Register | Login | Logout | Contact Us

Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

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 › Pressed for time? Java/JEE Interview FAQs › Java Application Architecture Interview Essentials › 04: ♦ How to go about designing a medium size JEE application?

04: ♦ How to go about designing a medium size JEE application?

Posted on May 4, 2015 by Arulkumaran Kumaraswamipillai

A very popular open-ended question to judge your Java/JEE experience.

Q. How would you go about designing a medium sized JEE application?
A. Don't start with 3-tier architecture, logical layers, Spring/Hibernate framework etc. The phases of designing any systems are:

Requirements Gathering => Baseline Architecture => Design Alternatives & impact analysis => Choice of

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
⊟ Pressed for time? Jav
⊞ Job Interview Ice B
⊞ FAQ Core Java Jol
⊞ FAQ JEE Job Inter
⊞ FAQ Java Web Ser
⊟ Java Application Ar
001A: ♦ 7+ Java
001B: ♦ Java arc
04: ♦ How to go
⊞ Hibernate Job Inter
⊞ Spring Job Intervie
⊞ Java Key Area Ess
⊞ OOP & FP Essenti
⊞ Code Quality Job I
⊞ SQL, XML, UML, JSC
⊞ Hadoop & BigData Int
⊟ Java Architecture Inte
♥♦ 01: 30+ Writing

technologies/frameworks/tools etc => Capacity/Infrastructure planning => Logical/physical modelling.

#1. Ask the right questions: Gather functional & non functional requirements

Non functional requirements include:

1. How many transactions per minute or hour should the system handle?
2. How many concurrent users should it handle?
3. Where does it get the data from? database, web services, topics/queues, etc. What integration styles are required? Identify the communication protocols and message formats between the client and server.
4. Any requirements to externalize business rules?
5. Any special security requirements like two factor authentication, two-way SSL, WS-security to encrypt credit card details, etc
6. Data retention, auditing, logging, fault tolerance, system monitoring, and disaster recovery requirements.
7. Any load balancing and caching requirements.
8. Any static resources to be on CDN (Content Delivery Networks) for performace

#2. Draw a proposed solution diagram.

You must know the different integration styles and high level architectures like SOA, WOA, MOM, EDA, etc. Identify all the key components of the solution. Describe how each high level requirement is going to be addressed by the overall solution and its components. This is known as the **baseline architecture**.

- What should be a typical Java EE architecture?
- Java/JEE Architectural Patterns with lots of diagrams
- Java/JEE Integration Patterns with lots of diagrams

001A: ♦ 7+ Java int
001B: ♦ Java archit
01: ♥♦ 40+ Java W
02: ♥♦ 13 Tips to w
03: ♦ What should
04: ♦ How to go ab
05: ETL architectur
1. Asynchronous pı
2. Asynchronous pı
Scala Interview Q&As
Spring, Hibernate, & I
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 I
Low Latency (7)
Memory Managemen
Performance (13)
QoS (8)
Scalability (4)
SDLC (6)
Security (13)
Transaction Managen

80+ step by step Java

Each functional and non-functional requirement needs to be mapped to the technical solution. Gaps in requirements need to be identified.

#3. Identify the design alternatives, and analyze pros and cons of each alternative.

If you just take technical design alone, there will be many possible design alternatives, and each alternative has its own pros and cons along with likely trade-offs to be made in your design decisions. You will have to list the relevant **assumptions**, **potential risks**, **likelihood and impacts** of those risks to the business. At times, **tactical solutions** need to be favored over **strategical solution** due to business demands, budgetary constraints, and time to market. List all design choices and pros and cons for each choice. It is also imperative to not cut corners as a particular choice might look attractive now, but in a longer term require more rework and budget.

So, design is often all about making the informed choices and trade offs. You make the design choices based on the functional and non-functional requirements, budgetary and non-budgetary constraints, environmental and political factors, and collective experience. Your architectural decisions need to adhere to the frameworks, policies and standards in place and need to be approved by the relevant stake holders, architecture review board, superiors, and peers. So, this requires good communication skills both written and oral to convince the relevant stake holders. You need to look at things from both business and technology perspective, and present it based on the target audience without too much technical jargon.

