Devoxx France 2017

modulo
les
modules

Jean-Michel Doudoux **@jmdoudoux**

Le contenu de Java 9

LE gros morceau de Java 9

- C’est le projet Jigsaw
- Pour la mise en œuvre d’un système de modules

Avec toutes les contraintes que cela va induire

- Ruptures (visibilité, accessibilité, organisation du code, …)
- Mode de compatibilité (modules/classpath)

Le contenu de Java 9

Mais Java 9 propose de nombreuses autres fonctionnalités

- Dans le langage
- Dans les API
- Dans la plate-forme
- Dans la JVM

Spécifications définies dans la JSR 379
+ JSR 376 (modules system), ...

Intègre 89 JEPs

Jean-Michel Doudoux

CTO chez Oxiane Luxembourg

http://www.jmdoudoux.fr

@jmdoudoux

Java Champions

Auteur de 2 didacticiels
Diffusés sous licence GNU FDL

- Développons en Java (3400 pages)
- Développons en Java avec Eclipse

Co-fondateur du LOR'n'JUG, membre du YaJUG

Roadmap

- Les changements mineurs dans le langage
- De nouvelles API
- Des évolutions dans les API
- Des mises à jour des outils du JDK
- Des améliorations dans la JVM
- Le support de standards
- Des fonctionnalités diverses
- Les fonctionnalités deprecated ou retirées
- Les fonctionnalités reportées

Les changements mineurs dans le langage

JEP 213 : Milling project Coin

JEP 213

Quelques petites améliorations dans le langage

Trois sont liées au projet Coin (JSR 334) partiellement livré en Java 7

- étendre l'utilisation de @SafeVarags sur des méthodes privées
- étendre l'utilisation de l'opérateur diamant aux classes anonymes
- améliorer l'utilisation du try with resources

Plus deux autres améliorations :

- le caractère underscore est maintenant un mot clé réservé
- le support des méthodes privées dans les interfaces

Try with resources

JEP 213

Java 7 : introduit le try-with-resources

Oblige à déclarer une nouvelle variable qui sera celle gérée

Java 9 :

Utilisation possible de variables finales ou effectivement finales

Pour éviter de définir de nouvelles instances

```
// Java 7 et 8
public void ecrire(FileInputStream fis)
  throws Exception {
  try(FileInputStream fisr = fis) {
    // utilisation de fisr
  }
}
```

```
// Java 9
public void ecrire(FileInputStream fis)
  throws Exception {
  try(fis) {
    // utilisation de fis
  }
}
```

<> avec les classes anonymes

JEP 213

L'opérateur diamant peut être utilisé dans les classes anonymes

```
Comparator<Personne> compareParTaille = new Comparator<>() {
    @Override
    public int compare(Personne p1, Personne p2) {
        return p2.getTaille() - p1.getTaille();
    }
};
```

Le type générique doit pouvoir être inféré par le compilateur

```
public class Contenu<T> {

  private T valeur;

  public Contenu(T valeur) {
    this.valeur = valeur;
  }
}
```

```
<T> Contenu<T> creerContenu(T valeur) {
  return new Contenu<>(valeur) { };
}
```

```
Contenu<?> creerContenuAvecListePersonne(Personne valeur) {
  List<?> liste = Arrays.asList(valeur);
  return new Contenu<List<?>>(liste) { };
}
```

@SafeVarags

JEP 213

Les annotations sont héritées uniquement sur des classes

Avant Java 9, utilisable sur

- Des méthodes finales et statiques
- Et des constructeurs

Java 9 : utilisable sur des méthodes privées

Erreur de compilation dans les autres cas d'utilisation

Méthodes privées dans les interfaces

JEP 213

Permet de partager du code entre deux méthodes

En respectant l'encapsulation

Ne peuvent pas être privées :

- Les méthodes par défaut
- Les méthodes abstraites

```
public interface MonInterface {

    private static void m1() {}

    private void m2() {}

    private default void m3() {} // erreur

    private abstract void m4();  // erreur
}
```

