

Web Services vs EJB vs RMI, advantages and disadvantages?

Asked 14 years, 11 months ago Modified 9 years ago Viewed 73k times



59

My web server would be overloaded quickly if all the work were done there. I'm going to stand up a second server behind it, to process data.



What's the advantage of EJB over RMI, or vice versa?



What about web services (SOAP, REST)?



java

web-services

ejb

distributed

rmi

Share

asked Jan 6, 2010 at 15:03

Improve this question

Follow

community wiki

[Dean J](#)

3 Answers

Sorted by:

Highest score (default)



124

EJBs are built on top of RMI. Both imply Java clients and beans. If your clients need to be written in something else (e.g., .NET, PHP, etc.) go with web services or something



else that speaks a platform-agnostic wire protocol, like HTTP or XML over HTTP or SOAP.



+100



If you choose RMI, you don't need a Java EE EJB app server. You have to keep client and server JVMs in synch; you can't upgrade the client without upgrading the server. You have to write all the services that the EJB app server provides for you (e.g., connection pooling, naming and directory services, pooling, request queuing, transactions, etc.).

RMI is quite low level when you think about it. Why would you drop all the way back to CORBA?

A better choice is EJB 3.0 versus Spring. It depends on whether you like POJO development, want a choice of relational technologies besides ORM and JPA, among other things.

You can pay for a Java EE app server (e.g., WebLogic, WebSphere) or use an open source one (JBoss, Glassfish and OpenEJB and ActiveMQ), or you can stick to Spring and deploy on Tomcat, Jetty, Resin or any other servlet/JSP engine.

Spring offers a lot of choice by being technology agnostic: persistence (Hibernate, iBatis, JDBC, JDO, JPA, TopLink), remoting (HTTP, Hessian, Burlap, RMI, SOAP web service), etc.

EJB 3.0 is a spec with many vendors; Spring can only be had from Spring Source.

I would recommend [Spring](#). It's very solid, has lots of traction, isn't going anywhere. It leaves all your options open.

Web services are great in theory, but there are some gotchas that you need to watch out for:

1. Latency. Fowler's first law of distributed objects: "Don't!" An architecture consisting of lots of fine-grained distributed SOAP services will be elegant, beautiful, and slow as molasses. Think carefully before distributing.
2. Marshalling from XML to objects and back consumes CPU cycles that aren't providing any business value besides allowing your clients to speak a platform-agnostic protocol.
3. SOAP is a standard that is becoming more bloated and complex every day, but it has lots of tool support. Vendors like it because it helps drive sales of ESBs. REST is simple but not as well understood. It's not supported by tools.

Spring's web service module is very good, but do be careful about choosing to deploy this way. Write in terms of POJO service interfaces. These will allow you to get the conceptual isolation you want, defer the deployment choice until the last moment, and let you change your mind if the first thought doesn't perform well.

Share Improve this answer

Follow

community wiki
3 revs, 2 users 98%
duffyymo

Spring Web Services? Spring is a big word to search on. :-)

– Dean J Feb 19, 2010 at 15:40



9

Between EJB and RMI, EJB would certainly be better - it has everything RMI has and much more via the container (object pooling, transaction management, etc.)



Between EJB and web services, web services would give you more portability if you want to be able to call them from non-java apps in the future. EJB again gives you things like transaction management and pooling that you might not get "out of the box" with web services.




Personally, if I were doing it, I would probably use EJB or some similar remote object framework (spring remoting comes to mind as well). If you need the ability to call the objects from a non-java app, you can always front your EJBs with simple web service proxies as needed.

community wiki

Eric Petroelje

1 Could you quickly compare Spring Remoting to EJB? I don't need the ability to work with non-Java apps, but have found EJB's unwieldy in the past, whereas web services feel more straightforward and simpler to write/maintain. – Dean J Jan 6, 2010 at 15:18

2 @Dean J - EJBs were pretty complicated in the older versions of J2EE, but have been greatly simplified in 3.0. I haven't used spring remoting much, but here's an article that compares the two a bit:
onjava.com/pub/a/onjava/2005/06/29/spring-ejb3.html?page=1 – Eric Petroelje Jan 6, 2010 at 15:27 

I'll give a look at that article, and re-evaluate EJB3; EJB2 just felt ugly. – Dean J Jan 6, 2010 at 15:28

1 @Dean J - Spring Remoting and EJB's do not have to be separate choices. Spring simply provides an abstraction layer for remote service calls in which the actual remoting protocol is hidden. We use Spring Remoting over HTTP, EJB and JMS for the same service interfaces in a current system. The protocol just changes depending on the location of the caller (Http for client app, EJB for another service within a container, JMS from a trusted non JEE server). – Robin Jan 6, 2010 at 15:57



Re: web services (SOAP, REST) If your back end servers are not going to be exposed publicly, then you are not

3

getting any benefit from using platform independent web service interfaces such as SOAP/REST.



In fact you'll be incurring a penalty with all of the overhead added by the XML tags wrapping the data across a remote call, not to mention the hit you'll take from marshalling and unmarshalling the XML to java objects.

Although any distributed call is going to require some level of serialization - even RMI/EJB, but the price is greater when serializing to human readable XML.

You may not need to code remote calls in java at all, you could front your service with a plain apache httpd instance, which is configured to load balance across multiple java servers using [mod_jk](#) or [mod_proxy](#).

These modules can be used to load balance across servlet containers such as tomcat/jetty, or ejb containers such as jboss/glassfish.

Share Improve this answer

answered Feb 19, 2010 at 7:25

Follow

community wiki
[crowne](#)

2 In fact with a Node.js server and a JSON REST API I'm 100% sure the seriaziation and deserialization of Javascript objects will blow any Java thing. – [bluehallu](#) Jun 6, 2014 at 12:51
