

Would you, at present date, use JBoss or Glassfish (or another) as Java EE server for a new project?

[closed]

Asked 16 years, 1 month ago Modified 11 years, 7 months ago

Viewed 63k times



136



As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, [visit the help center](#) for guidance.

Closed 12 years ago.

If you started a new Java EE project today which is to be finished in about a year, which application server would you choose and why?

Part of your answer should include your arguments for your decision. And also how much experience you have with the Java EE server you choose and with the other available servers on the market. These are interesting as we all get a sense of the investigation and thought that was put into your answer.

[java](#)[jboss](#)[jakarta-ee](#)[glassfish](#)[Share](#)[edited Nov 11, 2008 at 11:16](#)[Improve this question](#)[Follow](#)[community wiki](#)[3 revs, 2 users 83%](#)[user14070](#)**9 Answers**

Sorted by:

Highest score (default)

**180**

I have used WebLogic, WebSphere, JBoss, GlassFish, Resin, Jetty, Tomcat, and a few others over the last 10+ years. So, if I were considering a new project, I would ask myself a few questions first. One thing that I would not question anymore is that I would flat refuse to use JSPs unless I was tortured until I cried for my mommy.

Do I have to be compatible/deploy to a specific product because of someone's mandate? Is there no way to ignore them or convince them otherwise? If so, there's your answer.

Do I have to use EJBs? Really? Avoid them if at all possible--they are really only needed for very large, enterprise-class systems. Remember that they are merely tools, and big ones at that (can anyone say "Golden Sledgehammer"?). They are heavily overused,

so really, really question whether you need them. If you do need them, then that removes several of your options including my favorite, Jetty.

Do you have to use any of the other major J2EE technologies like JMS, ESB, etc? If so, and you really can't do without, then you are again constrained to a full-blown J2EE container. Carefully think and investigate before you commit to BPM, for example, and avoid AquaLogic BPM at (almost) all costs--it is ugly in the extreme.

If you really must use a full-blown J2EE container, consider open-source first because it is more robust, better supported, and more cost-effective. They have larger customer bases and more open support interaction, so they tend to get better fixes faster. However, Resin is immature and I would avoid it relative to GlassFish or JBoss--I found it problematic to deploy and support. I would prefer JBoss because of its wider customer base, maturity, etc. GlassFish is harder to incorporate into an automated build/deployment process, but it might be nicer for some of its specific features (if you need them).

Do I have a special reason to need Apache? Then lean towards Tomcat, perhaps plus something.

Can I make do with just servlets? Then I would use Jetty--it is the lightest, fastest, easiest, most flexible solution. If I am leaning against being able to use Jetty, I would question all my assumptions of why. YAGNI applies.

Best is to use `StringTemplate/WebStringTemplate` on Jetty: a clean, robust, fast, maintainable solution with no licensing fees, solid reputation and support, etc. That is where I start nowadays.

Most applications/systems choose lots of fancy J2EE features when all they really need is servlets and JDBC with some decent architecture/design. Question why you think you need more.

Of the full-blown containers, I would avoid WebLogic and WebSphere unless you are supporting a MAJOR public website (my current employer's website is deployed on WebLogic and it gets eleven+ million hits per month, others have been comparable). WebLogic's real claim-to-fame is their relatively easy clustering, but avoid their proprietary vendor-lock-in features at (almost) all cost. WebSphere is simply a nightmare that I would avoid literally at all cost--I refuse to do projects involving WebSphere after having done a couple in the past. Neither product is worth the massive licensing fees, unless you truly have a special need that drives the use of a proprietary feature. In a decade as a senior architect/engineer for lots of Fortune 500 companies, I have yet to see such a need. On the other hand, I have seen LOTS of pain due to picking such proprietary products.

Even for the really large, high traffic, public websites, the proprietary products are still questionable. I would rather spend that multi-million dollars per year of licensing fees

on some good hardware and some quality time from a handful of really good consultants to address a simple scalability solution. The extra millions per year could then be used to produce something worthy of selling on that nice website...

EDIT: another piece to consider...

I have recently encountered [Terracotta](#). I am rethinking everything, and looking to deploy it in a significant system soon. In particular, Terracotta does clustering better than anything else, so I would NO LONGER recommend WebLogic for its clustering.

