2-2013 | Sommer | www.ijug.eu

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

Java aktuell

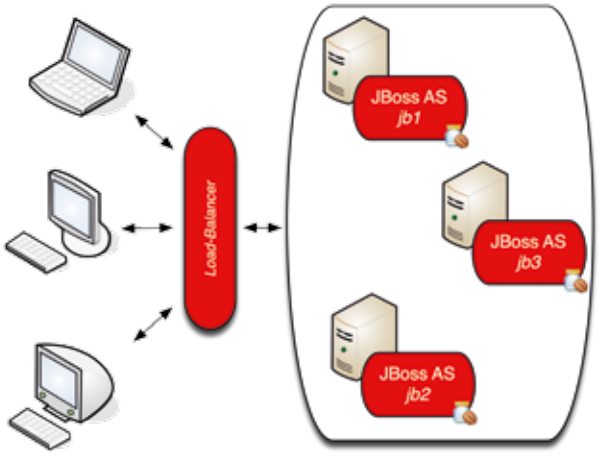
Java verbreitet sich überall

D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

iJUG Verbund

Sonderdruck

4 191978 304903 02



Lastverteilung zum Erreichen der Hochverfügbarkeit, Seite 20

Build-Kreislauf beim Continuous Integration, Seite 59

Unbekannte Kostbarkeiten des SDK
Heute: Der ZIP-File-System-Provider

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 stellen in dieser Reihe derartige Features des SDK vor: die unbekannten Kostbarkeiten.

Mit Java SE 7 wurde die zweite Version des neuen Ein-/Ausgabesystems (NIO.2) eingeführt. Es abstrahiert von realen Dateisystemen und stellt Klassen und Methoden für das Erzeugen, Kopieren, Verschieben, Umbenennen und Löschen von Dateien bereit. Als Standard-Dateisysteme werden die typischen Java-Plattformen, also Windows und Unixoide, unterstützt. Zusätzlich werden jedoch auch ZIP-Archive als Dateisysteme behandelt und deren Dateien sind unter demselben API wie gewöhnliche Dateien verfügbar, was die Handhabung von ZIP-Archiven im Vergleich zu den bisherigen Möglichkeiten stark vereinfacht.

ZIP-Dateien in Java

ZIP-Dateien sind durch die Klasse „ZipFile" im Package „java.util.zip" repräsentiert. Die Klasse „ZipEntry" und verschiedene Klassen für In- und Output-Streams vervollständigen das Package. JAR-Files werden durch die Klasse „JarFile", eine direkte Unterklasse von „ZipFile", im Package „java.util.jar" repräsentiert. Auch hier vervollständigen weitere Klassen in diesem Package die Handhabung von JAR-Dateien.

Als kleines Beispiel realisieren wir eine Methode zum Lesen einer Datei innerhalb einer ZIP-Datei. Die Methode liefert für den gegebenen ZIP-Dateinamen und den Eintrag (ZIP-Entry) innerhalb der ZIP-Datei einen „BufferedReader", mit dem der Inhalt des ZIP-Entry gelesen werden kann (siehe Listing 1). Typisch für diese Art der Verwendung ist die Suche nach dem ZIP-Entry durch eine Iteration über alle ZIP-Entries der Datei.

Der ZIP-File-System-Provider

Mit Java 7 wurde der sogenannte „ZIP-File-System-Provider" eingeführt. Auf seine Entstehungsgeschichte und Implementierung gehen wir später kurz ein. Zunächst soll das obige Beispiel mit dessen Hilfe reimplementiert werden. Die folgende Methode zeigt diese Implementierung, wobei die Verwendung allgemeiner Klassen und Methoden von NIO.2 und der völlige Verzicht auf ZIP- beziehungsweise JAR-spezifische Klassen und Methoden hervorzuheben ist (siehe Listing 2).

Die Methode bekommt im ersten Parameter „zipFile" den Namen der ZIP-Datei und im zweiten Parameter „entryPath" den Pfad des ZIP-Eintrags übergeben und ist damit Signatur-identisch zur ursprünglichen Methode. Die Methode „URI.create()" erwartet im Parameter zunächst ein Schema mit anschließendem schemaspezifischem Anteil. Im Falle des ZIP-Filesystem-Providers muss dieser hierarchisch sein, sodass der Name der ZIP-Datei mithilfe der Klasse „Path" zunächst in diese Form umgewandelt wird.

