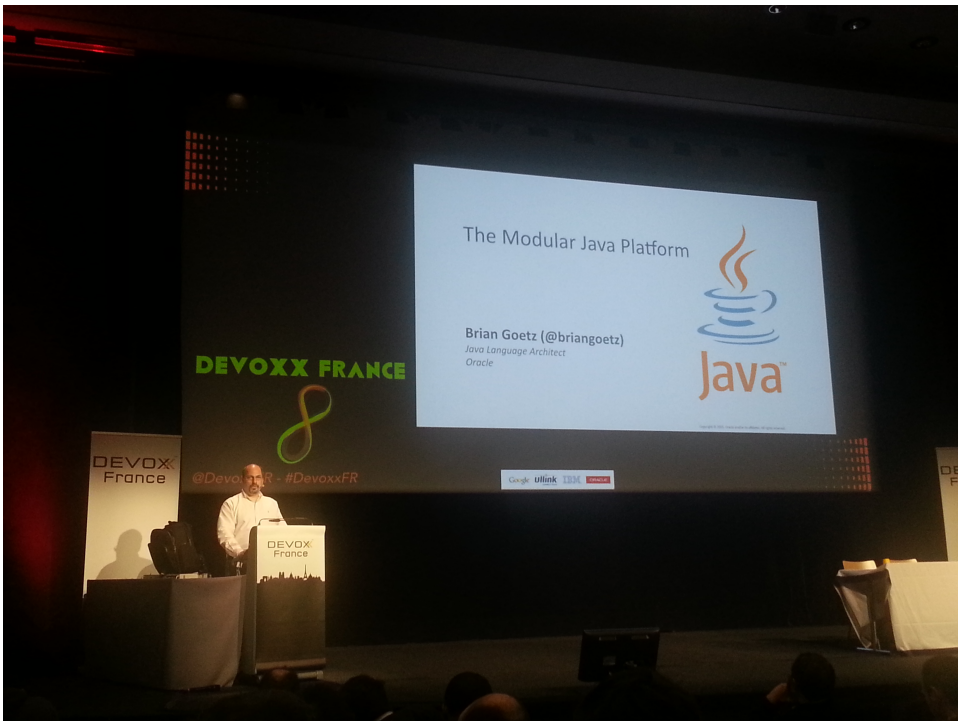
Modular Java Platform

Date : 9 avril 2015
Format : Conférence
Speakers : Brian Goetz, Java Language Architect chez Oracle



La prochaine grosse fonctionnalité de Java 9 est la modularisation (connu sous le nom de jigsaw).
A ses débuts, Java était petit : 2000 classes. Il a grossit au cours des versions.
Le runtime JavaSE est énorme : 55 Mb. C'est un problème à la fois pour les devices à la Raspberry Pi et pour les serveurs Sun.
Dans JME, il y'a un système de profiles : compact1 (11 Mo), compact2 (17 Mo), compact 3 (30Mo). Le problème est que si l'on a besoin d'une seule classe du compact3, il faut prendre tout le profil compact3.

Start performance vs runtime performance

La modularisation a un impact sur le système de sécurité de la JVM : la moitié du profil compact1 contient des packages en sun.* et *.internal.*
Les développeurs utilisent ces classes qui ne devraient être utilisées que par les classes du JDK.
A contrario, il existe des classes que personnes n'utilisent. Suite à un sondage de Brian, personne dans l'assemblée utilise le système de sécurité
(SecurityManager ::checkPackageAccess)

Jar Hell : application comportant 100 jars dans le classpath. Comment contrôler qu'il n'y a pas le même JAR de 2 versions différentes ? Le classpath n'est plus adapté aux applications

d'aujourd'hui. Lors de la création de Java, il n'avait pas envisagé qu'une application puisse comporter plus de 2 ou 3 JARs.

La modularisation du JDK doit permettre de résoudre tous les problèmes évoqués.