L'identifiant _

JEP 213

```
public class TestUnderscore {
  private int _ = 0;
}
```

Java 8 : warning à la compilation

```
TestUnderscore.java:2: warning: '_' used as an identifier
  private int _ = 0;
              ^
  (use of '_' as an identifier might not be supported in releases after Java SE 8)
1 warning
```

Impossible d'utiliser _ dans les lambdas : erreur de compilation

```
TestUnderscore.java:4: error: '_' used as an identifier
  BiFunction<Integer, Integer, Long> bf = (a, _) -> (long) a + _;
                                           ^
  (use of '_' as an identifier is forbidden for lambda parameters)
```

L'identifiant _

JEP 213

Java 9 : n'est plus valide, erreur de compilation

en prévision d'une future utilisation dans les lambdas

```
TestUnderscore.java:2: error: as of release 9, '_' is a keyword, and may not be used as
an identifier
  private int _ = 0;
              ^
1 error
```

De nouvelles API

StackWalking API

JEP 259

Architecture en couches et utilisation de frameworks

- Génère de très grosses stacktraces
- Généralement seules les frames de nos classes sont intéressantes

API pour parcourir, filtrer et accéder aux infos de stacktraces

L'interface StackWalker.StackFrame

- Encapsule des informations sur une frame de la pile
- String getClassName(), Class<?> getDeclaringClass(), String getFileName(),
- int getLineNumber(), String getMethodName(), ...

StackWalking API

JEP 259

La classe thread-safe StackWalker

surcharges de la fabrique static StackWalker getInstance()

<T> T walk(Function<? super Stream<StackWalker.StackFrame>,? extends T> function)

```
public static List<String> filtrerStackCourante() {
   return StackWalker.getInstance().walk(s -> s
        .map(frame -> frame.getClassName() + "." + frame.getMethodName() + "():"
            + frame.getLineNumber())
        .filter(c -> c.startsWith("com.jmdoudoux")).limit(10)
        .collect(Collectors.toList()));
 }
```

Platform specific desktop features

JEP 272

Enrichir les possibilités d'interactions avec le système hôte

- Bureau, barre de tâches, événements, ...

Nouveau package java.awt.desktop dans le module java.desktop

Nouvelles méthodes dans la classe java.awt.Desktop

Toutes les fonctionnalités ne sont pas supportées

- Sur toutes les plateformes

Reactive Stream (Flow API)

JEP 266

Ensemble minimal de 4 interfaces et une classe

Pour mettre en œuvre des Reactive Streams (<> Stream de Java 8)

Mécanismes de type publication/souscription asynchrone

- Pour consommer des éléments d'un flux au fur et à mesure de leur disponibilité
- Peut être vu comme une combinaison des patterns Iterator et Observer

L'interface fonctionnelle Flow.Producer<T>

- Pour le producteur d'éléments
- void subscribe(Flow.Subscriber<? super T>)

Reactive Stream (Flow API)

JEP 266

L'interface Flow.Subscriber<T>

Pour consommer des éléments d'un producteur

void onNext(T)

void onComplete()

void onThrowable(Throwable)

void onSubscribe(Flow.Subscription)

L'interface Flow.Subscription

Pour gérer la communication entre producteur et consommateur

void cancel()

void request(long)

Reactive Stream (Flow API)

JEP 266

L'interface fonctionnelle Flow.Processor<T, R>

- Pour transformer un flux en un autre
- Hérite de Flow.Publisher<T> et Flow.Subscriber<R>
- S'utilise entre un producteur et un consommateur

Intérêts :

