Java 13

Welke nieuwe features zijn geïmplementeerd?

Oracle heeft sinds Java SE 10 een zes-maanden-release-schema aangekondigd. De deadline voor nieuwe features in Java is fixed gemaakt. Dit betekent dat het aantal JDK Enhancement Proposals (JEP) dat geïmplementeerd kan worden per release variabel is geworden. Eén keer in de drie jaar zal een zogenaamde Long-term support (LTS) versie worden gemaakt. Java SE 11 is zo'n LTS versie en zal door Oracle nog tot september 2023 ondersteund worden. Dit zou betekenen dat Java 17 ook weer een LTS wordt. De versies die tussen LTS versies uitkomen, krijgen alleen support tot een nieuwe versie uitkomt. Dit artikel beschrijft welke nieuwe features in Java 13 geïmplementeerd zijn.

Voor Java 13 zijn er zijn nieuwe features (JEP's) geïmplementeerd. Om met wat Java 13 features te kunnen spelen, zonder de early access echt te hoeven installeren, is alles uitgeprobeerd binnen een docker container met openjdk 13.

```
$ docker run -it --rm             \
      -v $(pwd)/src/main/java:/src \
      openjdk:13 /bin/bash
$ cd /src
```

JEP 350: Dynamic CDS Archives

Class Data Sharing is een JVM feature. Het maakt het mogelijk voor JVM's om geladen classes te delen via gedeeld geheugen. Het doel is een snellere opstarttijd van applicaties en minder geheugengebruik.

De feature is al beschikbaar sinds Java 1.5, maar het was een commerciële feature tot JDK 10. Origineel was de code in listing 1 nodig om van CDS gebruik te kunnen maken.

In Java 12 is JEP 341 geïmplementeerd. Met die implementatie worden CDS archives standaard gegenereerd en krijg je "out-of-the-box" betere opstarttijden.

Deze JEP 350 voor Java 13 gaat nog een stap verder in het makkelijker maken om met CDS te werken. Het hoofddoel is verbetering van de bruikbaarheid van het delen van Class Data Sharing (AppCDS). Het elimineert ook de noodzaak voor gebruikers om proefruns uit te voeren om een class list voor elke toepassing te maken. Een CDS archive kan gegenereerd worden als de applicatie stopt met `-XX:ArchiveClassesAtExit=<filename>` en gebruikt worden door de applicatie te draaien met `-XX:SharedArchiveFile=<filename>`. Dit gebeurt dan bovenop de system default archive.

JEP 351: ZGC: Uncommit Unused Memory

ZGC staat voor Z Garbage Collector en is geïntroduceerd in Java 11. Het is een schaalbare Garbage Collector met lage latentie. Tot nu toe gaf de ZSG ongebruikt heapgeheugen nog niet terug aan het Operating Systeem. Deze JEP verhelpt dat probleem.

JEP 353: Reimplement the Legacy Socket API

De implementaties van java.net.Socket en java.net.ServerSocket zijn redelijk oud en deze JEP levert een moderne implementatie.

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

```
# Eerst het creëren van een lijst van classes
java -XX:+UseAppCDS
     -XX:DumpLoadedClassList=classes.lst
     -jar app.jar

# Dan het archive genereren
java -XX:+UseAppCDS -Xshare:dump
     -XX:SharedClassListFile=classes.lst
     -XX:SharedArchiveFile=app-cds.jsa
     --class-path app.jar

# En dan het gebruik van het archive
java -XX:+UseAppCDS -Xshare:on
     -XX:SharedArchiveFile=app-cds.jsa
     -jar app.jar
```

Listing 1.

De moderne implementatie zal de standaard zijn in Java 13, maar de oude implementaties zullen nog niet weggehaald worden. Deze kunnen nog geactiveerd worden met system property: `jdk.net.usePlainSocketImpl`
Het doel is gemakkelijker onderhoud en gemakkelijker debuggen.

JEP 354: Switch Expressions (Preview)

Switch Expressions zijn in java 12 geïntroduceerd als preview feature. Deze JEP maakt aanpassingen in deze feature. Het introduceert het `yield` statement om een returnwaarde te geven in plaats van de eerst voorgestelde manier via de `break`. Er wordt nu dus onderscheid gemaakt tussen een switch expressie (met return waarde) en een switch statement (zonder returnwaarde). Het is de bedoeling dat de switch expressie yield gaat gebruiken en het switch statement break.

In listing 2 wordt de switch gedemonstreerd.

Aangezien het hier om een preview feature gaat, moet deze aangezet worden in de java en javac commando's.

```
javac -source 13 --enable-preview
App.java &&\
java --enable-preview App NLJUG
```

Met als output:

```
Note: App.java uses preview language
features.
Note: Recompile with -Xlint:preview
for details.
rockz
Yeah
```

JEP 355: Text Blocks (Preview)

Deze JEP introduceert Text Blocks. Het is een "multi-line literal" met als doel strings toe te staan die meerdere regels bevatten, zonder het helemaal te hoeven escapen. De output moet op een voorspelbare manier geformatteerd zijn. Code in listing 3, uitvoer in listing 4. Opvallend is dat de extra whitespace die je wel in de code ziet, weg is bij het printen naar de output.

Conclusie

Java 13 heeft niet echt zichtbare features voor ontwikkelaars, tenzij je bereid bent om de preview features te activeren. Wel zijn weer een aantal goede optimalisatieslagen gedaan. Laten we hopen dat de laatste twee JEP's snel evolueren uit de preview en mainstream worden in Java 14! ■

```
public class App {
    public static void main(String[] args) {
        System.out.println(switch (args[0]) {
            case "NLJUG" -> "rockz";
            case "Foo" -> "Bar";
            default -> {
                int len = args[0].length();
                yield len;
            }
        });
        System.out.println(switch (args[0]) {
                case "Ivo": yield "Woltring";
                case "NLJUG": yield "Yeah";
                default: {
                        yield "echoing: " + args[0];
                }
        });
    }
}
```

Listing 2.

```
public class MultiLineStrings {
public class MultiLineStrings {
    public static void main(String[] args) {
        String niet = "<html>\n" +
                      "    <body>\n" +
                      "        <p>Hello, world</p>\n" +
                      "    </body>\n" +
                      "</html>\n";
        String maar = """
                      <html>
                          <body>
                              <p>Hello, world</p>
                          </body>
                      </html>
                      """;
        System.out.println(maar);

        String query = """
                       SELECT `EMP_ID`,
                              `LAST_NAME` FROM `EMPLOYEE_TB`
                       WHERE `CITY` = 'INDIANAPOLIS'
                       ORDER BY `EMP_ID`, `LAST_NAME`;
                       """;
        System.out.println(query);
    }
}
```

Listing 3.

```
$ javac -source 13 --enable-preview MultiLineStrings.java && \
java --enable-preview MultiLineStrings
Note: MultiLineStrings.java uses preview language features.
Note: Recompile with -Xlint:preview for details.
<html>
    <body>
        <p>Hello, world</p>
    </body>
</html>

SELECT `EMP_ID`,
       `LAST_NAME` FROM `EMPLOYEE_TB`
WHERE `CITY` = 'INDIANAPOLIS'
ORDER BY `EMP_ID`, `LAST_NAME`;
```

Listing 4.

REFERENTIES:

- **https://openjdk.java.net/pro jects/jdk/13/**
- **http://ivo2u.nl/oz**
- **https://www.oracle.com/technetwork/java/java-se-support-roadmap.html**