Examples:

1. RESTful web service Vs. SOAP web service
2. Web Service Vs. messaging using a MOM
3. Build new component, reuse existing, or buy, etc

Look at from different key areas like Transaction Management, Security, Performance, etc. Click on each

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)

100+ Java pre-interview coding tests

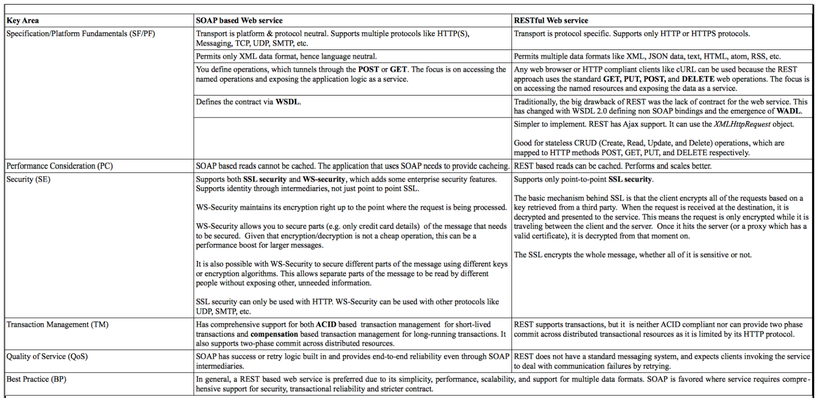
open all | close all
⊞ Can you write code? (
⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

How good are your .....?

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

diagram to enlarge.

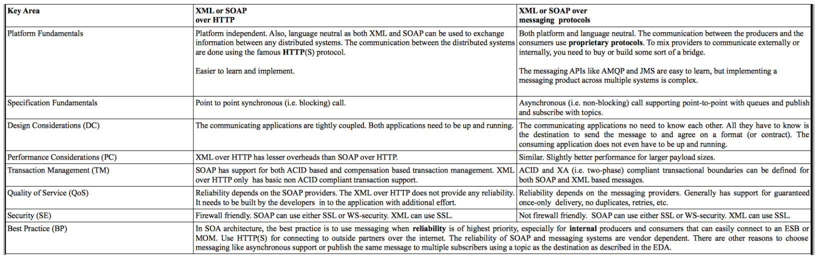
Example 1: SOAP Vs RESTful



Key Area	SOAP based Web service	RESTful Web service
Specification/Platform Fundamentals (SF/PF)	Transport is platform & protocol neutral. Supports multiple protocols like HTTP(S), Messaging, TCP, UDP, SMTP, etc.	Transport is protocol specific. Supports only HTTP or HTTPS protocols.
	Permits only XML data format, hence language neutral.	Permits multiple data formats like XML, JSON data, text, HTML, atom, RSS, etc.
	You define operations, which tunnels through the **POST** or **GET**. The focus is on accessing the named operations and exposing the application logic as a service.	Any web browser or HTTP compliant clients like cURL can be used because the REST approach uses the standard **GET, PUT, POST,** and **DELETE** web operations. The focus is on accessing the named resources and exposing the data as a service.
	Defines the contract via **WSDL**.	Traditionally, the big drawback of REST was the lack of contract for the web service. This has changed with WSDL 2.0 defining non SOAP bindings and the emergence of **WADL**.
		Simpler to implement. REST has Ajax support. It can use the *XMLHttpRequest* object. Good for stateless CRUD (Create, Read, Update, and Delete) operations, which are mapped to HTTP methods POST, GET, PUT, and DELETE respectively.
Performance Consideration (PC)	SOAP based reads cannot be cached. The application that uses SOAP needs to provide cacheing.	REST based reads can be cached. Performs and scales better.
Security (SE)	Supports both **SSL security** and **WS-security**, which adds some enterprise security features. Supports identity through intermediaries, not just point to point SSL. WS-Security maintains its encryption right up to the point where the request is being processed. WS-Security allows you to secure parts (e.g. only credit card details) of the message that needs to be secured. Given that encryption/decryption is not a cheap operation, this can be a performance boost for larger messages. It is also possible with WS-Security to secure different parts of the message using different keys or encryption algorithms. This allows separate parts of the message to be read by different people without exposing other, unneeded information. SSL security can only be used with HTTP. WS-Security can be used with other protocols like UDP, SMTP, etc.	Supports only point-to-point **SSL security**. The basic mechanism behind SSL is that the client encrypts all of the requests based on a key retrieved from a third party. When the request is received at the destination, it is decrypted and presented to the service. This means the request is only encrypted while it is traveling between the client and the server. Once it hits the server (or a proxy which has a valid certificate), it is decrypted from that moment on. The SSL encrypts the whole message, whether all of it is sensitive or not.
Transaction Management (TM)	Has comprehensive support for both **ACID** based transaction management for short-lived transactions and **compensation** based transaction management for long-running transactions. It also supports two-phase commit across distributed resources.	REST supports transactions, but it is neither ACID compliant nor can provide two phase commit across distributed transactional resources as it is limited by its HTTP protocol.
Quality of Service (QoS)	SOAP has success or retry logic built in and provides end-to-end reliability even through SOAP intermediaries.	REST does not have a standard messaging system, and expects clients invoking the service to deal with communication failures by retrying.
Best Practice (BP)	In general, a REST based web service is preferred due to its simplicity, performance, scalability, and support for multiple data formats. SOAP is favored where service requires comprehensive support for security, transactional reliability and stricter contract.	