- Mécanisme de type back pressure
- Consommation de moins de ressources

Des évolutions dans les API

L'API Process

JEP 102

La gestion des processus en Java est limitée

- Avant Java 5 : Runtime.getRuntime().exec()
- Java 5 : ProcessBuilder

Java 9 enrichi l'API avec l'interface ProcessHandle

- Pour obtenir le processus courant : static ProcessHandle current()
- Pour obtenir des infos : ProcessHandler.Info info()
- Terminer un processus : boolean destroy() et destroyForcibly()
- Pour obtenir tous les processus : Stream<ProcessHandle> allProcesses()
- Pour obtenir les processus fils : Stream<ProcessHandle> children() ou descendants()
- ProcessHandle toHandle() de la classe ProcessBuilder

L'API Process

JEP 102

L'API permet de déclencher des traitements à la fin d'un processus

CompletableFuture<ProcessHandle> onExit() de ProcessHandle

Méthode CompletableFuture<Process> onExit() de Process

L'OS peut restreindre les possibilités d'accès ou de gestion des processus

Fabriques pour collections

JEP 269

Pas de solution simple pour créer des instances immuables

```
Set<String> set = new HashSet<>();
set.add("a");
set.add("b");
set.add("c");
set.add("d");
set =
Collections.unmodifiableSet(set);
```

```
Set<String> set = Collections.unmodifiableSet(new HashSet<>(Arrays.asList("a", "b",
"c", "d")));
```

```
Set<String> set = Collections.unmodifiableSet(new HashSet<String>() {{
    add("a"); add("b"); add("c"); add("d");
}});
```

```
Set<String> set = Collections.unmodifiableSet(Stream.of("a", "b", "c",
"d").collect(toSet()));
```

Fabriques pour collections

JEP 269

Ajout de fabriques statiques dans les interfaces List, Set et Map

Pallier au fait qu'il n'existe pas de syntaxe littérale pour créer une collection

Faciliter la création d'instances immuables

12 surcharges de la méthode of()

pour 0, 1 ou plusieurs éléments

```
Set<String> set =  Set.of("a", "b", "c", "d");
```

Fabriques pour collections

JEP 269

Les contraintes sur les instances obtenues :

- Sont des implémentations fournies par Java 9
  - inner classes statiques de java.util.ImmutableCollections
- Sont des values based
- Sont sérialisables si les éléments sont Serializable
- NPE si null est passé en paramètre de of() pour List et Set
- Pas null comme clé ou valeur en paramètre de of() pour Map

Attention : pas d'interface dédiée pour immuable comme en Guava

-> utiliser des conventions de nommage

```
Set<String> setImmuable =  Set.of("a", "b", "c", "d");
```

Trois nouvelles méthodes dans Optional

Stream<T> stream()

Renvoie un Stream vide ou contenant la valeur

Optional<T> or(Supplier<? extends Optional<? extends T>>)

Si une valeur est présente alors renvoie un Optional de la valeur

Sinon renvoie l'Optional fourni par le Supplier

void ifPresentOrElse(Consumer action, Runnable emptyAction)

Si une valeur est présente alors action est exécutée

Sinon c'est emptyAction

Quatre nouvelles méthodes dans l'API Stream

Les deux premières sont complémentaires à limit() et skip()

Stream<T> takeWhile(Predicate<? Super T>)

Renvoie les éléments du Stream tant que le prédicat est vérifié

```
Stream<Integer> chiffres = Stream.iterate(0, n -> n + 1);
chiffres.takeWhile(n -> n < 10).forEach(System.out::println);
```

Stream<T> dropWhile(Predicate<? Super T>)

Ignore les éléments du Stream tant que le prédicat est vérifié

```
Stream<Integer> chiffres = Stream.iterate(0, n -> n + 1);
chiffres.dropWhile(n -> n <= 9).limit(10).forEach(System.out::println);
```

Lors de l'utilisation sur des Stream non ordonnés

Quatre nouvelles méthodes dans l'API Stream

static Stream<T> ofNullable(T)

