04-2015 | Winter | www.ijug.eu

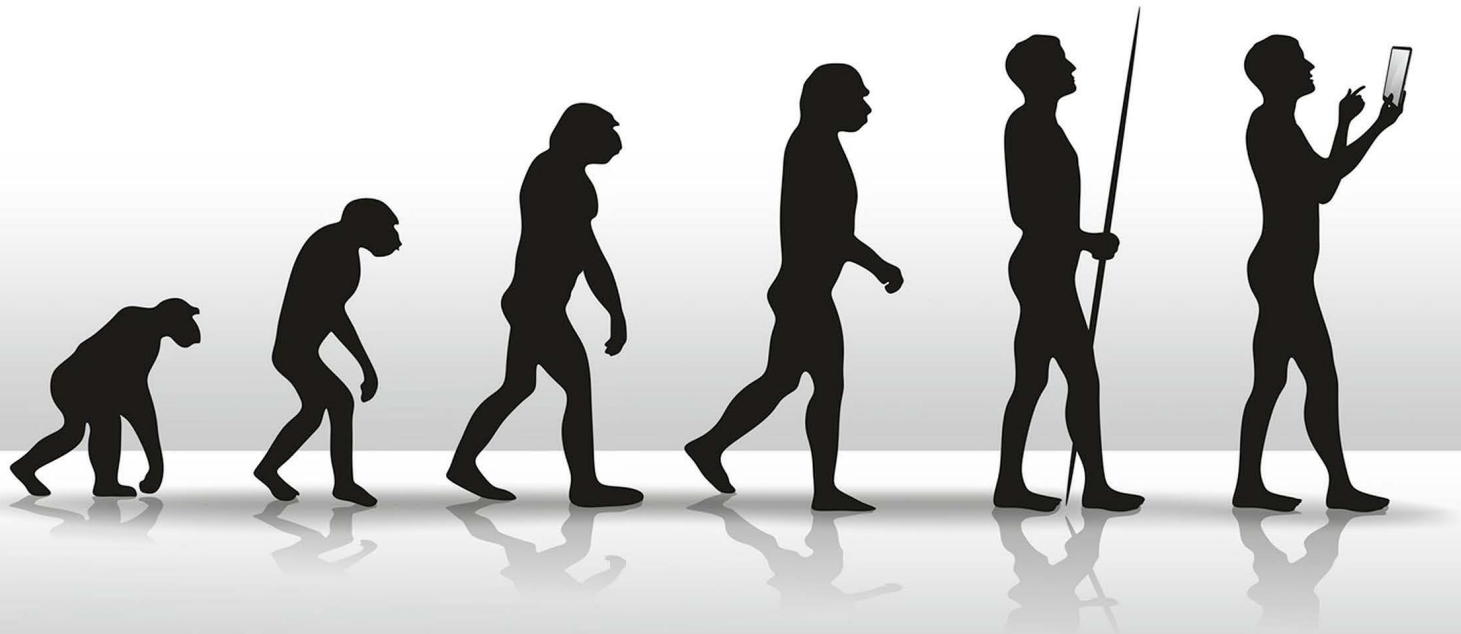
Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

Aus der Community – für die Community

Java aktuell

Java entwickelt sich weiter



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

Neue Frameworks
AspectJ, Eclipse Scout, Citrus

Raspberry Pi
Projekte mit Java

Java-Performance
Durch Parallelität verbessern

Web-Anwendungen
Hochverfügbar und performant

iJUG Verbund

4 191978 304903 04

Die Position „Software-Architekt" ist in der Software-Branche etabliert

Java-Champion Kirk Pepperdine gibt Performance-Tipps

Citrus bietet komplexe Integrationstests mit mehreren Schnittstellen

Unbekannte Kostbarkeiten des SDK

Heute: String-Padding

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.

Wie lassen sich Strings mit Leerzeichen zu einer Mindestlänge auffüllen? Wahrscheinlich 99 Prozent aller Java-Entwickler würden antworten: Mit den „StringUtils" aus Apache Commons Lang. Man kann dies aber auch sehr einfach mit SDK-Bordmitteln realisieren.