SOAP Vs RESTful

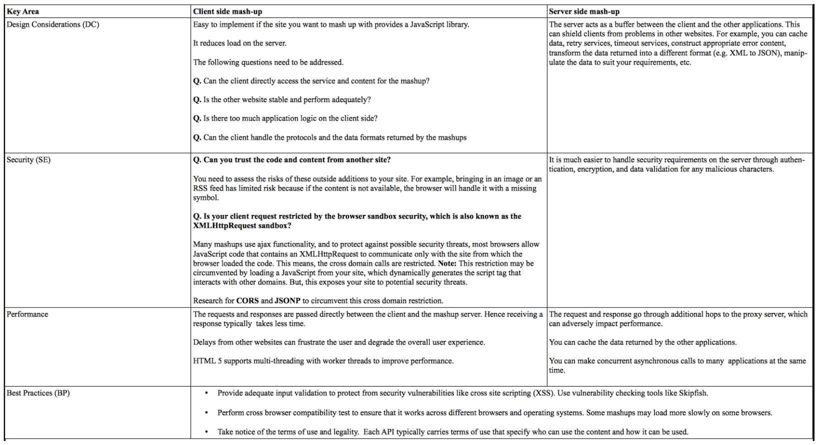
Example 2: HTTP Vs Messaging protocols



Key Area	XML or SOAP over HTTP	XML or SOAP over messaging protocols
Platform Fundamentals	Platform independent. Also, language neutral as both XML and SOAP can be used to exchange information between any distributed systems. The communication between the distributed systems are done using the famous **HTTP**(S) protocol. Easier to learn and implement.	Both platform and language neutral. The communication between the producers and the consumers use **proprietary protocols**. To mix providers to communicate externally or internally, you need to buy or build some sort of a bridge. The messaging APIs like AMQP and JMS are easy to learn, but implementing a messaging product across multiple systems is complex.
Specification Fundamentals	Point to point synchronous (i.e. blocking) call.	Asynchronous (i.e. non-blocking) call supporting point-to-point with queues and publish and subscribe with topics.
Design Considerations (DC)	The communicating applications are tightly coupled. Both applications need to be up and running.	The communicating applications no need to know each other. All they have to know is the destination to send the message to and agree on a format (or contract). The consuming application does not even have to be up and running.
Performance Considerations (PC)	XML over HTTP has lesser overheads than SOAP over HTTP.	Similar. Slightly better performance for larger payload sizes.
Transaction Management (TM)	SOAP has support for both ACID based and compensation based transaction management. XML over HTTP only has basic non ACID compliant transaction support.	ACID and XA (i.e. two-phase) compliant transactional boundaries can be defined for both SOAP and XML based messages.
Quality of Service (QoS)	Reliability depends on the SOAP providers. The XML over HTTP does not provide any reliability. It needs to be built by the developers in to the application with additional effort.	Reliability depends on the messaging providers. Generally has support for guaranteed once-only delivery, no duplicates, retries, etc.
Security (SE)	Firewall friendly. SOAP can use either SSL or WS-security. XML can use SSL.	Not firewall friendly. SOAP can use either SSL or WS-security. XML can use SSL.
Best Practice (BP)	In SOA architecture, the best practice is to use messaging when **reliability** is of highest priority, especially for **internal** producers and consumers that can easily connect to an ESB or MOM. Use HTTP(S) for connecting to outside partners over the internet. The reliability of SOAP and messaging systems are vendor dependent. There are other reasons to choose messaging like asynchronous support or publish the same message to multiple subscribers using a topic as the destination as described in the EDA.	

