

Java8: Jigsaw nélkül?

Az elmúlt másfél év során sok információ került ki a Project Jigsaw kapcsán, mint a Java7, majd a Java8 egyik nagy erössége, s a jelen állás szerint csak a Java9 nagy dobása lesz, mivel a Mark Reinhold úgy ítélte a projekt állását, hogy az elkövetkező egy évben (a Java8 majd csak **2013 szén** fog kiadódni) nem lehet tisztességesen befejezni, legalábbis ez olvasható a [blogbejegyzésében](#). , így továbbra se lesz semmi a **JSR 337** kódnéven emlegetett modularitási és profilozási lehetőségekből a következő nagyobb Java kiadásban. Mark megemlítette a fbb dilemmáját is: fejezzék be a modularitást és adják ki később a Java8-at vagy adják ki idben, és a következő kiadásba kerüljön bele egy letisztult és korszerű modularitást támogató megoldás.

A bejelentés viszonylag nagyobb port vert fel a Java közeli blogokon és portálokon, s a vélemények között akad mindkét opcióhoz támogató, a [dzone.com](#) oldalon idéztek kett ilyen véleményt:

We didn't go too far with modularity in Groovy 2... because we didn't want to duplicate what was supposed to be coming in Java 8. But now, it means it won't be in the hands of developers before 2 years, and in production before 3 years :-(It means that Jigsaw, if it ever ships, will have been 5+ years in the making. That sounds a lot, even for such a big feature and task.

Guillaume Laforge (major Groovy contributor)

Illetve:

DON'T HOLD THE TRAIN. The five years from Java 6 (which itself was disappointing) to Java 7 nearly killed the language.

Johannes Brodwall

A nagy kérdés az, hogy a platform kibír-e még 2-3 évet, amíg a Java9 megjelenik, és nem jelenik-e meg a jelenlegi JCP támogatással bíró *de jure* szabvány Jigsaw helyett egy alternatív megoldás, amely *de facto* szabvány lesz. Sokan az OSGi-t emlegetik, illetve a Maven mintáját gondolják átvehetnek...

Saját vélemény

A Jigsaw – mint modularizációs ötlet – alapvetően a JavaFX miatt született 2009-2010 környékén, mivel a kliens oldalon (asztali, hordozható vagy hordható eszközökön) futó beépülő programok csak akkor képesek hatékonyan futni, ha nem kell maguk alá telepíteni egy 50-150MB-ot méretű futtató környezetet, hanem csak a futásukhoz szükséges modulokat töltsék le és indítsák el. Ezen modularizáció nélkül a JavaFX programcskák indulása akár másodpercekig is eltarthat, ha a felhasználó sikeresen feltelepítette a JRE-t, amelyhez rendszergazdai jog kell... nyilvánvaló, hogy ezen okból fel kell darabolni a Java platformot jelentő API-t.

Azonban az elmúlt két-három évben a JavaFX egy fikarcnyit se mozdult el a nullával jellemezhető piaci részesedéséről, az elterjedben lévő platformok közül az iOS biztos nem fogja támogatni se a Java, se a JavaFX környezetet; az Android esetén kevéssé számít az, hogy modulárisan induljon el a VM, mert alapjaiban más megoldást alkalmaz erre a problémára; s végül a Windows Phone se fog Java/JavaFX programokat futtatni. A meglévő platformok közül az asztali és laptop/notebook vonal nem igényli a modularitást, a szerver oldali Java esetén pedig ott van az OSGi a futás idej modularitására (Eclipse, Glassfish és még néhány szerver oldali megoldás használja az OSGi-t), a fordítás idej modularitásra pedig ott a Maven és a többi függőséget jól kezelő rendszer.

Felmerül a kérdés, hogy *kell-e a JDK modularizációja*, vagy már elhaladt felette a kor?