Renvoie un Stream avec l'élément ou vide s'il est null

Equivalent à `(e == null) ? Stream.empty() : Stream.of(e);`

Une nouvelle surcharge de la méthode iterate() avec Predicate

```
static <T> Stream<T> iterate(T seed, Predicate<? super T> predicate, UnaryOperator<T> f)
```

en Java 8

```
IntStream.iterate(1, i -> i + 1).limit(10).forEach(System.out::println);
```

en Java 9

```
IntStream.iterate(1, i -> i <= 10, i -> i + 1).forEach(System.out::println);
```

```
1
2
3
4
5
6
7
8
9
10
```

Spin-wait hints

JEP 285

- onSpinWait() ajoutée dans la classe Thread
  - Méthode vide annotée avec @HotSpotIntrinsicCandidate
  - Pour éventuellement réduire la consommation CPU
  - Dans les boucles d'attente dans des threads
- Utilisation possible d'instructions du microprocesseur
  - Par le compilateur C2 de la JVM HotSpot
  - Exemple l'instruction PAUSE sur x86
- Utilisée dans plusieurs implémentations de classes
  - Phaser, StampedLock, SynchronousQueue<E>

Des mises à jour des outils du JDK

JDK/JRE organisation des fichiers

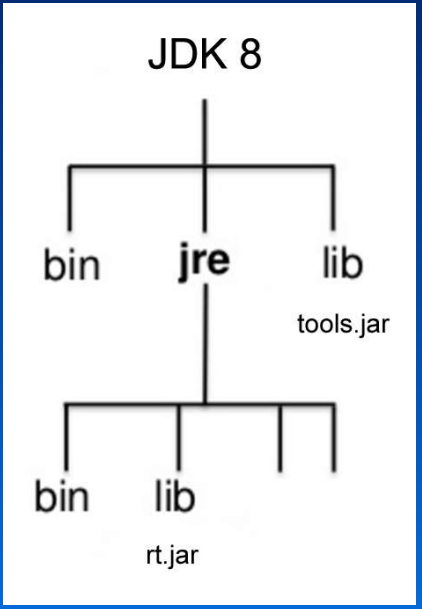
JEP 220

Avant Java 9

JDK contient un JRE dans le sous-répertoire jre

- le répertoire bin contient des outils de dev
- mais aussi certaines commandes du JRE

Des fichiers jar dans jdk/lib et jre/lib



JDK/JRE organisation des fichiers

JEP 220

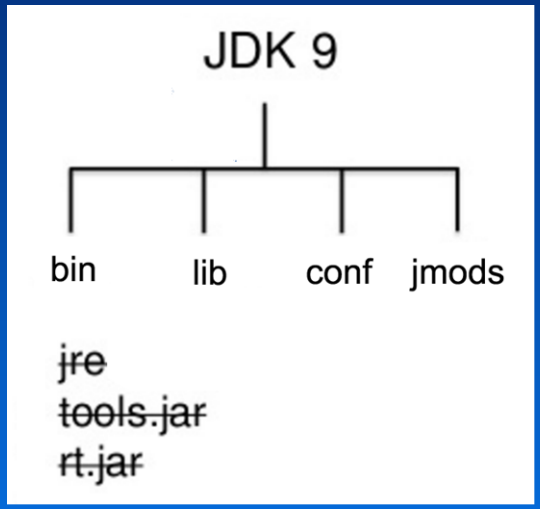
Avec Java 9

Plus de distinction JDK/JRE

- Dans l'organisation des fichiers

Répertoires :

- bin : les outils
- conf : fichiers de config
- lib:
- jmods : les modules



JDK/JRE organisation des fichiers

JEP 220

Le mécanisme endorsed est retiré

utilisait JAVA_HOME/lib/endorsed

pour remplacer une version d'une API livrée en standard

Le mécanisme d'extension est retiré

utilisait JAVA_HOME/lib/ext

Les fichiers lib/rt.jar et lib/tools.jar sont retirés

+ lib/dt.jar d'autres fichiers jar

GNU style command Lines options

JEP 293

Les options des outils en ligne de commande de Java ont de nombreuses particularités :

certaines versions longues utilisent un tiret (-version), d'autres deux (--version)

certains mots sont séparés par un tiret (--no-header), d'autres non (-javaagent)

certaines options courtes ont une lettre (-d), d'autres deux lettres (-cp)

