**Java Architect interview questions**

**Database systems**

 What are heuristic exceptions? What can be done when a heuristic exception happens? (People are so often talking about heuristics that I really wonder if they know what they are talking about)

 What does ACID mean? (I know everybody knows the simple answer, but do some research. A and D mean nearly the same, and the C does not make much sense...)

 What does BASE mean? (This becomes more and more important in the area of distributed computing)

 What are isolation levels? (Standard question, but one should know the answer)

 What kind of caches are there in Hibernate ORM (or similiar products)? 

**Modelling and Coding**

* What are anemic domain models? (People always talk about OOP, but then they use this kind of models for Java server applications all the time)
* What are the SOLID principles? (A must known...)
* What code generation tools do you know? Which ones did you use? (If the interviewee does not like or use code generation tools, then he should at least explain, why he doesn't like them)
* Which ways of creating your own DSLs doy ou know? Which ones did you use? (Would be nice if the interviewee would mention other languages like Groovy or Scala and their capabilities of creating DSLs)
* Which UML diagram types do you know? What are UML stereotyypes? What are UML color standards? (Colors: check Peter Coad)

**Java VM**

* What kind of garbage collectors are there in Java 6? What are the differences? Which ones are usually used for Java EE applications? How can you find out which garbage collector should be used for a Java EE application? (I think an architect is often concerned with performance and scalability issues, so some know-how about GC might be helpful)
* What is the generational hypothesis concerning garbage collection?
* What kind of locks in the Java VM do you know? (Or more general, what is the difference between a mutex and a semaphore and similiar questions.)
* What are the most important languages running on a Java VM? Which ones did you use? What are their advantages and disadvantages? (The selection of the right language for the right task is important)
* What monitoring tools do you know? Which ones did you use in production?

**Java EE**

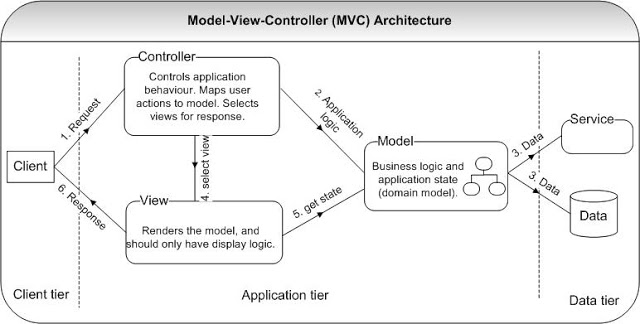
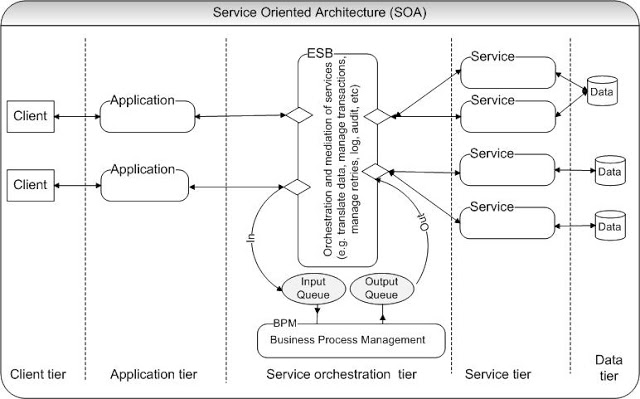
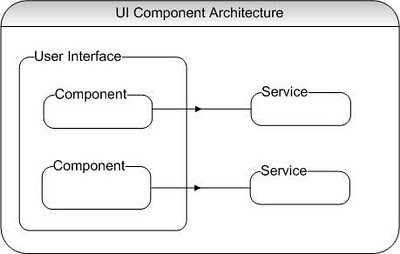
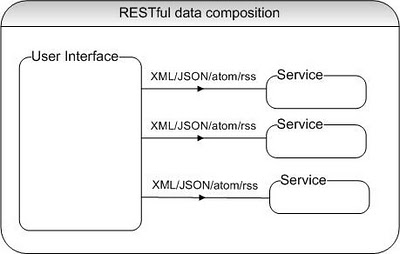
* What are the most important technologies / specifications in Java EE 6?
* What is Weld?
* Which Java web frameworks do you know? What are their advantages and disadvantages? Which ones did you use? (The most important or interesting ones should be mentioned)
* Which ways of making asynchronous calls within a Java EE 6 server do you know? (E.g. there is a new annotation in EJB 3.1 for this issue)
* Which ways of asynchronous communication between web browsers and web servers do you know? (Comet, web sockets, and so on)
* What are the most important performance issues in Java web applications? (Check the Dynatrace blog)
* What are the most important performance issues in Java EE server applications? (Check the Dynatrace blog)

