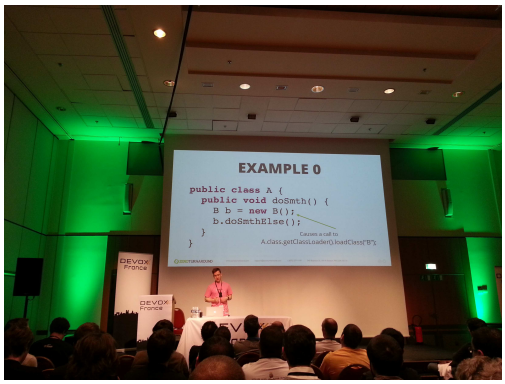
**Do you really get Classloaders ?**
Speaker : **Oleg Selajev** @shelajev - ZeroTurnaround
Devoxx France 2014
Vendredi 17/04/2014 10h40 – 11h30




Expérience du CLassloader acquise lors du développement de JRebel :

- Intégré à 20 serveurs d'app
- Ont solutionnés une centaine de problèmes liés aux classloaders

**Basics**

Classe abstraite java.lang.ClassLoader avec pour méthodes principales :
Class loadClass(String name)
ClassLoader getParent() : tout ClassLoader a un parent. Très importante pour ce talk.

Exemple 0
public class A {
static main(String.. args) {
B b = new B() ; => effectue un appel à A.class.getClassLoader().loadCLass("B") ;

Intentions :

- Chaque class loader a un class loader parent
- Le class loader parent est habituellement consulté en premier
- Pourtant, dans les Java EE web module classes sont cherchés en premier => permet d'embarquer des librairies plus récentes que celles fournies par le serveur d'app
- Chaque WAR de chaque EAR possède un classloader
- Permet d'avoir un namespaces entre applications

CL Hierarchy in Java EE

- Container
    - App1.ear
        - WAR1
        - WAR2

- App2.ear
  - WAR3
- App3.ear
  - WAR4

Quelles en sont les conséquences ?

Exemple 1 avec Tomcat :
Appel à new Util1().sayHello() depuis une servlet nommée Test1
Lors du démarrage => java.lang.NoClassDefFoundError : Util1
Affiche le classloader de la classe : Test1.class.getClassLoader().getUrl() => affiche cl-demo-jar.jar. Ce jar ne contient pas Util1.class.

Problème : pas la classe recherchée dans le Classpath = NoDef

Diagnostique :

- IDE class lookup (Ctrl + Shift + T)
- Print Test1.class.getClassLoader().getUrl()

Exemple 2 :

new Util2().sayHello() => NoSuchMethodError
La classe au runtime est différente de celle qui a été utilisée pour compiler l'application
javap -c Util2 permet d'afficher le bytecode de la classe Util2 ne contient pas la méthode sayHello()
=> volontaire pour la prés

Alien classes : IncompatibleClassCHangeError, AbstractMethodError, ClassCastException, Illegal
Diagnostique :

- -verbose :class
- Print getClassLoader().getResource()
- Utilisation de javap

Exemple 3 : CLassCastException « Util3 cannot be cast to Util3 »
Util3 u = (Util3) Factory3.instanceUntyped() ;

Diagnostique :
L'IDE ne voit pas le problème.
Test3.class.getCLassLoader().getResource(Util3.class)
Factory3.class.getCLassLoader().getResource(Util3.class)
2 fois la même classes : Util3.class présente à la fois dans shared.jar et cl-demo-jar.jar
La classe Factory3 appartient au Shared ClassLoader. Elle ne voit pas le Web ClassLoader.

Exemple 4 :

Factory3.instance().sayHello() => LinkageError Classloader constraint violation *<error message>*
Le message d'erreur est très explicite

Exemple 5 :
Factory3.instancePackage().sayHello() => IllegalAcessError
La méthode instancePackage() est de portée erreur

Diagnostique lorsqu'il y'a plus d'une seule classe :

1. -verbose :class
2. ClassLoader.getResource()

ClassLoader Leaks

- Mémoire occupée par les champs statiques
- Le fait qu'une instance référence sa classe qui référence également son classloader et ses parents pose problèmelors du redéploiement

Modern Patterns

- Chaque JAR a son propre class loader
- Class loaders are siblings ; + central repository
- Explicit imports / exports
- Repository can find relevant class loaders by package

Implémentations : JBoss Module, OSGI

**Conclusion**

Ne pas faire confiance à l'IDE et qualifier les raisons du disfonctionnement.