Over HTTP or Messaging

Example 3: Client side Vs Server side mashups



Key Area	Client side mash-up	Server side mash-up
Design Considerations (DC)	Easy to implement if the site you want to mash up with provides a JavaScript library. It reduces load on the server. The following questions need to be addressed. **Q.** Can the client directly access the service and content for the mashup? **Q.** Is the other website stable and perform adequately? **Q.** Is there too much application logic on the client side? **Q.** Can the client handle the protocols and the data formats returned by the mashups	The server acts as a buffer between the client and the other applications. This can shield clients from problems in other websites. For example, you can cache data, retry services, timeout services, construct appropriate error content, transform the data returned into a different format (e.g. XML to JSON), manipulate the data to suit your requirements, etc.
Security (SE)	**Q. Can you trust the code and content from another site?** You need to assess the risks of these outside additions to your site. For example, bringing in an image or an RSS feed has limited risk because if the content is not available, the browser will handle it with a missing symbol. **Q. Is your client request restricted by the browser sandbox security, which is also known as the XMLHttpRequest sandbox?** Many mashups use ajax functionality, and to protect against possible security threats, most browsers allow JavaScript code that contains an XMLHttpRequest to communicate only with the site from which the browser loaded the code. This means, the cross domain calls are restricted. **Note:** This restriction may be circumvented by loading a JavaScript from your site, which dynamically generates the script tag that interacts with other domains. But, this exposes your site to potential security threats. Research for **CORS** and **JSONP** to circumvent this cross domain restriction.	It is much easier to handle security requirements on the server through authentication, encryption, and data validation for any malicious characters.
Performance	The requests and responses are passed directly between the client and the mashup server. Hence receiving a response typically takes less time. Delays from other websites can frustrate the user and degrade the overall user experience. HTML 5 supports multi-threading with worker threads to improve performance.	The request and response go through additional hops to the proxy server, which can adversely impact performance. You can cache the data returned by the other applications. You can make concurrent asynchronous calls to many applications at the same time.
Best Practices (BP)	• Provide adequate input validation to protect from security vulnerabilities like cross site scripting (XSS). Use vulnerability checking tools like Skipfish. • Perform cross browser compatibility test to ensure that it works across different browsers and operating systems. Some mashups may load more slowly on some browsers. • Take notice of the terms of use and legality. Each API typically carries terms of use that specify who can use the content and how it can be used.	

Client side Vs Server side mashups

#4. Make a decision on technology stack and frameworks to be used

— AngularJS for web tier and Spring/Hibernate for the service and data tiers.

— Git for source control & Jenkins for continuous integration.

— Eclipse or InteliJ IDE for development.

— unit testing, integration testing, and performance testing frameworks & tools. Java/JEE testing frameworks in detail with examples & tutorials

— JBoss application server to run the web services

— and so on list of popular Java/JEE frameworks & tools to jog your memory.

Build a **vertical slice** for a typical use case as a proof of concept for the baseline architecture. Revise and improve on your design in the successive iterations. Here is a typical vertical slice of JEE representing the tiers.

#5. Infrastructure & Capacity planning

— **Infrastructure planning:** hosts, servers, operating system, application/web servers, firewall rules, inter zone connectivity, etc

— **Capacity planning:** physical memory, hard disk space, CPU cores, JVM heap sizes, etc

#6. Logical & physical modelling

— Identify the data requirements, and come up with logical and physical ER (**Entity-Relationship**) diagrams. ERD basics interview Q&A

— **UML diagrams** like class, state chart, sequence, deployment, etc. UML Diagrams Basics Interview Q&As

Note: If you are designing a low latency application like a trading system, then read up on

- Writing low latency applications in Java Interview Q&A
- 13 Tips to write low latency applications in Java
- Java GC tuning for low latency applications
- Capture throughput & latencies with "Metrics Core" tutorial

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

398 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**

‹ Two Pointers: Partitioning around a pivotal number

Core Java Modifiers, Generics, and Annotations: differences between X and Y interview Q&A ›

Posted in Java Application Architecture Interview Essentials, 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

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.