JAVA 18

Java 17 LTS (Long Term Support) is nu uit en we zien een grote beweging van Java SE 1.8 (8) naar Java SE 11 en zelfs naar 17, omdat de drempel tussen Java SE 11 en 17 relatief klein is. We hebben er overigens al een paar versies aan mogen wennen. Mark Reinhold (Chief Architect, Java Platform Group, Oracle) heeft aangekondigd dat Oracle nog sneller wil gaan bewegen naar nieuwe LTS-versies. Sinds Java SE 10 wordt er een cyclus van drie jaar gehanteerd tussen LTS-versies en een releasecyclus van zes maanden tussen elke release. Oracle wil nu naar twee jaar tussen LTS-versies [2] en de halfjaarlijkse releases behouden. In Java SE 18 staan 9 features (JEP's) gepland.

Om met wat Java SE 18 features te kunnen spelen, zonder de early access echt te hoeven installeren, is alle code in dit artikel uitgevoerd binnen een Docker container met OpenJDK 18 (zie Listing 1 en referentie 3).

L1

```
$ docker run -it --rm \
      -v $(pwd)/src/main/java:/src \
      openjdk:18-slim /bin/bash
$ cd /src
```

Dit artikel is in drie hoofdsecties verdeeld. Het eerste deel gaat over nieuwe 'standaard features'. Het tweede deel gaat in op 'preview en incubator features' en het derde deel gaat in op het 'deprecated verklaren of verwijderen van features'. Bij elke feature wordt het JEP-nummer vermeld.

{ NIEUWE STANDAARD FEATURES }

400: UTF-8 by Default

Veel methoden in de standaard API zijn afhankelijk van een standaardtekenset die op verschillende manieren kan worden geconfigureerd. Deze JEP is bedoeld om de standaard tekenset UTF-8 te maken. Dit wordt bereikt door twee waarden van de systeemeigenschap `file.encoding`. Als deze is ingesteld op `COMPAT`, wordt de standaard tekenset gevonden zoals in JDK 17 en eerder. Als deze is ingesteld op `UTF-8`, is de standaard tekenset UTF-8.

Ivo Woltring is Principal Expert en Codesmith bij Ordina JTech en houdt zich graag bezig met nieuwe ontwikkelingen in de software wereld.

408: Simple Web Server

Om de JDK toegankelijker te maken, wordt een eenvoudige out-of-the-box static webserver meegeleverd voor het serveren van statische content (Listing 2).

Commandline:

```
$ jwebserver -b 0.0.0.0 -p 8000
$ jwebserver --help
```

Java code:

L2

```
**
 * A Simple Static WebServer
 * <br/>
 * Snippet referenced in the code:
 * {@snippet class="JEP408SimpleWebServer"
   region="example"}
 */
public class JEP408SimpleWebServer {
    public static void main(String[] args) {
        // @start region="example"
        var server = SimpleFileServer.
        createFileServer(
          new InetSocketAddress(8000), // @replace
          regex="8000" replacement="PORT_HERE"
          Path.of("/src"), // @replace regex="/
          src" replacement="PATH_HERE"
          OutputLevel.VERBOSE);  // @link
          regex="OutputLevel" target="SimpleFileServer"
        server.start();
        // @end
        System.out.println( "Started: http://
        localhost:8000" );
    }
}
$ java JEP408SimpleWebServer.java
Started: http://localhost:8000
172.17.0.1 - - [05/Mar/2022:17:56:13 +0000] "GET /
HTTP/1.1" 200 -
```

413: Code Snippets in Java API Documentation

`{@snippet}` in Javadoc is een verbetering op de `@code` of the `<pre></pre>` blocks in Javadoc. Het is configureerbaar en kent vele opties. Na het genereren van de Javadoc boven de `JEP-408SimpleWebServer` class (Listing 1) krijg je het volgende resultaat (Listing 3 en Afbeelding 1):

L3

```
$ javadoc --snippet-path . -d ./doc
JEP408SimpleWebServer.java
$ jwebserver -b 0.0.0.0 -p 8000
```

416: Reimplement Core Reflection with Method Handles

Op dit moment gebruiken verschillende delen van `java.lang.reflect` verschillende mechanismen. Deze zullen opnieuw worden geïmplementeerd om het onderhoud te vereenvoudigen.

418: Internet-Address Resolution SPI

De API `java.net.InetAddress` zet op dit moment de hostnamen om naar IP-adressen en omgekeerd, door gebruik te maken van de OS native domeinnaamresolver. Deze JEP definieert een serviceproviderinterface (SPI) voor hostnaam- en adresresolutie, zodat `java.net.InetAddress` gebruik kan maken van andere resolvers dan de ingebouwde resolver van het platform.

{ PREVIEW EN INCUBATOR FEATURES }

417: Vector API (Third Incubator)

Deze JEP introduceert een API om vector berekeningen te compileren tot optimale vectorinstructies op de ondersteunde CPU-architecturen. Deze derde incubator fase richt zich vooral op verbeteringen uit feedback en op verbeterde implementatie en performance. Deze JEP bouwt voort op JEP 414 uit Java SE 17 en JEP 338 die in Java SE 16 is geintroduceerd. Zie referentie [3] voor voorbeeldcode.

419: Foreign Function & Memory API (Second Incubator)

Dit is de voortzetting van het werk dat is begonnen met JEP 370 in Java SE 14 en JEP 383 voor toegang tot Foreign Memory en JEP 389 Foreign Linker API. Het biedt een manier om te werken aan geheugen buiten de Java runtime (dus buiten de garbagecollector) en biedt een alternatief voor Java Native Interface (JNI). Dit item is uitgebreid behandeld in het artikel over Java SE 14 [4].

420: Pattern Matching for switch (Second Preview)

Dit is de tweede preview van pattern matching voor switch-statements die voor het eerst is uitgebracht in Java 17. In deze review zijn vooral kleine verbeteringen aangebracht aan de hand van gebruikers feedback. Zie referentie [3] voor voorbeeldcode.

Afbeelding 1

{ DEPRECATED EN VERWIJDERDE FEATURES }

421: Deprecate Finalization for Removal

'Finalization' in Java wordt deprecated verklaard. Het zal nog steeds ingeschakeld zijn, maar het kan volledig worden uitgeschakeld met de commandline optie `--finalization=disabled`. Dit omvat alle finalize()-methoden, dus ook de finalize in try-catch-finally-blokken. In plaats van deze verouderde manier van resources vrij maken te gebruiken, zouden we voortaan de try-with-resources moeten gebruiken of Cleaners [5].

{ CONCLUSIE }

Java beweegt duidelijk mee met wat de markt wil en nodig heeft. Ook wordt er steeds vaker nagedacht over wat handig is voor ontwikkelaars, zoals de out-of-the-box static webserver en de snippets. Ook het langzaam maar zeker opruimen van obsolete features of features die niet stabiel zijn gebleken, kan de taal alleen maar goed doen. Ik ben zeer benieuwd of we ook werkelijk vaker LTS-releases zullen krijgen.

{ REFERENTIES }

1 https://openjdk.java.net/projects/jdk/18/
2 https://mreinhold.org/blog/forward-even-faster
3 http://ivo2u.nl/Vy
4 http://ivo2u.nl/oz
5 https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/ref/Cleaner.html