Les valeurs sont précisées avec un = (-D<cle>=<valeur>), d'autres avec un espace

même l'option pour obtenir l'aide varie : -help, -?, --help

GNU style command Lines options

JEP 293

Le but est d'harmoniser les nouvelles options avec une syntaxe utilisant style GNU

- sensible à la classe
- option longue : commence par --, chaque mot séparé par -
  - la valeur est séparée avec un espace ou un =
  - --module-path path
- option courte : un seul caractère, commence par -
  - peuvent être groupée : -ea
  - la valeur est séparée avec un espace ou suit l'option
- plusieurs valeurs séparées par une virgule, sauf fichiers/répertoires (; sous Windows sinon :)

Les anciennes options restent (compatibilité)

Remove GC options

JEP 214

Deprecated en Java 8 (JEP 273)

Ces combinaisons sont supprimées

```
DefNew + CMS        : -XX:-UseParNewGC -XX:+UseConcMarkSweepGC
ParNew + SerialOld  : -XX:+UseParNewGC
ParNew + iCMS       : -Xincgc
ParNew + iCMS       : -XX:+CMSIncrementalMode -XX:+UseConcMarkSweepGC
DefNew + iCMS       : -XX:+CMSIncrementalMode -XX:+UseConcMarkSweepGC
                      -XX:-UseParNewGC
CMS foreground      : -XX:+UseCMSCompactAtFullCollection
CMS foreground      : -XX:+CMSFullGCsBeforeCompaction
CMS foreground      : -XX:+UseCMSCollectionPassing
```

Policy for retiring javac –source and -target

JEP 182

Java 8

Compilation avec d'anciennes versions

```
public class TestAssert {

  public int assert;

}
```

```
C:\java>javac -source 1.3 TestAssert.java
warning: [options] bootstrap class path not set in conjunction with -source 1.3
warning: [options] source value 1.3 is obsolete and will be removed in a future release
warning: [options] target value 1.4 is obsolete and will be removed in a future release
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.
TestAssert.java:3: warning: as of release 1.4, 'assert' is a keyword, and may not be
used as an identifier
  public int assert;
             ^
  (use -source 1.4 or higher to use 'assert' as a keyword)
5 warnings
```

Policy for retiring javac –source and -target

JEP 182

Java 9

Support de la version courante et de trois versions précédentes

```
C:\java>"C:\Program Files\Java\jdk-9\bin"\javac -source 8 -target 8
--boot-class-path "C:\Program Files\Java\jdk1.8.0_60\jre\lib\rt.jar" Hello.java
```

9 par défaut et 8 , 7 et 6 (warning si utilisé)

```
warning: [options] source value 1.6 is obsolete and will be removed in a future
release
warning: [options] target value 1.6 is obsolete and will be removed in a future
release
warning: [options] To suppress warnings about obsolete options, use -Xlint:-
options.
```

Compile For Older Platform versions

JEP 247

Option --release N du compilateur

N peut prendre les valeurs de –source et -target

Equivalent à

```
javac -source N -target N –bootclasspath rtN.jar…
```

```
C:\java>"C:\Program Files\Java\jdk-9\bin"\javac --release 8 Hello.java

C:\java>
```

Multi-release jar files (MRJAR)

JEP 238

Avant Java 9

- Un jar pour chaque version de Java

```
Racine du fichier jar
  MaClasse.class
  META-INF
    MANIFEST.MF
```

Avec Java 9

- Un jar peut contenir plusieurs .class pour chaque version
  - La version courante à la racine
  - Les autres versions, dans META-INF/versions/N
- Multi-Release: true dans MANIFEST.MF
- Mise à jour
  - Des outils jar, javac, java
  - De la classe java.util.jar.JarFile

```
Racine du fichier jar
  MaClasse.class
  META-INF
    MANIFEST.MF
    versions
      9
        MaClasse.class
      10
        MaClasse.class
```

