Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

Register | Login | Logout | Contact Us

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Interview › JEE Interview Q&A › JMS › JMS versus AMQP, Enterprise Integration Patterns (EIP), and Spring Integration versus Apache Camel

JMS versus AMQP, Enterprise Integration Patterns (EIP), and Spring Integration versus Apache Camel

Posted on February 8, 2015 by Arulkumaran Kumaraswamipillai — No Comments ↓

0 Like Share Tweet 0 G+1 Share

Q. Why do you need **AMQP** when there is **JMS**?
A. AMQP stands for Advanced Message Queuing Protocol, and was developed to address the problem of interoperability by creating a standard for how messages should be structured and transmitted between platforms the same way as SMTP, HTTP, FTP, etc. have created interoperable systems. This standard binary wire level protocol for

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

open all | close all

- ⊞ Ice Breaker Interview
- ⊞ Core Java Interview (
- ⊟ JEE Interview Q&A (3
  - ⊞ JEE Overview (2)
  - ⊞ Web basics (8)
  - ⊞ WebService (11)
  - ⊞ JPA (2)
  - ⊞ JTA (1)
  - ⊞ JDBC (4)
  - ⊟ JMS (5)
    - ♦ 16 FAQ JMS ir
    - Configuring JMS
    - JMS versus AM(
    - Spring JMS with
    - Spring JMS with
  - ⊞ JMX (3)
  - ⊞ JNDI and LDAP (1)
- ⊞ Pressed for time? Jav
- ⊞ SQL, XML, UML, JSC
- ⊞ Hadoop & BigData Int
- ⊞ Java Architecture Inte

messaging would therefore allow hetrogeneous disparate systems between and within companies to exchange messages regrdless of the message broker vendor or platform.
RabbitMQ, **Apache Qpid**, **StormMQ**, etc are open source message broker softwares (i.e. **MOM** – message-oriented middlewares) that implements the Advanced Message Queuing Protocol (**AMQP**).

Q. How does **AMQP** differ from **JMS**?
A. JMS is a standard messaging API for the Java platform. It provides a level of abstraction that frees developers from having to worry about specific implementation and wire protocols. This is similar to the JDBC API that allows you to easily switch databases. With JMS, you can switch from one JMS complian message broker (e.g. Web Methods) with another one (e.g. MQSeries or WebspehreMQ) with little or no changes to your source code. It also provides interoperability between other JVM based languages like Scala and Groovy. Altough JMS brokers can be used in .NET applications, the whole JMS specification does not guarantee interoperability, and integration between Java to .NET or Java to Ruby, is proprietary and can be quite tricky. In scenarios where you want to send a message from a Java based message producer to a .NET based message consumer, then you need a message based cross platform interoperability that is what AMQP does. With AMQP, you can use any AMQP compliant client library, and AMQP compliant message broker.

Q. What are the different alternatives to integrate various enterprise systems?
A.

Alternative 1: Custom Solution. Implement an individual solution that works for your problem without separating problems into little pieces. For example, use Apache CXF for Web services, overnight batch job runs to load data feeds, JMS for messaging, etc. This is more suited for small use cases. This has higher maintainability and developer effort. The developer has to handle errors, service retries,

⊞ Scala Interview Q&As
⊞ Spring, Hibernate, & I
⊞ Testing & Profiling/Sa
⊞ Other Interview Q&A 1
⊞ ▶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)

transactional management, etc. This is suited if you want to integrate one or two applications using one or two protocols.
Alternative 2: Using an (opensource) integration framework like Spring Integration or Apache Camel. This helps you integrate systems in a standardised way adhering to the enterprise integration patterns (EIP). Apache Camel is a light weight integration framework that allows you to use HTTP, FTP, JMS, EJB, JPA, RMI, JMS, JMX, LDAP, and Netty to name a few. You use the same concept to integrate various protocols in Apache Camel. This increases maintainability and reduces developer effort. This is more suited if you want to integrate several applications with different protocols.
Alternative 3: Using an ESB (Enterprise Service Bus) to integrate your applications. For example, Oracle Service Bus, TIBCO ESB, webMethods, etc. Under the hood, the ESB also uses an integration framework and provide more services and management functionalities like monitoring, high availability, clustering, graphical user inteface for routing and configuring, etc. Usually, an ESB is a complex and powerful product with a higher learning curve. Suited for very large integration projects. Projects requiring BPM (Business Process Managemnt) integration and other integrated services like monitoring, clustering, etc.

Q. What is an architecture that enables separate applications to work together, but in a de-coupled fashion such that applications can be easily added or removed without affecting the others?
A. This is achieved via a Message Oriented Middleware (aka a message bus).

