JUG Trier

Java >8 für Umsteiger

Frank Dietrich
(Clearstream S.A., Luxembourg)

Eine kurze Übersicht über Änderungen in Java 9 - 12

- Neuigkeiten
- Änderungen
- Entfernungen

generelle Änderungen

- fester Releasezyklus
  März (non LTS) und September (LTS)
- Abwärtskompatibilität
  @Deprecated Funktionalität kann entfernt werden

Java 9 – September 2017 (non LTS)

- implementiert 91 JEP
  http://openjdk.java.net/projects/jdk9/
- Wechsel von Java 8 auf 9 nicht so leicht wie zuvor
- größte Änderung: Java Platform Module System
- tools.jar wurde in Module aufgeteilt
- JavaDB nicht mehr Bestandteil vom JDK

Java 9
JEP 102 - Process API Updates

- ProcessHandle liefert Process Informationen
  PID, Startzeit, CPU Zeit, Kommandozeile
- CompleteableFuture um Aktion nach dem Beenden auszuführen

Java 9
JEP 158 – Unified JVM Logging

- Kommandozeilen Optionen für einheitliches Logging
  java -Xlog:os+cpu=info -version
- alle Optionen anzeigen
  java -Xlog:help

Java 9
JEP 211 – Elide Deprecation warning

- keine Deprecation Warnung mehr für import
  import java.io.LineNumberInputStream;
  Note: DeprecatedImport.java uses or overrides a deprecated API.
  Note: Recompile with -Xlint:deprecation for details.

Java 9
JEP 213 – Milling Project Coin

- @SafeVArgs an privaten Methoden
  @SafeVarargs
  private void foobar(Object... args) { … }
- final Variablen in try-with-resource
  AutoCloseable closeable = new FileInputStream(…)
  try (closeable) { … }
- ‘_’ ist kein gültiger Bezeichner mehr
- private Methoden in Interfaces

Java 9
JEP 214 – Remove GC Combinations

- die entfernten Kombinationen wurden in Java 8 als Deprecated markiert
- bestimmte GC Kombinationen wurden entfernt

```
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
```

Java 9
JEP 221 – new Doclet API

- Ersatz für die Doclet API, Zugunsten der Standard language-model API und der Compiler Tree API

```
com.sun.javadoc --> jdk.javadoc.doclet

com.sun.tools.doclets.standard.Standard --> jdk.javadoc.doclet.StandardDoclet
```

Java 9
JEP 222 – JShell

- Java Shell als Read-Eval-Print Loop
- interaktive Shell zum ausführen von Java Code-Schnipseln
  ```
  jshell> Long.toHexString(3_405_691_582L)
  $1 ==> "cafebabe"
  ```
- Code-Completion, Dokumentation, Eingabehistorie

Java 9
JEP 223 – New Version-String

- die Struktur des Java Version Strings hat sich geändert

```
java.runtime.version         : 1.8.0_202-b08
java.version                 : 1.8.0_202
java.vm.specification.version: 1.8
java.vm.version              : 25.202-b08

java.runtime.version         : 9.0.4+11
java.version                 : 9.0.4
java.vm.specification.version: 9
java.vm.version              : 9.0.4+11
```

Java 9
JEP 224 + 225 – javadoc

- javadoc wurde erweitert und erzeugt nun HTML5 Seiten
- die Dokumentation enthält eine Suchfunktion

Java 9
JEP 226 – UTF-8 Property Files

- die Zeichenkodierung für ResourceBundleDateien wurde auf UTF-8 geändert
- bisherige Kodierung war ISO-8859-1

```
-Djava.util.PropertyResourceBundle.encoding=ISO8859-1
```

Java 9
JEP 229 – PKCS12 keystore

- der Standardtyp eines Java Keystores wurde zu PKCS12 geändert
- JKS ist ein JDK spezifisches Format
- bestehende Keystores werden weiterhin unterstützt

Java 9
JEP 240 + 241 – hprof / jhat removal

- der “Heap and CPU Profiling” Agent wurde entfernt

```
java -Xrunhprof:format=b,file=heap.dump Application
```

- ebenso jhat, das Programm zum Analysieren

```
jhat hprof.dump
```

- einen Heap-Dump erstellen auch andere Tools

```
jcmd <pid> GC.heap_dump heap.dump
jmap -dump:format=b,file=heap.dump 17030
```

- anzeigen lässt er sich auch komfortabler, als mit jhat

```
visualvm –openfile heap.dump
```