Die Fabrikmethode „FileSystems.getFileSystem()" verwendet das Schema des URI, um ein passendes File-System zurückzuliefern. Im Beispiel erhält man das Dateisystem des ZIP-Archivs. Der zweite Parameter der Methode sind dateisystemspezifische Properties, die, wie in unserem Beispiel, auch leer sein dürfen. Zuletzt wird mit „Files.newBufferedReader()" das Methoden-Ergebnis erzeugt. Im Vergleich zur ursprünglichen Implementierung entfällt die Suche nach dem ZIP-Eintrag und die Realisierung erscheint insgesamt einfacher und konsistenter.

Das Erzeugen eines ZIP-Archivs

Das Erzeugen eines ZIP-Archivs mithilfe des ZIP-File-System-Providers ist ebenfalls deutlich einfacher als bisher und verwendet ausschließlich Standard-APIs (siehe Listing 3).

Die Methode bekommt die zu kopierende Datei (toCopy), den Namen des ZIP-Archivs (zipFile), das Verzeichnis im ZIP-Archiv (entryDir) sowie den im ZIP-Archiv zu verwendenden Dateinamen (entryFile) übergeben. Im Gegensatz zum ersten Beispiel ist der zweite Parameter für „FileSystems.newFileSystem()" nicht leer, sondern enthält für den Schlüssel „create" den Wert „true". Damit wird das ZIP-Archiv angelegt, falls es noch nicht existiert. Der Default-Wert ist „false" und wurde somit implizit im ersten Beispiel verwendet. Die Syntax entspricht dem Double-Brace-Initialization-Pattern, das wir in den verborgenen Kostbarkeiten in der Java aktuell 3/2012 [1] vorgestellt haben. Die weiteren Methoden-Aufrufe sind selbsterklärend.

Entwicklungsgeschichte und Implementierung

Laut [2] wurde der ZIP-File-System-Provider als Demonstrator-Projekt der Möglichkeiten von NIO.2 begonnen. Nach der Fertigstellung wurde beschlossen, ihn in das JDK 7 zu übernehmen. Der Quell-Code ist im JDK 7 im Unterverzeichnis „demo/nio/zipfs" einzusehen. Der Provider wird dem Laufzeitsystem über die Service-Provider-Schnittstelle bekannt gegeben, die wir in den verborgenen Kostbarkeiten in Java aktuell 4/2011 [3] beschrieben haben. Das zu implementierende Interface ist FileSystemProvider im Package „java.nio.file.spi". Eine Einführung in den ZIP-File-System-Provider findet man in [4]. Falls ein eigener File-System-Provider realisiert werden soll, hilft [5] weiter.

Fazit

Mit dem SDK 7 wurde der ZIP-File-System-Provider eingeführt. Er realisiert über das Standard-Dateisystem-API des NIO.2 den Zugriff auf ZIP-Archive und erleichtert damit den Umgang mit solchen Archiven erheblich.

Weitere Informationen

[1] Bernd Müller. Unbekannte Kostbarkeiten des SDK – Heute: Double Brace Initialization und Instance Initializier. Java aktuell 3/2012.

[2] Xueming Shen. The ZIP filesystem provider in JDK 7: https://blogs.oracle.com/xuemingshen/entry/the_zip_filesystem_provider_in1

[3] Bernd Müller. Unbekannte Kostbarkeiten des SDK – Heute: Der Service-Loader. Java aktuell 4/2011

[4] Oracle, Zip File System Provider: http://docs.oracle.com/javase/7/docs/technotes/guides/io/fsp/zipfilesystemprovider.html

[5] Oracle, Developing a Custom File System Provider: http://docs.oracle.com/javase/7/docs/technotes/guides/io/fsp/filesystemprovider.html

Bernd Müller
bernd.mueller@ostfalia.de

Bernd Müller ist Professor für Software-Technik an der Ostfalia. Er ist Autor des Buches „JavaServer Faces 2.0" und Mitglied in der Expertengruppe des JSR 344 (JSF 2.2).


```
public BufferedReader newBufferedReaderForEntry(String zipFile,
                                    String entryPath) throws IOException {
    ZipFile zf = new ZipFile(zipFile);
    Enumeration<? extends ZipEntry> entries = zf.entries();
    while (entries.hasMoreElements()) {
        ZipEntry entry = entries.nextElement();
        if (entry.getName().equals(entryPath)) {
            return new BufferedReader(new InputStreamReader(zf.getInputStream(entry)));
        }
    }
    throw new IOException("ZIP-Entry not found");
}
```

