OSGi, Java Modularity and Jigsaw

Asked 13 years, 3 months ago Modified 10 years, 5 months ago Viewed 10k times

94

So as of yesterday morning I hadn't a clue as to what OSGi even was. *OSGi* was just some buzzword that I kept seeing cropping up over and over again, and so I finally set aside some time to brush up on it.

It actually seems like pretty cool stuff, so I'd like to start off by stating (for the record) that I'm not anti-OSGi in any respect, nor is this is some "OSGi-bashing" question.

At the end of the day, it seems that OSGi has - essentially - addressed JSR 277 on Java Modularity, which recognized that there are shortcomings with the `JAR` file specification that can lead to namespace resolution and classloading issues in certain corner cases. OSGi also does a lot of other really cool stuff, but from what I can ascertain, that's its biggest draw (or one of them).

To me - as a fairly new (a few years now) Java EE developer, it is absolutely mind-boggling that we are in the year 2011 and currently living in the era of Java 7, and that these classloading issues are still present; particularly in enterprise environments where one app server could have hundreds of JARs on it, with many of them depending on different versions of one another and all running (more or less) concurrently.

My question:

As interested as I am in OSGi, and as much as I want to start learning about it to see where/if it could be of use to my projects, I just don't have the time to sit down and learn something that large, at least now.

So what are non-OSGi developers to do when these problems arise? What **Java** (Oracle/Sun/JCP) solutions currently exist, if any? Why was Jigsaw cut from J7? How sure is the community that Jigsaw will get implemented next year in J8? Is it possible to get Jigsaw for your project even though its not a part of the Java platform yet?

I guess what I'm asking here is a combination of panic, intrigue and a facepalm. Now that I finally understand what OSGi is, I just don't "get" how something like Jigsaw has taken 20+ years to come to fruition, and then how that could have been canned from a release. **It just seems fundamental.**

And, as a developer, I am also curious as to what my solutions are, sans OSGi.

Also, **Note**: I know this isn't a "*pure programming*"-type question, but before some of you get your noses bent out of shape, I wanted to state (again, for the record) that I

deliberately put this question on SO. That's because I have nothing but the utmost respect for my fellow SOers and I'm looking for an architectural-level answer from some of the "Gods of IT" that I see lurking around here every day.

But, for those of you who absolutely *insist* that a SO question be backed with some code segment:

```
int x = 9;
```

(Thanks to anybody who can weigh-in on this OSGi/Jigsaw/classloader/namespace/JAR hell stuff!)

java osgi classloader java-platform-module-system

Share
Improve this question
Follow

edited Jul 5, 2014 at 4:06

Charles
51.4k ● 13 ● 106 ● 143

asked Sep 21, 2011 at 10:54

IAmYourFaja
56.8k ● 185 ● 484 ● 776

Well, Java 9 is here as of yesterday, with Jigsaw. Nice to read: The Top 10 Jigsaw and Java 9 Misconceptions Debunked – David Tonhofer Sep 23, 2017 at 15:42

3 Answers

Sorted by: Highest score (default)

100

First understand that Jigsaw's primary use case is to modularise the JRE itself. As a secondary goal it will offer a module system that may be used by other Java libraries and applications.

My position is that *something like* Jigsaw is probably necessary for the JRE only, but that it will create far more problems than it claims to solve if used by other Java libraries or apps.

The JRE is a very difficult and special case. It is over 12 years old and is a frightful mess, riddled with dependency cycles and nonsensical dependencies. At the same time is is used by approximately **9 million** developers and probably **billions** of running systems. Therefore you absolutely cannot refactor the JRE if that refactoring creates breaking changes.

OSGi is a module system that helps you (or even *forces* you to) create software that is modular. You cannot simply sprinkle modularity on top of an existing non-modular codebase. Making a non-modular codebase into a modular one inevitably requires

some refactoring: moving classes into the correct package, replacing direct instantiation with the use of decoupled services, and so on.

This makes it hard to apply OSGi directly to the JRE codebase, yet we still have a requirement to split the JRE into separate pieces or "modules" so that cut-down versions of the JRE can be delivered.

I therefore regard Jigsaw as a kind of "**extreme measure**" to keep the JRE code alive while splitting it up. It does **not** help code to become more modular, and I'm convinced that it will actually increase the maintenance required to evolve any library or application that uses it.

Finally: OSGi exists whereas Jigsaw does not exist yet and may never exist. The OSGi community has 12 years of experience in developing modular applications. If you are seriously interested in developing modular applications, OSGi is the only game in town.