Share Improve this answer

edited Feb 2, 2009 at 5:10

Follow

community wiki

3 revs

[Rob Williams](#)

7 For future reference, you can usually find the definitions for acronyms via a Google or Wikipedia search. YAGNI = You Aren't Going to Need It = don't overdo your design JMS = Java Message Service ESB = Enterprise Service Bus BPM = Business Process Management – [Rob Williams](#) Nov 17, 2008 at 19:37

21 Your comments about Java EE and EJB's are a little outdated. J2EE?! That was like 5 years ago. Take a look at Java EE 6, and modernize your perspective! – [Brian Leathem](#) Jan 18, 2011 at 3:43

6 @Brian: I agree with Brian, especially with EJBLite, it has become much lighter weight. – [Thang Pham](#) Mar 23, 2011 at 21:44

7 @Brian, look at the post - it was written three years before your comment. And I'd still say that Spring is lighter than even a slimmed down Java EE. – [duffy](#) Jun 29, 2012 at 23:13

2 What is the verdict now in 2012? Does JBoss 7 AS come king over Glassfish in the Java 6 EE realm? Or the other way around? – [Rolando](#) Jul 19, 2012 at 2:16 ✎



10



The term "application server" is ambiguous. With GlassFish v3, you can start small with, say, a traditional web container and evolve (using OSGi and simple "add container" functionality) to add anything you'd like : JPA, JAX-RS, EJB's, JTA, JMS, ESB, etc... Yet it's the same product, same admin interface, etc. Does this qualify as an application server to you? -Alexis (Sun)

Share Improve this answer

answered [Jun 29, 2009 at 10:53](#)

Follow

community wiki
[Alexis MP](#)

1 Unfortunately Glassfish is not an official product anymore, but "only" the reference implementation.
– [Thorbjørn Ravn Andersen](#) Jun 13, 2014 at 21:12



9

The first question I usually ask myself is "Can I do this with Tomcat?". If the answer is no because I need JMS or JTA then I resort to an application server.



I used WebLogic 8 about 3 years ago happy with WebLogic's ease of use and the licensing/cost model. We used it for two projects one was a web service and the other was a portal. We did not encounter any problems with WebLogic or WebLogic Portal in either of those projects.



For the last two years I was working with WebSphere. Any time I negotiated with IBM it was always ended up costing twice as much as a WebLogic equivalent but corporate policy dictated WebSphere had to be used. I found the learning curve on WebSphere to be considerably steeper than WebLogic and our build/deploy/test life-cycle was so time consuming that we used Tomcat in the development environment. But the the biggest issue I had with WebSphere was when we encountered a bug that forced us to upgrade to the next patch release only to run into new problem parsing web.xml. It took a 48hr shift to work through all that.

At the moment though I am using JBoss. About 3 months ago I was about to embark on my new project with Tomcat and Jetspeed 2, But I noticed that Jetspeed 2 seems a bit stagnant right now and JBoss Portal 2.7.0 was just released with JSR 286/Portlet 2.0 support. I gave JBoss a spin and found it very easy to set-up and administer. The build/deploy/test cycle is very quick and I

rarely have to restart the server unless I have changed a Spring XML file somewhere.

Share Improve this answer

edited Dec 24, 2008 at 17:47

Follow

community wiki

[2 revs](#)

[bmatthews68](#)

Nice answer! Have you tried Jetty? And what's your opinion on it in case you have? – user14070 Jan 15, 2009 at 0:35



I've been using jBoss for 3-4 years.

7

Arguments for jBoss:



1. Open source.
2. Commercial support available.
3. Large, active user community.



Arguments against jBoss:

1. No general-access, supported Java EE 5 container release.
2. Lots of documentation but verbose; can be hard to find the answers to "How do I do x?"
3. Administrative tools for 4.x poor compared to other commercial offerings.

Share Improve this answer

edited May 2, 2013 at 6:59

Follow

community wiki

2 revs, 2 users 96%

johnstok

"No general-access, supported JEE 5 container release." I guess that is no longer the case, correct? – [Raedwald](#) Mar 28, 2011 at 13:39

@Raedwald: yes, now that JEE 6 has been around for some time ;-) – [ymajoros](#) Sep 6, 2011 at 18:43



4

Checkout GlassFish 3.1! Built on top of the modular, Java EE 6 based GlassFish v3 kernel, version 3.1 delivers clustering, centralized administration and high availability.



Refer to http://blogs.oracle.com/nazrul/entry/glassfish_3_1 for more details.