Listing 1

```
public BufferedReader newBufferedReaderForEntry(String zipFile,
                                    String entryPath) throws IOException {
    Path path = Paths.get(zipFile);
    URI uri = URI.create("jar:file:" + path.toUri().getPath());
    Map<String, String> env = new HashMap<>();
    FileSystem zipFileSystem = FileSystems.newFileSystem(uri, env);
    return Files.newBufferedReader(zipFileSystem.getPath(entryPath), StandardCharsets.UTF_8);
}
```

Listing 2

```
public void copyIntoZip(String toCopy, String zipFile,
                String entryDir, String entryFile) throws IOException {
    Path path = Paths.get(zipFile);
    URI uri = URI.create("jar:file:" + path.toUri().getPath());
    Map<String, String> env = new HashMap<String, String>() {{
        put("create","true"); // DBI, siehe [1]
    }};
    FileSystem zipfs = FileSystems.newFileSystem(uri, env);
    Path external = Paths.get(toCopy);
    Files.createDirectory(zipfs.getPath(entryDir));
    Path pathInZipfile = zipfs.getPath(entryDir + "/" + entryFile);
    Files.copy(external, pathInZipfile);
    zipfs.close();
}
```

Listing 3

DevFest Vienna 2012 – es war ein voller Erfolg!

Am 10. und 11. November 2012 hat die Java Student User Group Wien (JSUG) gemeinsam mit der Google Developer Group Vienna ihre erste eigene Konferenz abgehalten. Am ersten Tag gab es zwölf Vorträge für die knapp 150 Teilnehmer, Unmengen an Pizza, Drinks, Süßigkeiten, T-Shirts, Gimmicks und sozialer Interaktion. Es folgte eine tolle Afterparty inklusive Gewinnspiel und Verlosung von Raspberry Pis, Arduino Duos und Büchern. Ein Android Birthday Hackaton am Folgetag schloss die erfolgreiche Veranstaltung ab. Das DevFest Vienna 2013 ist für Anfang Oktober 2013 geplant. *Kontakt: Dominik Dorn, dominik.dorn@jsug.at*

www.ijug.eu

JETZT ABO BESTELLEN

Sichern Sie sich 4 Ausgaben für 18 EUR

Für Oracle-Anwender und Interessierte gibt es das Java aktuell Abonnement auch mit zusätzlich sechs Ausgaben im Jahr der Fachzeitschrift DOAG *News* und vier Ausgaben im Jahr Business News zusammen für 70 EUR. Weitere Informationen unter www.doag.org/shop/

FAXEN SIE DAS AUSGEFÜLLTE FORMULAR AN

0700 11 36 24 39

ODER BESTELLEN SIE ONLINE

go.ijug.eu/go/abo

Interessenverbund der Java User Groups e.V.
Tempelhofer Weg 64
12347 Berlin

Java aktuell

+++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN

☐ **Ja**, ich bestelle das Abo Java aktuell – das IJUG-Magazin: 4 Ausgaben zu 18 EUR/Jahr

☐ **Ja**, ich bestelle den kostenfreien Newsletter: Java aktuell – der iJUG-Newsletter

ANSCHRIFT

Name, Vorname

Firma

Abteilung

Straße, Hausnummer

PLZ, Ort

GGF. ABWEICHENDE RECHNUNGSANSCHRIFT

Straße, Hausnummer

PLZ, Ort

E-Mail

Telefonnummer

X

Die allgemeinen Geschäftsbedingungen* erkenne ich an, Datum, Unterschrift

*Allgemeine Geschäftsbedingungen:

Zum Preis von 18 Euro (inkl. MwSt.) pro Kalenderjahr erhalten Sie vier Ausgaben der Zeitschrift "Java aktuell - das iJUG-Magazin" direkt nach Erscheinen per Post zugeschickt. Die Abonnementgebühr wird jeweils im Januar für ein Jahr fällig. Sie erhalten eine entsprechende Rechung. Abonnementverträge, die während eines Jahres beginnen, werden mit 4,90 Euro (inkl. MwSt.) je volles Quartal berechnet. Das Abonnement verlängert sich automatisch um ein weiteres Jahr, wenn es nicht bis zum 31. Oktober eines Jahres schriftlich gekündigt wird. Die Wiederrufsfrist beträgt 14 Tage ab Vertragserklärung in Textform ohne Angabe von Gründen.