JShell

JEP 222

Read Eval Print Loop

Interpréteur interactif en ligne de commandes

Projet Kulla

Evite d'avoir à ajouter une méthode main()

ou utiliser des fonctionnalités de l'IDE (ex: Scrapbook sous Eclipse)

Outils retirés de Java 9

JEP 241

jhat

- Outil expérimental fourni avec Java 6
- D'autres outils de visualisation du heap sont disponibles

Java DB

- Base de données Apache Derby
  - Fournie avec les JDK 7 et 8
  - Dans le sous-répertoire db

Des améliorations dans la JVM

New version String format

JEP 248

Le numéro de version n'a jamais été simple à comprendre

- hormis pour les deux premières versions 1.0 et 1.1

Les versions suivantes 1.2 à 1.5 ont été rebaptisées Java 2 :

- J2SE pour Java 2 Standard Edition, J2EE, J2ME

A partir de la version 1.5, Java a été renuméroté Java 5

- Cependant la version indiquée par la JVM est toujours de la forme 1.x.y

Oracle a introduit un nouveau format de version

- pas forcement plus clair

New version String format

JEP 248

Les versions non relatives à la sécurité (Limited Update Releases)

sont des multiples de 20

Les versions relatives à la sécurité (Critical Patch Updates)

valeurs multiples de 5 avec éventuellement l'ajout de 1 pour garantir une valeur impaire

par rapport à la version LUR sur laquelle elle se base, exemple :

LUR : 7u40

CPU suivantes : 7u45, 7u51, 7u55

Différentes représentations d'une même version d'un JDK

JDK 7 Update 60, 1.7.0_60, JDK 7u60

New version String format

JEP 248

Nouveau format inspiré de SemVer

Composé de quatre parties séparées par un point

$MAJOR : version majeure qui fait l'objet d'une spécification

$MINOR : version mineure concernant des bug fixes mais aussi des mises à jour des API standards

$SECURITY : version contenant des corrections critiques notamment relatives à la sécurité

$PATCH

Une API est fournie pour obtenir des info sur le numéro de version

G1 par défaut

JEP 223

Ramasse-miettes conçus pour réduire les temps de pause

- Notamment pour des heaps de taille importante
- Découpe une génération en régions de taille fixe

Remplace le Parallel GC comme GC par défaut

pour les JVM 32 et 64 bits en mode server

Compact Strings

JEP 254

Réduire l'empreinte mémoire

- Nécessaire au stockage interne des chaînes de caractères

Important car une grosse part du heap

- est utilisée pour stocker des chaînes de caractères

Jusqu'à Java 8 :

- Tableau de caractères (char[]), chacun sur 16 bits
- Encodés en UTF-16

Compact Strings

JEP 254

Java 6 : première tentative avortée (retirée en Java 7)

-XX:+UseCompressedStrings

Java 9 :

Tableau d'octets (byte [])

Un octet précise le format : UTF16 ou Latin1

Important : aucun impact sur l'interface de la classe String

Désactivation avec -XX:-CompactStrings

Indify String concatenation

JEP 280

Remplacer le byte-code généré par le compilateur

Pour concaténer des chaînes avec StringBuilder (invokevirtual)

En utilisant invoke dynamic

Permettra de futures optimisations sans changer le byte-code

généré par le compilateur

Le support de standards

Unicode 7.0 et 8.0

JEP 227 267

Java 8 supporte Unicode 6.2

Java 9 support Unicode 6.3, 7.0 et 8.0

Javadoc

JEP 224 225

Maintenant en HTML5

javadoc –html5

Vue par packages ou modules

Recherche dans la Javadoc

Implémentation en Javascript côté client

PKCS12 dans le keystore

JEP 229

Utilisation par défaut dans le keystore

- En remplacement de JKS

Améliore la sécurité

- En utilisant des algorithmes plus forts

SHA-3

JEP 287

Implémentation de fonctions de hachage SHA-3

Selon les spécifications NIST-FIPS 202

Algorithmes implémentés :