Motivation

Selbst in Zeiten von Micro-Services und JSON gibt es immer noch Anwendungen, die Textdateien austauschen. Nicht selten gehorchen diese einer Spalten-Struktur, sodass Strings links oder rechts mit Leerzeichen aufgefüllt werden müssen, um diese Spalten-Struktur zu erreichen. Die „StringUtils"-Klasse der Commons-Lang-Bibliothek stellt zur Realisierung die Methoden „leftPad()", „rightPad()" und „center()" zur Verfügung. Da Commons Lang eine sehr populäre Bibliothek ist, spricht nichts gegen die Verwendung, auch wenn andere Klassen und Methoden als die String-Padding-Methoden zum Einsatz kommen. Sollen jedoch lediglich Strings aufgefüllt werden, so kann dies alternativ auch mit SDK-Bordmitteln realisiert werden.

Der SDK-Formatter

Die Klasse „java.util.Formatter" definiert eine durchaus komplexe Formatierungs-Syntax, die an die Formatierungs-Syntax der Funktion „printf()" der Sprache C angelehnt ist. Wir wollen an dieser Stelle nur den kleinen Bereich des String-Padding beleuchten, raten aber, sich einmal etwas intensiver mit dieser Klasse beziehungsweise der darin beschriebenen Formatierungs-Syntax zu beschäftigen, da sie eventuell noch weitere interessante, aber noch unbekannte Möglichkeiten für andere Datentypen enthält.

Die grundlegende Syntax des Formatter ist „%[argument_index$][flags][width][.precision]conversion". Der „argument_index" wird verwendet, um im Formatierungs-String die Parameter über ihre Position in der Argumentliste zu referenzieren. Die Methode „String#format()" verwendet als ersten Parameter einen Term der obigen Syntax. Die weiteren Varargs-Parameter der Methode werden über den Argument-Index im Formatierungs-String ersetzt. Die Nummerierung beginnt bei „1", sodass „%1$" den ersten Parameter, „%2$" den zweiten etc. darstellt.

```
String.format("%1$10s", "hello")
String.format("%1$-10s", "hello")
```

Listing 1

```
public static String leftPad(String str, int size) {
  return String.format("%1$" + size + "s", str);
}

public static String rightPad(String str, int size) {
  return String.format("%1$-" + size + "s", str);
}

public static String center(String str, int size) {
  int padLeft = (size + str.length()) / 2;
  return rightPad(leftPad(str, padLeft), size);
}
```

Listing 2

```
@Test
public void equalWithApache() {
  Assert.assertEquals(StringUtils.leftPad(STRING, SIZE),
                      SdkPadding.leftPad(STRING, SIZE));
  Assert.assertEquals(StringUtils.rightPad(STRING, SIZE),
                      SdkPadding.rightPad(STRING, SIZE));
  Assert.assertEquals(StringUtils.center(STRING, SIZE),
                      SdkPadding.center(STRING, SIZE));
}
```

Listing 3

Das Flag („flags“) „-“ erzeugt eine linksbündige Ausgabe, der Default ist rechtsbündig. Die Breite („width2“) gibt die zu erreichende Gesamtlänge dieses Parameters an. Die Genauigkeit („precision“) ist nur für Gleitkommazahlen relevant. Die Konvertierungsoption („conversion“) bestimmt schließlich, welcher Art der Parameter ist. Für uns ist „s“ für Strings die zu verwendende Konvertierung.

Die Verwendung für ein String-Padding ist damit einfach zu realisieren. Die bereits erwähnte Methode „String#format()“ erwartet als ersten Parameter einen Formatierungs-String, gefolgt von „Varargs“-Parametern für die „$“-Parameter wie im folgenden Beispiel *(siehe Listing 1)*.