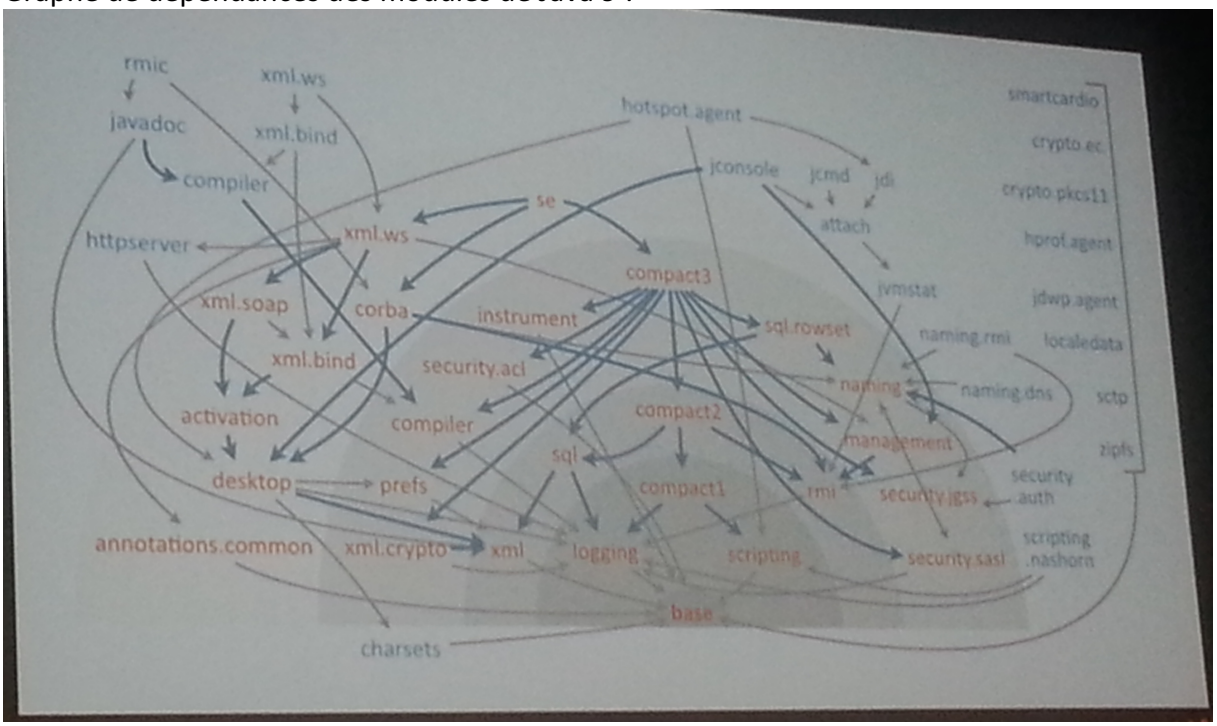
Qu'est-ce qu'un module ?

- Maitrise de sa visibilité

La 1ière étape consiste à modulariser le JDK. C'est l'objectif du projet Jigsaw qui comporte 3 JEP et une JSR :

- JEP 200 : The Modular JDK
- JEP 201 : Modular source code
- JEP 220 : Modular Run-Times Image
- JEP TBD JSR 376 : Java Platform Module System

Graphe de dépendances des modules de Java 9 :



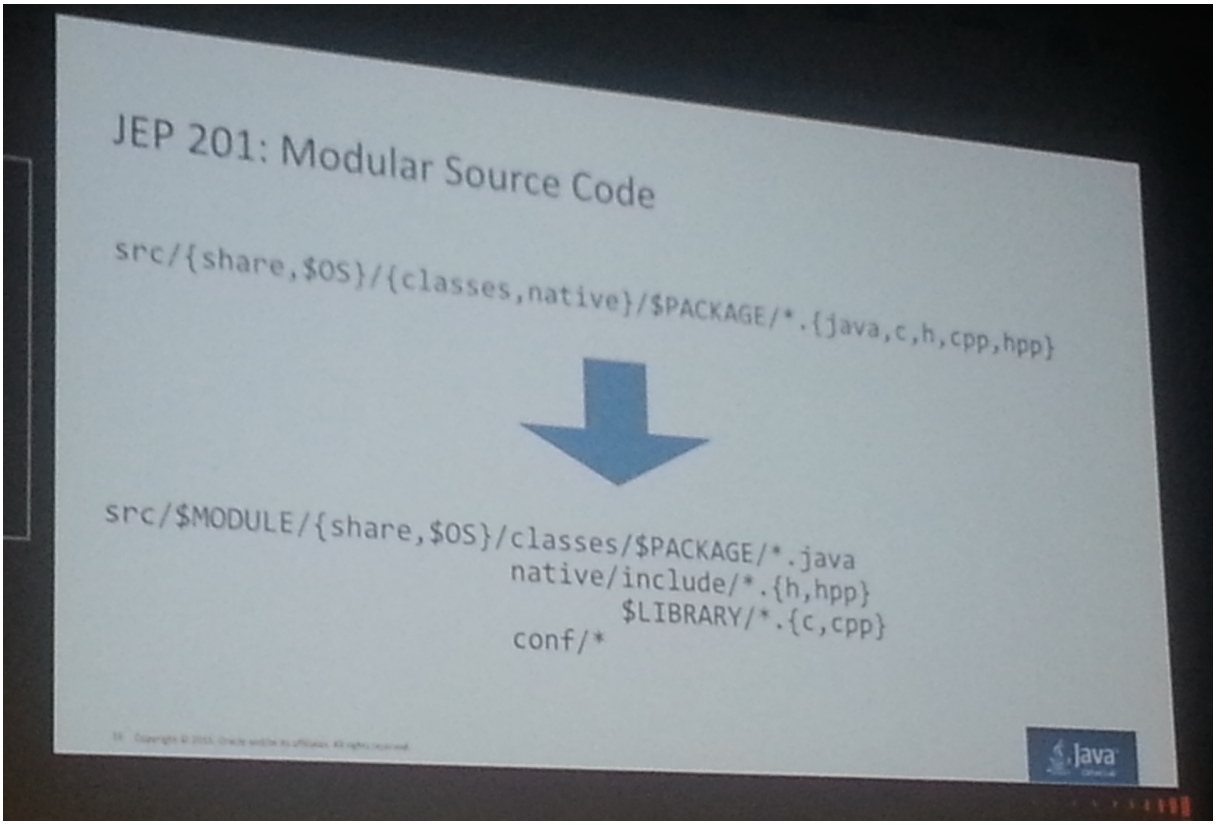
Légende :
Module en orange : module du JRE
Module en bleu : modules du JDK
Grey arrows : implementation dependencies
Blue arrows : runtime dependencies

