**Duties & Responsibilities:**

•             Helping and guiding development teams in their migration for the building application using msjava ant build, web-services using CXF, Database Pool using msjava API’s, Migration from Websphere to Tomcat & RAD to Eclipse.

•             Participate in Architectural design reviews before providing solution to the team.

•             Providing ‘Hello World’ kind of examples to the Dev team for new challenges and issues.

•             Creating POC’s (Proof of Concept) for new technical challenges faced.

•             Interacting directly with Dev teams and providing solution with their ESA migration phase.

•             Creating cook book recipes or wiki with new learning’s or challenges faced during the ESA migration phase.

•             Sometime need to run and setup entire project on your desktop and refactor of code to change library usage, implement best practices etc.

•             Partial understanding of application code through code compilation, code walkthrough, execution flow, overall design.

**Basic Technical Requirements**

•             Must understand WAS configuration, connection pooling, session persistence, Web Services, scalability

•             Must have done large scale refactoring of code to change library usage, implement best practices etc.

•             Good knowledge of data structures

•             Detail knowledge of web services development using Java (using axis, cxf etc.)

•             Equally conversant on Linux and Windows platforms

•             Must be hands-on

•             Understanding of Ant build system

•             Some knowledge of Spring framework

•             Experience in Tomcat preferred

•             Coordination and communication skills a plus

•             Knowledge of Kerberos, SSL, Site-minder.

1. **Msjava – discontinued Microsoft JVM.**
2. **Site minder : SiteMinder** is a centralized Web access management system that enables user authentication and single sign-on, policy-based authorization, identity federation, and auditing of access to Web applications and portals.

[What are the differences when deploying on Tomcat vs. Websphere?](http://stackoverflow.com/questions/7520293/what-are-the-differences-when-deploying-on-tomcat-vs-websphere)

If I were to deploy an application on Tomcat vs. Websphere, what are things that I need to consider?

Do I have to develop my Java code differently if developing in one app server vs another?

Edit:

I will be funneling people from a website into a web app that does credit card processing and e-signatures (cc processing and e-sigs are through separate services). That is its sole job

You cannot use EJBs on Tomcat (unless you add OpenEJB). If your WebSphere deployment uses EJBs, you'll have to remove them to deploy on Tomcat.

If you use any Java EE features beyond servlet/JSP engine and JNDI naming service you'll have to eliminate them from your app.

Tomcat accepts WAR packages. If you package your app into an EAR on WebSphere, you'll have to change it to WAR for Tomcat.

Both use JNDI for data sources. There might be some nagging differences in naming conventions, but if you stick to the standard they should be portable.

If you use any WebSphere specific code in your app, you'll have to remove it to deploy on Tomcat.

If your app is servlets, JSPs, and JDBC you can deploy on either one without any problems.

Tomcat is a Servlet Container, whereas WebSphere is an Application Server. Which is "best" depends entirely on what type of container your project needs.

Websphere is a full J2ee Application server   
It means all components of J2ee are available in Websphere ,  
for instance : EJB stuff  
Tomcat is a Servlet container it means that no ejb stuff will operate insde tomcat.  
  
You can use Websphere when you need a reliable high performance full J2ee application server .  
but Tomcat is for place where you have no EJB stuff.  
Websphere could well integrate with other IBM techs like tivolli , access manager , its EGL ,...

Application server can execute Enterprise application, i,e (servlets, jsps, and EJBs) it is having two containers 1. Web Container(for interpreting/executing servlets and jsps) 2. EJB container(for executing EJBs). it can perform operations like load balancing , transaction demarcation etc etc

## Application Server vs Web Server

1**. Application Server**supports **distributed transaction and EJB**. While Web Server only supports Servlets and JSP.

2. Application Server can contain web server in them. most of App server e.g. JBoss or WAS has Servlet and JSP container.

3. Though its not limited to Application Server but they used to provide services like **Connection pooling**, **Transaction management**, messaging, clustering, load balancing and persistence. Now Apache tomcat also provides connection pooling.

4. In terms of l*ogical difference between web server and application server*. web server is supposed to provide http protocol level service while application server provides support to web service and expose business level service e.g. EJB.

5. Application server are more heavy than web server in terms of resource utilization.

Personally I don't like to ask questions like ***Difference between Application Server and Web Server***. But since its been asked in many companies, you got to be familiar with some differences. Some times different interviewer expect different answer but I guess on Java's perspective until you are sure when do you need an application server and when you need a web server, you are good to go.

Read more: <http://javarevisited.blogspot.com/2012/05/5-difference-between-application-server.html#ixzz3PJksPDFC>