Java 9
JEP 247 – compile for older version

- --release kompiliert für n-3 Platform Versionen
- die jeweiligen Klassenrümpfe der Zielplatformen befinden sich in $JDK_ROOT/lib/ct.sym
- so ist sichergestellt, dass der erstellte Bytecode auf der Zielplattform ausgeführt werden kann

Java 9
JEP 248 – G1 default GC

- Standard Garbage-Collector ist der G1
- Test sind nötig bezüglich des Laufzeitverhaltens und des Speicherbedarfs

Java 9
JEP 252 – CLDR locale data

- Gebietsschema Informationen werden nun vom CLDR des Unicode Consortiums verwendet
- unter Java 8 musste die Verwendung der CLDR Daten explicit angegeben werden
  -Djava.locale.providers=JRE,CLDR
- die Verwendung eines anderen Gebietsschema Providers kann zu gändertem Verhalten bei Datums-, Zeit- und Zahlenformaten führen

Java 9
JEP 252 – CLDR locale data

```
Calendar calendar = Calendar.getInstance(Locale.GERMANY);
calendar.set(2019, Calendar.MAY, 1);
DateFormat dayFormatter = new SimpleDateFormat("EEE ", Locale.GERMANY);
for (int day = 13; day < 20; day++) {
    calendar.set(Calendar.DAY_OF_MONTH, day);
    System.out.print(dayFormatter.format(calendar.getTime()));
}
```

Java 8 Mo Di Mi Do Fr Sa So
Java 9 Mo. Di. Mi. Do. Fr. Sa. So.

Java 9
JEP 254 – compact strings

- bessere Speichernutzung bei String Objekten
- bisher wurde die Zeichen eines String Objekts in einem char[] gespeichert (2 Byte pro Zeichen)
- String Objekte gehören zu den Komponenten, die mit den meisten Platz auf dem Heap belegen
- darüber hinaus benutzen die meisten Strings nur den Latin-1 Zeichensatz, für jedes Zeichen würde 1 Byte zur Speicherung genügen, 50% Platz ungenutzt
- neu: Speicherung der Zeichen in byte[] und zusätzlich die Kodierungsinformation

Java 9
JEP 259 – StackWalker

- Standard API um auf die Aufrufhistorie zugreifen zu können
- gegenüber einem Exception Stacktrace bietet es Zugriff auf zusätzliche Informationen, wie Zeitpunkte, Methodennamen, Filtermöglichkeiten

Java 9
JEP 260 – encapsulate internal API

- Maßnahmen um die Verwendung interner APIs vollständig zu verhindern
- Einteilung der internen APIs in
  nicht kritische API – z.B. sun.misc.BASE64Decoder
  kritische API – z.B. sub.misc.Unsafe
- kritische APIs für die in Java 8 noch keine Alternative existiert sind nicht gekapselt (Module jdk.unsuported)
- kritische APIs für die in Java 9 eine Alternative existiert sind deprecated in Java 9, werden in einer kommenden Version entweder gekapselt oder entfernt

Java 9
JEP 269 – Factory Methods

- statische Factory-Methoden zur Erzeugung von Collections (List, Set, Map) mit wenigen Elementen
- erzeugte Collections sind strukturell immutable
- die Implementierung ist frei neue Instanzen zu erzeugen oder wiederzuverwenden
- können Exceptions werfen
  NullPointerException - “null” Element
  IllegalArgumentException – doppelter Schlüsselwert

Java 9
JEP 271 – unified GC logging

- Nutzung des unified JVM logging (JEP158)
- einzeilige Ausgabe
  -Xlog:gc und -XX:PrintGC
- vergleichbare Informationen
  -Xlog:gc und -XX:PrintGCDetails
- Zeitstempel in der Ausgabe

Java 9
JEP 282 – jlink – The Java Linker

- Erstellung modularer Laufzeitumgebungen
- JRE mit Modulen die zur Laufzeit benötigt werden
  JDK 9: ~ 565 MB
  JRE mit java.base: ~ 45 MB
- benötigte Module können mit jdeps ermittelt werden

Java 9
JEP 289 – Deprecate Applet API

- noch keine endgültige Entfernung in Java 9

```
@Deprecated(since=“9")
    java.applet.AppletStub
    java.applet.Applet
    java.applet.AudioClip
    java.applet.AppletContext
    javax.swing.JApplet
```

Java 9
JEP 291 – Deprecate CMS GC

- Zur Verringerung von Maintenance Kosten und zugunsten anderer Kollektoren wird der Concurrent Mark Sweep Garbage Collector in einem kommenden entfernt werden.
- JVM Option -XX:+UseConcMarkSweepGC gibt eine Warnung aus

```
Option UseConcMarkSweepGC was deprecated in version 9.0 and
will likely be removed in a future release.
```