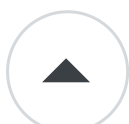
Share Improve this answer

answered May 8, 2011 at 3:49

Follow

community wiki

[Nazrul](#)



Another point that was not discussed here is performance. If this is a concern because of the type of

3

service or because of the number of users, then the following will be applicable:



- Tomcat seems to be slower than Glassfish
- Glassfish seems to be slower than Resin
- Resin is much slower than G-WAN + Java



Note that G-WAN relies on the JVM alone: it does not use any further containers (unless specified explicitly) so you might reserve it to performance-critical portions of your web applications.

As G-WAN supports other languages (C, C++, C#, D, Objective-C), you may even process some parts of the applications in raw C while keeping Java for other tasks.

Share Improve this answer

answered [Oct 4, 2012 at 8:39](#)

Follow

community wiki
[gert](#)



2



I might include your preferred OS as a decision criteria. It should make it easier to support if you are using the same vendor for OS and app server. If you already have a relationship with one or both vendors consider if they are good to deal with.



From a technical perspective I would choose GlassFish because it has support for more recent innovations. I do



not think JBoss is bad in anyway it simply isn't as up-to-date.

Most of my experience is on WebLogic but I have used JBoss and GlassFish. I just released a new site on a complete Sun open source stack (OpenSolaris, GlassFish, MySQL) and it was a great experience with only minor frustrations.

Share Improve this answer

answered [Nov 10, 2008 at 18:32](#)

Follow

community wiki
[Ed.T](#)

OS isn't really an issue, unless you have very specific binary dependencies (which shouldn't be the case for most java projects). We develop on windows, 32 and 64 bits, and deploy on Glassfish on Solaris. Most developers don't really know and don't have to care. Users don't see it (most of our developments being web applications). – [ymajoros](#) Sep 6, 2011 at 18:47



2

I still think that WebLogic is the best Java EE app server on the market. I think it's worth it if you can afford those license fees.



I've been surprised to see how far you can go by combining Tomcat, OpenEJB, and ActiveMQ. That would seem to me to be a low-cost alternative.





I'd also look into the Spring dm Server. It's based on Tomcat, but I think the OSGi piece they've added in could be everywhere in short order. If it's done with the same quality as the Spring framework, it'll be very good indeed.

Share Improve this answer


answered [Dec 24, 2008 at 18:16](#)

Follow

community wiki
[duffymo](#)

2 Problem I have with WebLogic is the vendor lock in, that's a nasty pill to swallow when you really don't need to! – [Manius](#) Oct 18, 2010 at 3:32

1 That's true of all Java EE vendors that I know of, not just WebLogic. If you use any vendor-specific features you're locked in. Gotta write code sometime. – [duffyymo](#) Oct 18, 2010 at 9:30

3 WebLogic is commercial-only--that's what I'm getting at - once you write a big check, you're "locked" to a larger degree than you are with an open source alternative. Obviously if you care about platform independence, you wouldn't use vendor specific features, so that's not what I'm referring to. Actually a survey I read once said devs believe that vendor-lock in avoidance is the #1 advantage of open source (not cost). – [Manius](#) Apr 24, 2011 at 19:25 

Complete nonsense? Do you believe that signing a multi-million dollar contract with a vendor *doesn't* lock you in? There's your proof. – [duffyymo](#) Jun 29, 2012 at 22:57

@ymajoros Did you mean "shouldn't" have vendor lock-in? Frankly, I can't make sense of your comment. – [Patrick M](#) Jul 6, 2012 at 18:12



An alternative: use no appserver at all.

1

Check out

<http://www.atomikos.com/Publications/J2eeWithoutApplicationServer>.



For web projects, keep a light web container if you have to, combined with something like Wicket to avoid the complexity of JSP/JSF or struts.



HTH Guy


Share Improve this answer

answered [Jun 2, 2009 at 20:25](#)

Follow

community wiki

[Guy Pardon](#)

If you don't want to learn using tools, don't use any. Or try to become a skilled professional and invest in your environment, you'll be rewarded. Must admit I did that for some projects. The same projects evolved no appserver, to a custom client-server in Spring, to pure Java EE and Glassfish. Wouldn't ever want to go back, the first solution was actually way too complex, it's as simple as can be today (and standard, would deploy on any Java EE app server without much change). – [ymajoros](#) Sep 6, 2011 at 18:49 

good answer, bad way to get the document – [msangel](#) Jan 3, 2015 at 1:59