Share

Improve this answer

Follow

edited Nov 3, 2012 at 23:04

answered Sep 21, 2011 at 13:06

Neil Bartlett

23.9k ● 4 ● 47 ● 80

Is this you, too? slideshare.net/mfrancis/… almost the same contents. – Jin Kwon Feb 26, 2013 at 12:21

There is also JBoss Modules: docs.jboss.org/author/display/MODULES/Introduction It is also used by Ceylon (ceylonlang.org). See this thread on the Ceylon user forum: groups.google.com/forum/?hl=de#!topic/ceylon-users/RmDskLDNkug – OlliP Oct 24, 2013 at 16:00

Does the final statement: *"If you are seriously interested in developing modular applications, OSGi is the only game in town"* still hold? – Adam Arold Jul 24, 2015 at 10:54

1 @AdamArold I believe so. Jigsaw still doesn't exist in any released version of Java. JSR 376 (Java Platform Module System) is still forming its expert group and hasn't even started on a first draft yet. Java 9 is not due for more than a year, and even when released is not guaranteed to have modularity (it slipped from Java 7, then from Java 8, and could easily slip again). Finally, the published requirements for JSR376 state that OSGi interop is required... so adopting OSGi remains a safe choice, and today the only practical choice. – Neil Bartlett Jul 26, 2015 at 21:26

2 @AdamArold Okay that's a bit of a different question! You could say that OSGi is a microservices architecture, but I know what you're saying. For me, the idea of modelling every service as a process is a much more complex problem, in terms of management and security and comms overhead. OSGi is simpler and faster. Having said that, it's very easy to use OSGi Remote Services to transition services in and out of the process barrier, so I believe it's a good choice for an implementation technology for microservices. – Neil Bartlett Jul 27, 2015 at 6:43

21

It's simple, if you want to do real component-based development in Java today then OSGi is the only game in town.

In my opinion, Jigsaw is a combination of a compromise of what's doable in the JDK and previous bad relationship between SUN and the OSGi guys. Maybe it will ship with Java 8, but we have to wait and see.

OSGi is no panacea if you are working in a typical enterprise setting and you will need to become familiar with how class loading works as a number of well-known libraries (looking at you, Hibernate) made assumptions about class visibility that are no longer valid inside OSGi.

I like OSGi, but I wouldn't try and retrofit it to an existing system. I'd also weigh the pros and cons in terms of greenfield development - I would recommend looking at the Apache or Eclipse products that simplify life for OSGi and not doing it all yourself.

If you're not doing OSGi, then your out of luck if you've come up with a system that has dependencies on different versions of the same library - all you can do is try and avoid the problem, though needing multiple versions of a library seems like a architecture 'smell' to me.

Share

Improve this answer

Follow

edited Sep 22, 2011 at 7:51

answered Sep 21, 2011 at 11:54

SteveD

5,405 ● 1 ● 25 ● 34

Yeah I figured there was a lot of 'bad blood' between Sun and the OSGi Alliance. I can't imagine Oracle is going to let things slide out of their control, though. You're really telling me that there are no plans to truly remedy (not hack) this JAR hell stuff anytime soon? That blows my mind! – IAmYourFaja Sep 21, 2011 at 11:59

6 @Mara It's not really fair to say there is bad blood. After all, Sun was one of the co-creators of OSGi around 12 years ago (JSR 8). Sun also rejoined the OSGi Alliance as a full member, about a year before the Oracle acquisition. They also developed quite a lot of software on top of OSGi, pre-Oracle. The most obvious example being Glassfish. However it is fair to say that there is friction between certain individuals at Sun/Oracle and OSGi. – Neil Bartlett Sep 21, 2011 at 20:39

14

I love your use of the phrase "corner cases" to describe the current situation.

> there are shortcomings with the JAR file specification that can lead to namespace resolution and classloading issues in certain corner cases

Anyhow, since many years I've been interested in tools and techniques that supports the creation of and, even better, enforce, code that is cleaner, more decoupled, more

coherent and more maintainable than what probably would have been the result without them. Test Driven Design and Junit was such a combination.

After having spent a couple of months moving a substantial part of our code base to OSGi, I would say that OSGi is an even better tool in this regard. And that is really a sufficient reason to move to OSGi. In the long run it will save you a lot of money.

And, as a bonus, it'll give you a chance to do a lot of cool stuff. Imagine, during a demo, seamlessly upgrading the authentication module, without traffic loss, to support OAuth ... it's suddenly a joy again to create stuff!

Share Improve this answer Follow

answered Sep 21, 2011 at 20:27

forty-two

12.8k ● 2 ● 27 ● 39