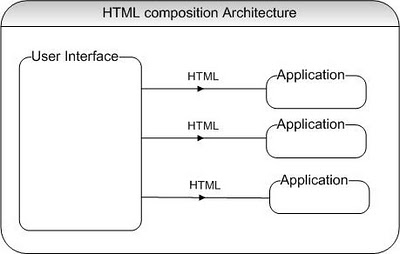
**Distributed Computing**

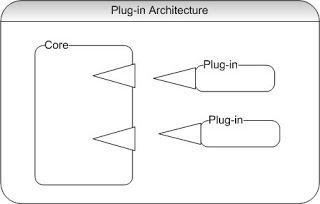
* What are the advantages and disadvantages of ESBs? Which ESBs do you know? Which ones did you use? (An ESB might be one of the important building blocks on the server side)
* What is the CAP theorem? (Now that's quite important)
* What is Terracotta used for? (I mention this product, because it is probably the most important one in this area)
* What is the tocken bucket algorithm? (An architect should know a few techniques for controlling network communication)
* What is REST used for? What are the advantages and disadvantages?
* What is Amdahls Law? What is Gustafson's Law?

<http://java.dzone.com/articles/technical-job-interview>

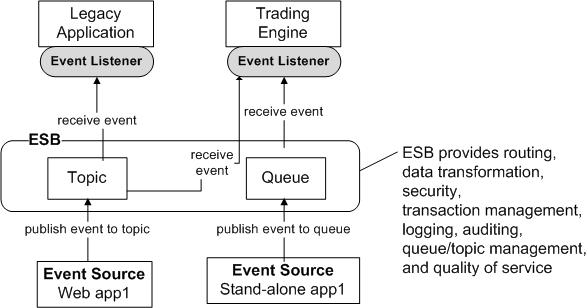
|  |
| --- |
| <http://java-success.blogspot.in/2011/10/java-interview-questions-and-answers-on.html>  Good caliber candidates have the ability to look at the big picture and drill down into details. The line between software development and software architecture is a tricky one. Regardless of you are an architect, developer, or both, one needs to have a good understanding of the overall software architecture. The following Java interview questions are very popular with the interviewers and it can significantly influence the decision  of Hire/No Hire. So, it really pays to have a good overview of various possible architectures. The questions shown below will also make a good platform for further questions depending on your answers.   Be prepared for a **white board session** on architectures, especially the bird's eye view of the lastapplication you had worked on. There will be lots of follow on questions like why a particular approach was used?, what are the benefits and drawbacks of a particular approach?, etc. |

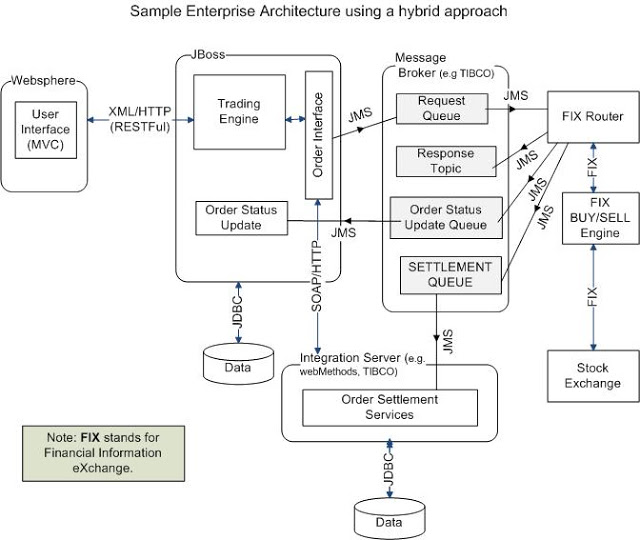
**Q.** Can you draw me a 1000 foot view of the architecture of the system you were/are involved in, in your current/last position?  
**Q.** Can you describe the architecture of a medium-to-large scale system that you actually designed or implemented?  
**Q.** Can you white board the components of the system you recently worked on?  
**Q.** How would you go about designing a JEE shopping cart application?  
**Q.**Can you discuss some of the high level architectures you are experienced with?  
  
**A.** There are a number of high level conceptual architectures as discussed below. These individual architectures can be mixed and matched to produce hybrid architectures.  
  
**Model-View-Controller  Architecture**  
  
Most web and stand-alone GUI applications follow this pattern. For example, Struts and Spring MVC frameworks and Swing GUI.  
  
[](http://3.bp.blogspot.com/-5xMabjujz90/TpfWzR8riUI/AAAAAAAAAHo/Q_6AbHukNc0/s1600/architecture-mvc.JPG)  
  
The model represents the core business logic and state. The view renders the content of the model state by adding display logic. The controller translates the interaction with the view into action to be performed by the model. The actions performed by a model include executing the business logic  and changing the state of the model. Based on the user interactions, the controller selects an appropriate view to render. The controller decouples the model from the view.   
  
  
  
**Service Oriented Architecture (SOA)**  
  
The business logic and application state are exposed as reusable services. An Enterprise Service Bus (ESB) is used as an orchestration and mediation layer to decouple the applications from the services.    
  
