Ausgabe 05/2020

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

D: 4,90 EUR | CH: 9,80 CHF | A: 5,60 EUR | Benelux: 5,80 EUR

4 191978 304903 05

Java aktuell

iJUG Verbund
www.ijug.eu

Microservices
Performancetests, Service Meshes, MicroProfile GraphQL und mehr

Javas Geheimnisse
Weniger bekannte Features und Eigenheiten

Deep Learning
Einblick in das Trend-Thema

Klein aber oho: MICROSERVICES

JavaLand

16. - 18. März 2021
in Brühl bei Köln

Save the Date

Hybride Veranstaltung

Was die JavaLand als Plattform für Wissentransfer und Networking ausmacht, kannst du im Phantasialand oder online erleben. Als Teilnehmer entscheidest du selbst, welche Variante du wählen möchtest.

Präsentiert von: DOAG Heise Medien

Community Partner: iJUG Verbund

www.javaland.eu

Das Glas ist also endlich wieder halb voll, und ich freue mich schon darauf, meine ersten Anwendungen auf Jakarta EE 9 zu portieren!

Referenzen

[1] Aktueller Jakarta EE 9 Release Plan: *https://eclipse-ee4j.github.io/jakartaee-platform/jakartaee9/JakartaEE9#jakarta-ee-9-schedule*

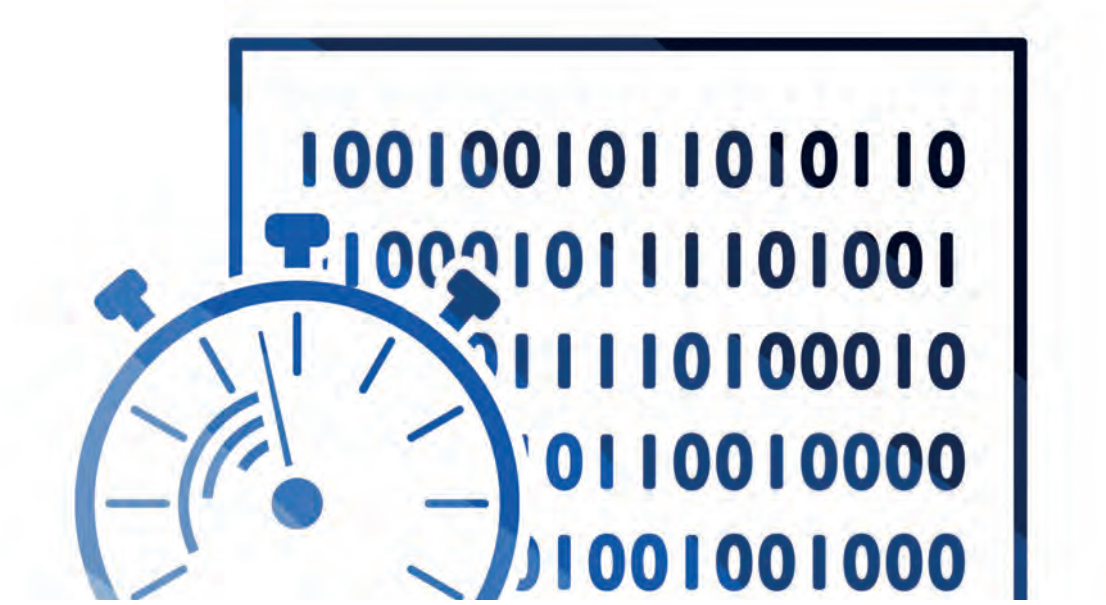
[2] Ursprünglicher Jakarta EE 9 Release Plan: *https://eclipse-ee4j.github.io/jakartaee-platform/jakartaee9/JakartaEE9ReleasePlan*

[3] JVM 8/11 Performance-Vergleich: *https://blog.qfotografie.de/2019/01/21/java-8-vs-java-11-benchmark-a-productive-business-application/*

Markus Karg

markus@headcrashing.eu

Markus Karg ist Entwicklungsleiter eines mittelständischen Softwarehauses sowie Autor, Konferenzsprecher und Consultant. JAX-RS hat der Sprecher der Java User Group Goldstadt von Anfang an mitgestaltet, zunächst als freier Contributor, seit JAX-RS 2.0 als Mitglied der Expert Groups JSR 339 und JSR 370.



Unbekannte Kostbarkeiten des SDK Heute: Temporäre Dateien und Verzeichnisse

Bernd Müller, Ostfalia

Das Java SDK enthält eine Reihe von Features, die wenig bekannt sind. Wären sie bekannt und würden sie verwendet, könnten Entwickler viel Arbeit und manchmal sogar zusätzliche Frameworks einsparen. Wir wollen in dieser Reihe derartige Features des SDK vorstellen: die unbekannten Kostbarkeiten.

```
public static Path createTempFile(String prefix, String suffix, FileAttribute<?>... attrs)

public static Path createTempFile(Path dir, String prefix, String suffix, FileAttribute<?>... attrs)

public static Path createTempDirectory(String prefix, FileAttribute<?>... attrs)

public static Path createTempDirectory(Path dir, String prefix, FileAttribute<?>... attrs)
```

Listing 1

Von Zeit zu Zeit haben wir die Anforderung, Dateien zu erzeugen, beispielsweise um Zwischenergebnisse abzuspeichern oder Dateifunktionen des Betriebssystems verwenden zu können. Häufig sind die Dateien nach ihrer Verwendung wieder zu löschen. Java kann uns das Erzeugen der Dateien zwar nicht abnehmen, aber beim Finden noch nicht verwendeter Namen helfen. Das Löschen der Dateien kann dagegen mehr oder weniger automatisiert werden. Dasselbe gilt für Verzeichnisse und damit sukzessive für Dateien in diesen temporären Verzeichnissen. Den entsprechenden Methoden gilt die Aufmerksamkeit unserer heutigen unbekannten Kostbarkeiten.