Q. How can the caller be sure that exactly one receiver will receive the document or perform the call?
A. Use the point-to-point channel

Q. How can the sender broadcast an event to all interested receivers?
A. Use the publish subscribe channel.

Q. What will the messaging system do with a message it cannot deliver?

Spring & HIbernate Tu
Tools Tutorials (19)
Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all
Can you write code? (
♦ Complete the given
Converting from A to I
Designing your classe
Java Data Structures
Passing the unit tests
What is wrong with th
Writing Code Home A
Written Test Core Jav
Written Test JEE (1)

How good are your .....?

open all | close all
Career Making Know-
Job Hunting & Resum

A. Put it on the dead letter channel.

Q. How can the sender make sure that a message will be delivered, even if the messaging system fails?
A. Use the “guaranteed delivery” mechanism.

Q. What are the diffrent ways to route messages?
A. EIP (Enterprise Integration Patterns) define different types of rules based routing to solve common enterprise intergration problems. Like GoF design patterns is the EIP allows integration architects and designers to share a common vocabulary.

1. **Content based routing** uses XPath predicates to route messages based on the message content. **Content enricher** supplements the original message with additional relevant information recieved from the other sources.
2. **Splitter** provides the EIP service engine to split messages into separate parts based on the XPATH expression. Splits a composite message into a series of individual message parts.
3. **Split Aggregator** is used to collect and store individual message parts until a complete set of co-related message parts has been recieved. Once all the related parts have been recieved, they are aggregated to form a single message.
4. **Static Recipent List** based routing inspects an incoming message, and depending upon the number of recipients mentioned in the list, it can forward the message to all channels associated with the “recipients list”.
5. A **Resquencer** is used to get a stream of related but out of sequence messages back into correct order. Because individual messages may follow different routes, some messages are likely to pass through the processing steps sooner than others, resulting in the messages getting out of order. A resequencer usually does not modify the message contents.
6. A **message filter** is a processor that eliminates undesired messages based on specific criteria. Filtering is controlled by specifying a predicate in the filter: when the predicate is true, the incoming message is allowed to pass; otherwise, it is blocked.

A message filter usually does not modify the message contents.

Q. How would you deal with large message volumes?
A.

- You can reduce the data volume with the use of "**Claim Check**" pattern, which allows you to replace message content with a claim check (a unique key), which can be used to retrieve the message content at a later time. The message content will be stored temporarily in a persistent store like a database or file system. This pattern is very useful when message content is very large and not all components require all information.
- A **Content Filter** can be used to remove unwanted data elements from a message. It is useful to simplify the structure of the message. Very often, messages are represented as tree structures containing many levels of nested, repeating groups because they are modeled after generic, normalized database structures. Very often, this level of nesting is superfluous and a Content Filter can be used to 'flatten' the hierarchy into a simple list of elements that can be more easily understood and processed by other systems.

Q. How would you go about choosing between **Spring Integration** and **Apache Camel** to solve common integration problems?
A.

Spring Integration provides an extension to the Spring programming model to support the well-known Enterprise Integration Patterns while building on the Spring Framework's existing support for enterprise integrationSpring Integration is more suited, if you already have got a Spring project and need to add some integration stuff to it. It requires almost no additional effort to learn Spring Integration if you know Spring itself. Nevertheless, Spring Integration only offers very rudimenary support for technologies such as AMQP, Spring Application Events, Feeds (e.g, RSS/ATOM), File, FTP, FTPS, SFTP, Gemfire, Groovy, HTTP (REST), TCP/IP, JDBC, JMS, JMX, Mail (IMAP/IDLE/POP3), MongoDB, Redis, RMI, Twitter, Web Services (SOAP), and XMPP. Integrations are implemented by writing a lot of XML based DSL(without a real DSL – Domain Specific Language). Spring Integration can be though of as a catch up game to Apache Camel as JavaEE did a catch up with Spring.

Apache Camel is a powerful open source integration framework based on known Enterprise Integration Patterns with powerful support for integration with core Spring. Apache Camel is almost identical to Mule ESB, and offers many components (even more than Spring Integration) for almost every technology you could think of. If there is no component available, you can create your own component very easily starting with a Maven archetype. Camel also supports a Spring based XML configuration as well as a "DSL" for Java, Groovy, and Scala. The benefits of using the Java DSL is that your IDE can auto complete your code as you start typing, rather than having to mess around with buckets of XML. The Java DSL is also very expressive as you can mix and match your own code within the language for Expression or Predicate evaluations. So, it has better readability and there are commercial tools like "Fuse IDE" for generating XML based DSL code.

Mule ESB is another choice and as the name suggests, it is an ESB including additional bells and whistles. This can be compared to "Apache ServiceMix", which is an extension to Apache Camel. Mule also only offers XML based DSL. The Mule Studio is a visual designer. Mule does provide proprietary connector support for systems like SAP, Tibco Rendevous, PayPal, Sibel CRM, IBM's CICS, etc.

So, the decision is not clear cut and it depends on your needs.

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**

‹ Dealing with concurrent modifications in Java

Understanding Overriding, Hiding, and Overloading in Java? How does overriding give polymorphism? ›

Posted in JMS, Understanding JEE

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

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.