Beide Methoden-Aufrufe geben einen String der Länge „10“ zurück, wobei beim ersten Aufruf links von „hello“, beim zweiten rechts von „hello“ mit Leerzeichen aufgefüllt wird. Zu Demonstrationszwecken haben wir die drei „StringUtils“-Methoden „leftPad()“, „rightPad()“ und „center()“ mit dem Formatter implementiert (Klasse „SdkPadding“, siehe *Listing 2*) und ebenfalls gegen „StringUtils“ getestet *(siehe Listing 3)*.

Diese Tests haben bei uns für eine Auswahl von Parametern ein identisches Verhalten beider Implementierungen gezeigt. Einer Verwendung steht nichts mehr im Wege.

Fazit

Die Klasse „Formatter“ definiert eine Syntax für die Formatierung verschiedener Datentypen, unter anderem auch für Strings. Das Auffüllen („Padding“) von Strings ist damit einfach zu realisieren. Die Methode „String#format()“ und weitere Methoden wie „PrintStream#printf()“ verwenden diese Formatierungsmöglichkeit.

Alle unbekannten Kostbarkeiten

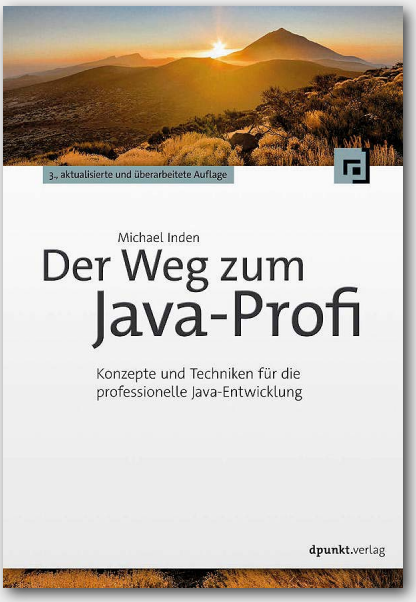
Bernd Müller veröffentlicht seit Jahren die "Unbekannten Kostbarkeiten des SDK" in der Java aktuell. Die früheren Beiträge sind auf seiner Website zu finden *(siehe „http://goo.gl/OJHajS“)*.

Bernd Müller
bernd.mueller@ostfalia.de

Bernd Müller ist Professor für Software-Technik an der Ostfalia (Hochschule Braunschweig/Wolfenbüttel). Er ist Autor mehrere Java-EE-Bücher, Sprecher auf nationalen und internationalen Konferenzen und engagiert sich in der JUG Ostfalen sowie im iJUG.

Der Weg zum Java-Profi

gelesen von Oliver Hock



Mit dem Buch „Der Weg zum Java-Profi“ findet der Autor Michael Inden einen Weg, der endlich da anknüpft, wo Ausbildung, Studium und Schulungen aufhören. Es gehört zwar eine Menge Mut dazu, sich diesen knapp 1.400 Seiten starken Band vorzunehmen, aber es ist ein Vorhaben, das schon bald Früchte tragen wird.

Der Inhalt dieses Buches schließt die Lücke zwischen dem Java-Beginner mit einigen Vorkenntnissen und dem absoluten Profi mit Expertenwissen, wobei die Intention auf dem Verständnis von Konzepten und dem praktischen Nutzen der Themen liegt. Dies wird unter anderem mit Code-Beispielen erreicht, die ergänzend zum Buch als kompilierbare Downloads verfügbar sind. Zudem sind Hintergrund-Informationen und Wissenswertes in entsprechenden Boxen besonders hervorgehoben.

Los geht es zunächst mit ein paar Java-Grundlagen. Getreu dem Motto „Was ich mich nie getraut habe zu fragen“ werden die Methoden „toString()“, „equals()“ und „hashCode()“ von „java.lang.Object“ beschrieben, ohne dabei in Sprach-Semantik oder gar Syntax-Details zu verfallen. Ferner behandelt dieses Kapitel allgemeine Datentypen, String- und Datumsverarbeitung sowie das Lesen und Schreiben von Dateien. Beson-