Java 9
JEP 295 – Ahead of Time Compilation

- Verbesserung der Startzeiten von Java-Programmen
- Code-Generierung durch Projekt Graal
- Implementierung ist für “experimentelle” Verwendung

```
- Linux x64 Systeme mit 64-bit Java, ParallelGC / G1GC
- AOT Kompilierung auf gleichem System wie zur Laufzeit
- Java Code der dynamisch generierte Klassen oder Bytecode
  verwendet wird nicht von AOT kompiliert
- AOT unterstützt keine Custom Class Loader
```

Java 10
JEP 286 – Local-Var. Type Inference

- Java-Code kann kompakter formuliert werden, ohne Typensicherheit aufzugeben

```
Map<CustomerID, Map<OrderID, String>> map = new HashMap<>();
List<Person> persons = new ArrayList<>();
var map = new HashMap<CustomerID, Map<OrderID, String>>();
var persons = new ArrayList<Person>();
```

- nur für lokale Variablen, loop Variablen, try-with-resource Variablen

```
for (Map.Entry<CustomerID, Map<OrderID, String>> entry :
map.entrySet()) { … }
for (var entry : map.entrySet()) {
```

- Gültigkeitsbereich sollte minimal bleiben um Fehler zu vermeiden

Java 10
JEP 310 – Application CDS

- Class-Data Sharing wird nun auch anderen Classloadern ermöglicht
- kürzere Startzeit der JVM, laden und verifizieren von (unveränderten) Klassen wird übersprungen
- geringerer Speicherverbrauch, mehrere laufende JVMs können dieses Abbild gemeinsam nutzen

Java 10
JEP 313 – Remove javah

- Native-Header Generation Tool wurde entfernt, Funktionalität wurde bereits in Java 8 durch javac übernommen

```
javac -d target/ HelloJNI.java
javah -jni -d header/ -cp target/ HelloJNI

javac -h header/ -d target/ HelloJNI.java
```

Java 11
JEP 320 – Remove Java EE/Corba

- in Java 9 bereits als “deprecated for removal” markiert

```
java.xml.ws (JAX-WS, also SAAJ and Web Services Metadata)
java.xml.bind (JAXB)
java.activation (JAF)
java.xml.ws.annotation (Common Annotations)
java.corba (CORBA)
java.transaction (JTA)
java.se.ee (Aggregator module for the six modules above)
jdk.xml.ws (Tools for JAX-WS)
jdk.xml.bind (Tools for JAXB)
```

- ebenso dazugehörige Tools

```
wsgen, wsimport (jdk.xml.ws)
schemagen, xjc (jdk.xml.bind)
idlj, orbd, servertool, tnameserv (java.corba)
```

Java 11
JEP 328 – Flight Recorder

- bislang eine kommerzielle Funktion vom Oracle JDK
- geringer Laufzeit-Overhead
- zeichnet Eventdaten der Anwendung, JVM und dem Betriebssystem auf
- Aufzeichnung kann über verschiedene Wege gestartet werden

```
remote über JMX, z.B. Mission Control

java -XX:StartFlightRecording …

$ jcmd <pid> JFR.start
$ jcmd <pid> JFR.dump filename=recording.jfr
$ jcmd <pid> JFR.stop
```

Java 11
JEP 335 – Deprecate Nashorn JSE

- nur die EMAScript Implementierung ist betroffen, nicht die javax.script API
- als “deprecated for removal” markierte Module

```
jdk.scripting.nashorn
jdk.scripting.nashorn.shell (jjs)
```

- JavaScript Implementierung der Graal VM könnte ein Ersatz werden

Java 11
JEP 336 – Deprecate Pack200

- Gründe der Einführung in Java 5 sind nicht mehr gegeben, z.B. Downloadgeschwindigkeit
- hoher Aufwand diese komplexe Technologie zu pflegen, class- und JAR-Dateiformat haben sich seit Einführung geändert, nachteilige Auswirkung z.B. bei der Modularisierung in Java 9 (4 APIs wurden entfernt)
- die Tools pack200, unpack200 und jar geben bei Verwendung einen Warnung aus

Appendix

- Nicolai Parlog über neuen Releasezyklus
  https://heise.de/-4165009
- Oracle JDK Release Notes
  https://www.oracle.com/technetwork/java/javase/jdk-relnotes-index-2162236.html
- Oracle Java SE Support Roadmap
  https://www.oracle.com/technetwork/java/java-se-support-roadmap.html