SHA3-224, SHA3-256, SHA3-384 et SHA3-512

```
String chaine = "Bonjour";
md = MessageDigest.getInstance("SHA3-256");
byte donnees[] = md.digest(chaine.getBytes());
```

UTF-8 properties files

JEP 226

Avant Java 9

- Les fichiers properties doivent être encodés en ISO-8859-1
- L'outil native2ascii pour convertir l'encoding du fichier

En Java 9

- La classe PropertyResourceBundle supporte l'encoding en UTF-8
- Si une séquence d'octets invalide est détectée, relecture en ISO-8859-1
- Option de la JVM pour forcer la lecture

```
-Djava.util.PropertyResourceBundle.encoding=ISO-8859-1
```

Les fonctionnalités diverses

Enhanced deprecation

JEP 277

Enrichir @Deprecated

boolean forRemoval() : prévue pour être retirée dans le future ?

false par défaut

String since() : version à partir de laquelle c'est deprecated

"" par défaut

Plusieurs @Deprecated ont été

ajoutées, modifiées, supprimés (java.awt.Component.show() et hide())

Nouvel outil jdeprscan pour rechercher les API Deprecated utilisées

Incubator modules

JEP 11

Permet de développer et livrer des modules non standard

- en cours de développement

Ces modules sont préfixés par jdk.incubator

Les packages exportés sont aussi préfixés par jdk.incubator

- Leur nom sera changé une fois finalisé

Leur utilisation requiert l'option --add-modules

Un seul module d'incubation dans Java 9

- HTTP 2.0 dans le module jdk.incubator.httpclient

Les fonctionnalités deprecated ou retirées

Applet

L'API Applet est deprecated

java.applet.AppletStub, java.applet.Applet, java.applet.AudioClip,
java.applet.AppletContext, javax.swing.Japplet
L'outil appletviewer

JEP 289

Le support du plug-in par les navigateurs

est retiré ou le retrait est planifié

Solution de remplacement : Java Web Start

Pas de planification de suppression de l'API dans le futur

Demos et exemples

JEP 298

Obsolètes et non maintenus

Retirés dans Java 9

Remove JVM TI hprof agent

JEP 240

Les fonctionnalités utiles sont dans d'autres outils

- Heap dump avec jmap –dump
- Allocation profiling avec VisualVM
- CPU profiling avec VisualVM ou Java Flight Recorder

Les fonctionnalités reportées

Les fonctionnalités reportées

Initialement, Java 9 devrait aussi intégrer :

- JEP 230 : Microbenchmark Suite basé sur JMH
- JEP 198 : Light weight Json API
- JEP 110 : client HTTP/2 et websocket (incubator)
- JSR 354 : Money and currency API

Malheureusement celles-ci sont reportées

- Dans une version ultérieure de Java

Conclusion

Conclusion

Java 9 propose aussi de nombreuses autres fonctionnalités :

- JEP 266 : More concurrency updates CompletableFuture
- JEP 193 : Variable Handles
- JEP 274 : Enhanced methods handles
- JEP 231 : Remove Launch-time JRE version selection
- JEP 158 : Unified JVM Logging
- JEP 271 : Unified GC Logging
- JEP 246 : Leverage CPU Instructions for GHASH and RSA
- JEP 265 : Marlin Graphics Renderer
- JEP 197 : Segmented Code Cache
- JEP 288 : Disable SHA-1 Certificates
- JEP 219 : Datagram Transport Layer Security (DTLS)
- JEP 268 : XML Catalogs
- ...

Conclusion

La release de Java est prévue pour le 27 juillet

Il est possible de télécharger les versions EA (Early Access)

https://jdk9.java.net/download/

Pour essayer ces fonctionnalités et les modules

Avec le support (encore expérimental) des principaux IDE

Consultez la documentation fournie par Oracle

http://docs.oracle.com/javase/9/index.html

Merci / Thank you