Die Klasse Files

Mit Java 7 wurde das Package `java.nio.file` eingeführt, das unter anderem die Klasse `Files` enthält. Mit Java 8, 11 und 12 bekam diese Klasse eine ganze Reihe neuer Methoden spendiert, die wir bereits in den *unbekannten Kostbarkeiten 06/2019* gewürdigt haben. Heute gehen wir noch weiter zurück und schauen uns Methoden an, die bereits in der Ursprungsversion der Klasse `Files` mit Java 7 Einzug hielten; sie beginnen alle mit dem Präfix `createTemp` und sind in *Listing 1* dargestellt. Beginnen wir mit der Methode `createTempFile()`. Die in *Listing 2* gezeigten Code-Zeilen erzeugen eine Datei mit einem zufälligen Namen und registrieren deren Löschung bei Programmende.

Der Dateiname hat das Präfix „`pre-`" und die Extension „`.txt`". Der mittlere und generierte Teil ist eine Ziffernfolge, die als Long-Zufallszahl erzeugt wird. Das Interessante ist das Registrieren einer automatischen Löschung bei Terminierung der JVM mit der Methode `File.deleteOnExit()`. Wir zitieren deren JavaDoc:

„Requests that the file or directory denoted by this abstract pathname be deleted when the virtual machine terminates. Files (or directories) are deleted in the reverse order that they are registered. Invoking this method to delete a file or directory that is already registered for deletion has no effect. Deletion will be attempted only for normal termination of the virtual machine, as defined by the Java Language Specification. Once deletion has been requested, it is not possible to cancel the request. This method should therefore be used with care."

Unter einem normalen Ende ist hier das Programmende oder ein CTRL-C auf Betriebssystemebene gemeint. Ein „`kill -9 <pid>`" gehört nicht dazu.

Die anderen Methoden des Listings 1 sind analog zu verwenden. Optional können ein Verzeichnis oder Dateiattribute angegeben werden, die jedoch nichts mit dem temporären Charakter der Erzeugnisse zu tun haben und daher von uns nicht näher erläutert werden.

Abschließend bleibt noch zu klären, wo die temporären Dateien und Verzeichnisse erzeugt werden. Das Elternverzeichnis der

```
Path tempFile = Files.createTempFile("pre-", ".txt");
tempFile.toFile().deleteOnExit();
```

Listing 2

temporären Dateien und Verzeichnisse wird durch das System-Property `java.io.tmpdir` festgelegt. Beim Linux-System des Autors ist dies `/tmp`.

Das automatisierte Löschen von temporären Dateien kann alternativ auch über Shutdown-Hooks realisiert werden. Da Shutdown-Hooks aber ganz allgemein eingesetzt werden können, sind sie eine eigene *unbekannte Kostbarkeit.*

Funktioniert das überall?

Wir haben das Beispiel mit OpenJDK 14, GraalVM 20 Native Image und WildFly 19 getestet und können die Funktionsfähigkeit bestätigen. Lediglich der Wert des System-Property im Native Image ist `/var/tmp`.

Zusammenfassung

Die Klasse `Files` erlaubt das Erzeugen temporärer Dateien, also Dateien, deren Namen generiert werden. Sie ermöglicht es, diese Dateien bei Programmende automatisch zu löschen, indem sie mit einem einfachen Methodenaufruf für diese Löschung registriert werden.

Bernd Müller
Ostfalia
bernd.mueller@ostfalia.de

Nach seinem Studium der Informatik und der Promotion arbeitete Bernd Müller für die IBM und die HDI Informationssysteme. Er ist Professor, Geschäftsführer, Autor mehrerer Bücher zu den Themen JSF und JPA, sowie Speaker auf nationalen und internationalen Konferenzen. Er engagiert sich im iJUG und speziell in der JUG Ostfalen.

2. + 3. DEZEMBER 2020

BERLINER EXPERTENSEMINAR

DOAG

NICOLAI PARLOG

Java after eight

KURSÜBERBLICK

In diesem Kurs werden die Java-Versionen 9 bis 15 beleuchtet. Dabei liegt der Fokus auf neuen Sprachfeatures wie Sealed Classes, Records, Text Blocks, switchExpressions und var, aber auch neue und erweiterte APIs sowie JVM- und Performance-Verbesserungen werden theoretisch vorgestellt und mit praktischen Übungen untermauert. Darüber hinaus werden der Migrationspfad von Java 8 zu 11 bzw. 15, der neue Release-Zyklus und die aktuelle Situation um Distributionen und Support besprochen.

ZIELGRUPPE

Erfahrene Java-Entwickler, die sich theoretisch und praktisch auf die neuen Java-Versionen vorbereiten möchten.

VORAUSSETZUNGEN

Einige Jahren praktische Erfahrung mit Java-Entwicklung und sicherere Java-8-Kenntnisse.

www.doag.org/go/expertenseminar_parlog