[](http://4.bp.blogspot.com/-itE-vaYFq2k/TpfXA7pjE5I/AAAAAAAAAHw/WyVIDkSSOcg/s1600/architecture-SOA.JPG)  
  
  
The above architecture has 5 tiers. The application tier could be using a typical MVC architecture. The service orchestration tier could be using ESB products like Oracle Service Bus, TIBCO, etc and BPM products like Lombardi BPM, Pega BPM, etc. In the above diagram, the ESB integrates with the BPM via messaging queues. The service tier consists of individual services that can be accessed through SOAP or RESTful web services. The SOA implementation requires change agents to drive adoption of new approaches. The BPM, application integration, and real-time information all contribute to dynamically changing how business users do their jobs. So, it needs full support from the business, requiring  restructuring and also it can take some time to realize the benefits of SOA. Cloud computing is at the leading edge of its hype and as a concept compliments SOA as an architectural style. Cloud computing is expected to provide a computing capability that can scale up (to massive proportions) or scale down dynamically based on demand. This implies a very large pool of computing resources either be within the enterprise intranet or on the Internet (i.e on the cloud).   
  
  
  
**User Interface (UI) Component Architecture**  
  
This architecture is driven by a user interface that is made up of a number of discrete components. Each component calls a service that encapsulates business logic and hides lower level details. Components can be combined to form new composite components allowing richer functionality. These components can also be shared across a number of applications. For example, JavaScript widgets, Java Server Faces (JSF) components, etc.   
  
[](http://2.bp.blogspot.com/-wwsi-ruhXBw/TpfXUXZoOxI/AAAAAAAAAH4/bP1crmprDp4/s1600/architecture-ui-component.JPG)  
  
  
  
**RESTful data composition Architecture**  
  
[](http://2.bp.blogspot.com/-ZpQcdSyIHms/TpfXkxcILNI/AAAAAAAAAIA/dkiTkzctO3I/s1600/architecture-restful-composition.JPG)  
  
The user interface can be built by calling a number of underlying services that are each responsible for building part of a page. The user interface translates and combine the data in different formats like XML(translate to HTML using XSLT), JSON (Java Script Object Notation), ATOM (feed for mail messages and calendar applications), RSS (for generating RSS feeds), etc. 

**HTML composition Architecture**  
  
[](http://1.bp.blogspot.com/-qNfK1SFf40s/TpfX-B6pikI/AAAAAAAAAII/y4zZDYixHfs/s1600/architecture-html-composition.JPG)  
  
In this architecture, multiple applications output fragments of HTML that are combined to generate the final user interface. For example, Java portlets used inside a portal application server to aggregate individual content..  
  
  
**Plug-in Architecture**

[](http://3.bp.blogspot.com/-qVn3pyioemM/TpfYKnHavzI/AAAAAAAAAIQ/nRlm0tf8Izc/s1600/architecture-plugin.JPG)

In this architecture, a core application defines an interface, and the functionality will be implemented as a set of plug-ins that conform to that interface. For example, the the Eclipse RCP framework, Maven build tool, etc use this architecture. 

**Event Driven Architecture (EDA)**  
  
[](http://2.bp.blogspot.com/-03TVBd4PM-M/TpfYSJcpG7I/AAAAAAAAAIY/NhRDAd9LFVw/s1600/architecture-event-driven.JPG)  
  
  
The EDA pattern decouples the interactions between the event publishers and the event consumers. Many to many communications are achieved via a topic, where one specific event can be consumed by many subscribers. The EDA also supports asynchronous operations and acknowledgments through event messaging. This architecture requires effective monitoring in place to track queue depth, exceptions, and other possible problems. The traceability, isolation, and debugging of an event can be difficult in some cases. This architecture is useful in scenarios where the business process is inherently asynchronous, multiple consumers are interested in an event(e.g. order status has changed to partially-filled ), no immediate acknowledgment is required (e.g. an email is sent with the booking details and itinerary), and real-time request/response is not required (e.g. a long running report can be generated asynchronously and made available later via online or via email).   
  
Most conceptual architectures use a**hybrid approach**using a combination of different architectures based on the benefits of each approach and its pertinence to your situation. Here is a sample hybrid approach depicting an online trading system.

[[](http://2.bp.blogspot.com/-ohI4Q4tKmJE/Tp-b3KnusNI/AAAAAAAAAI4/FcBslEsuiSI/s1600/architecture-hybrid.JPG)](http://2.bp.blogspot.com/-ohI4Q4tKmJE/Tp-b3KnusNI/AAAAAAAAAI4/FcBslEsuiSI/s1600/architecture-hybrid.JPG)

FIX is a **F**inancial **I**nformation e**X**change protocol. You could also notice a number of synchronous calls using XML/HTTP or SOAP/HTTP and asynchronous calls using JMS. The above diagram also depicts that an enterprise architecture can be complex with a number of moving parts. So, it is imperative that all these moving parts are properly monitored and tested for any potential performance issues. Most of these services will be running as a cluster or a load balanced service with either active/active or active.passive configuration for high availability and scalability.

**Other links:**

<http://ericbweimer.blogspot.in/2012/10/java-architect-interview-questions.html>

<http://www.javagyan.com/preparing-for-an-interview/java-j2ee-interview-questions>