for now i just kick off multiple processes.
– Ish Mar 24, 2009 at 19:19

I am facing a similar issue. I would like to know how was the Java Client implemented. I hope it does not run in a infinite while loop and pools for the message?
– TheMonkWhoSoldHisCode Oct 28, 2014 at 18:58

4 Answers

Sorted by: Highest score (default)

3

"On one hand I can probably create a simple java program that I kick off from the command line that picks each item in the queue and processes it. Is that safe?"

What's unsafe about it? It works great.

"Does it make more sense to have that program running as another thread inside the Tomcat container?"

Only if Tomcat has a lot of free time to handle background processing. Often, this **is** the case -- you have free time to do this kind of processing.

However, threads aren't optimal. Threads share common I/O resources, and your background thread may slow down the front-end.

Better is to have a JMS queue between the "port 80" front-end, and a separate backend process. The back-end process starts, connects to the queue, fetches and executes the requests. The backend process can (if necessary) be multi-threaded.

Share Improve this answer

Follow

answered Feb 5, 2009 at 2:43

S.Lott

391k ● 82 ● 517 ● 788

3

If you are using JMS, why are you placing the tasks into a DB?

You can use a durable Queue in JMS. This would keep tasks, even if the JMS broker dies, until they have been acknowledged. You can have redundant brokers so that if one broker dies, the second automatically takes over. This could be more reliable than using a single DB.

Share Improve this answer

Follow

answered Feb 5, 2009 at 2:45

Peter Lawrey

533k ● 82 ● 767 ● 1.1k

> Because I will retrieve some information from the target web site and place it alongside the row in the DB. This information will then need to be retrieved by a client at some later point in time. I'm not really using db for redundancy as much as for storage of data for later retrieval. – Ish Feb 5, 2009 at 3:04

1

If you are already using Spring, check out DefaultMessageListenerContainer. It allows you to create a POJO message driven bean. This can be used from within an existing application container (your WAR file) or as a separate process.

Share Improve this answer

answered Feb 5, 2009 at 2:42

Follow

John Meagher
24.6k ● 14 ● 56 ● 57

In this case is the consumer polling the queue continously or does it get notified somehow? – Ish Feb 6, 2009 at 6:24

I'm pretty sure the DefaultMessageListenerContainer polls. The nice thing about it is that it hides the poll/notification issue from you. You just implement a jms MessageListener and do whatever work you need to do. – John Meagher Feb 6, 2009 at 23:06

0

I've done this sort of thing by hosting the receiver in an app server, weblogic in my case, but tomcat works fine, too. Don't poll the queue, use an event-based model. This could be hand-coded or it could be a message-driven web service. If the database update is idempotent, you could update the database and send the email, then issue the commit on the queue. It's not a problem to have several threads that all read from the same queue.

I've use various JMS solutions, including tibco, activemq (before apache subsumed it) and joram. Joram was the more reliable opensource solution, but that may have changed now that it's part of apache.

Share Improve this answer

Follow

answered Feb 5, 2009 at 2:41

Don Branson
13.7k ● 10 ● 61 ● 102

Can you explain how to implement a event-based model? – TheMonkWhoSoldHisCode Oct 28, 2014 at 19:00

Have a look at docs.oracle.com/cd/E13222_01/wls/docs90/jms/…, the section titled "Receiving Messages Asynchronously."
– Don Branson Oct 28, 2014 at 21:28