Le module ***base*** contiendra les packages java.lang et java.util.
Système de dépendance transitive à la maven.

Démo en live de la liste des modules du JDK 9.
Puis utilisation de Java REPL : shell Java (jshell.sh). Autocomplétion. Méthode getModule() sur la classe Class.

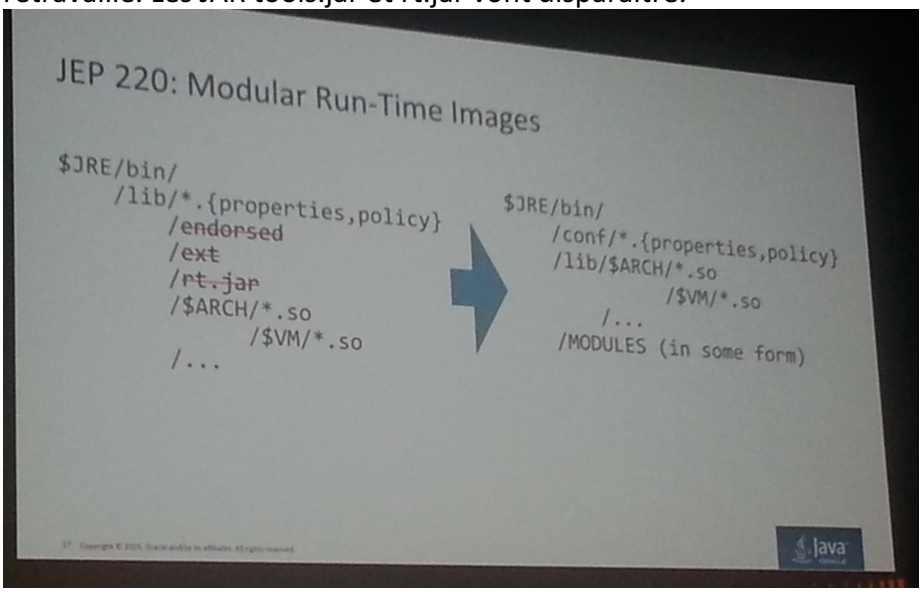
JEP 201 : Modular Source Code

Les fichiers du code source java, C et CPP doivent être repackagées



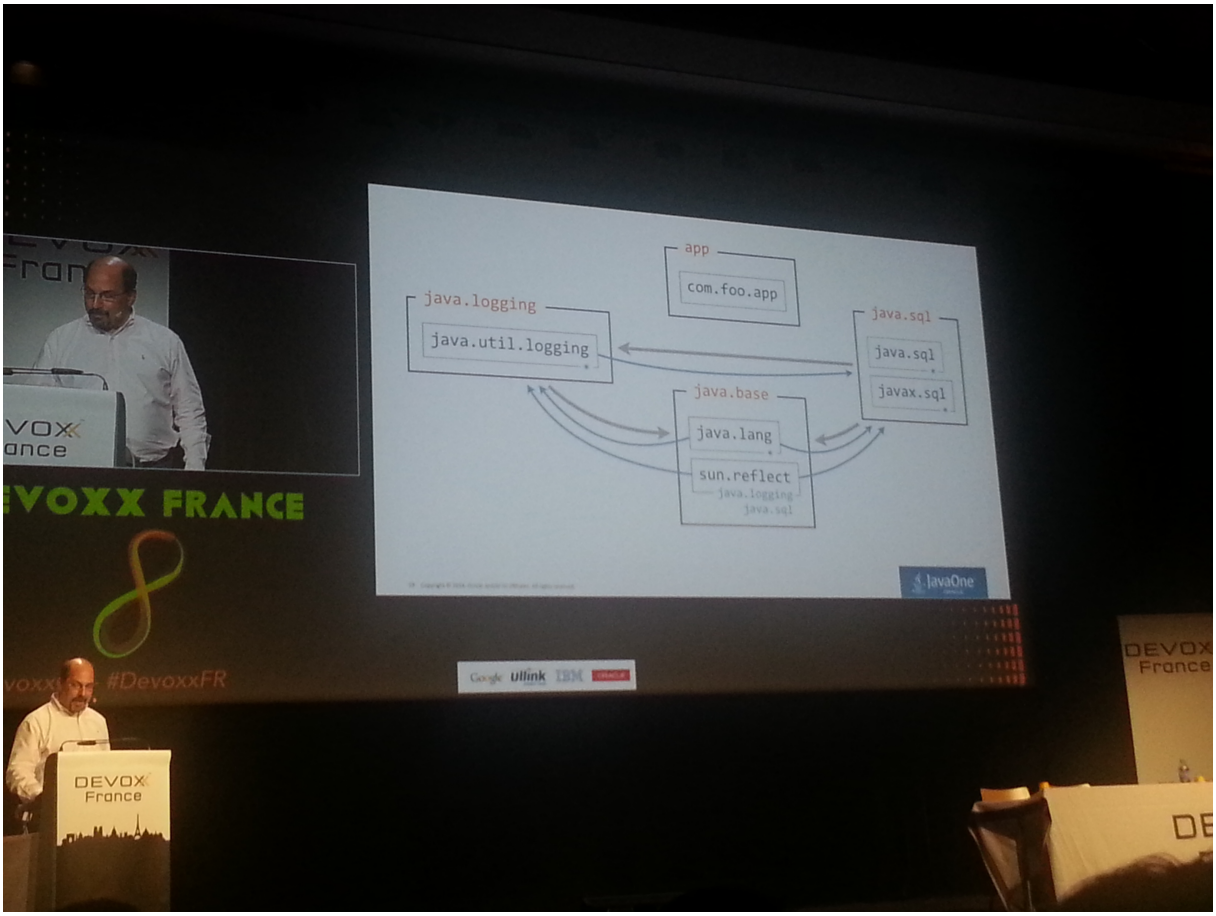
JEP 2220 : Modular Run-Time Images

L'arborescence du JRE avec ses répertoires lib, endorsed, ext et son rt.jar doit être retravaillé. Les JAR tools.jar et rt.jar vont disparaître.



L'appel à ClassLoader.getSystemResource() va renvoyer une URL différente :
jrt:/java.base/java/lang/Class.class

Le module java.sql contient les packages java.sql et javax.sql
Afin que le module java.sql puisse utiliser la classe String du module java.lang, ce dernier doit exporter la classe (rendre visible).
Le module java.sql a besoin des classes du package sun.reflect également compris dans le module java.base. Pour se faire, java.base n'expose ce package que pour le module java.sql



jmod est un nouveau format de module. C'est également le nom d'un outil pour créer des .jmod (comme l'exécutable jar)

L'outil jlink permet de créer une image sur mesure de la JVM avec les modules désirés.

Les projets open source utilisant les classes sun.* doivent se préparer à migrer.

JEP TBD JSR 376 : Java Platform Module System

- Permet aux développeurs de définir leurs propres modules
- S'intègre aux outils de builds comme Maven, Gradle et aux IDE
- Fournit une API de réflexion pour fournir des informations sur les modules
- S'intègre aux systèmes de packaging existant (ex : RPM)

Questions / Réponses

Question sur OSGI : pourquoi n'avoir pas retenu OSGI pour le JDK ? Le groupe d'expertise Jigsaw a passé beaucoup de temps à analyser OSGI. Il a été convenu que OSGI était trop grand et trop petit à la fois. Exemple : il a un système de cycle de vie qui dépasse le cadre de la modularisation.

Question sur la distribution des modules. Pas de réponse définitive. Ce n'est pas dans l'ADN d'Oracle de fournir un repository à la Maven Central. Ce dernier pourrait faire le job.