Daten aufbereiten mit JFreereport

Zahlen zeigen

Wer präzise Daten in Tabellen übersichtlich darzustellen hat, braucht intelligente Reporting-Software. Die JFreereport-Bibliothek unterstützt den Anwender bei der Eingabe sowie beim Druck und Export im Fremdformat. Mit wenigen Java-Zeilen entstehen so sauber strukturierte Berichte. Bernhard Bablok

Wer die neue Umsatzstatistik vor großem Publikum präsentiert, wird dafür kaum eine Ascii-Datei verwenden. Sie umfasst zwar alle Daten, ist aber total unübersichtlich. Ein buntes Diagramm wiederum ist sehr anschaulich, gerade fürs Erkennen von Tendenzen, mehr Details liefert aber ein Report, auf Deutsch: Bericht. Die folgende Definition mag nicht alle Sonderfälle abdecken, aber sie trifft den Kern: Ein Report ist die strukturierte Aufbereitung von Daten für Präsentation oder Druck.

Typischerweise unterstützt eine Reporting-Software nicht nur die reine Ausgabe, sondern auch einfache Operationen auf den Daten. Geht es zum Beispiel um nach Regionen und Produkten gegliederte Umsätze, ist es sinnvoll, dass das Reporting-Programm die Teilsummen innerhalb der Regionen über alle Produkte eigenständig bildet.

Reports eignen sich aber nicht nur für klassische Berichte wie die erwähnte Umsatzstatistik, sie können beispielsweise auch einen Onlinekauf dokumentieren oder technische Messdaten übersichtlich aufbereiten.

Übersichtliche Darstellung

So vielfältig wie die Einsatzgebiete sind die Anforderungen an die Layoutgestaltung. Die Übersichtlichkeit hängt von der Menge der Daten ab, das Aussehen auch von der Zielgruppe. JFreereport [1] gibt Java-Programmierern eine Bibliothek an die Hand, die eine einfache Architektur mit mächtigen Möglichkeiten für das Layout verbindet. Dieser Coffee-Shop gibt eine Einführung. Wer alle Feinheiten ausreizen will, kommt um eine intensive Beschäftigung mit den APIs, der Dokumentation oder sogar dem Quellcode nicht herum.

In der Version 0.8.4 ist das JFreereport-Paket 5,3 MByte groß und umfasst den Quellcode, die fertigen Jar-Dateien und alle nötigen externen Bibliotheken. Wie üblich entpackt man das Archiv etwa im Verzeichnis »/usr/local«:

```
tar -xzf jfreereport-0.8.4_9.tar.gz
 -C /usr/local
```

Es genügt, wenn sich die Jar-Bibliotheken zur Compile- und Laufzeit im »CLASSPATH« befinden. JFreereport ist in insgesamt 33 Jar-Dateien aufgeteilt, dazu kommen sechs externe Bibliotheken im »lib«-Verzeichnis, beispielsweise der GNU-Sax-Parser. Wer es sich einfach machen will, verwendet »jfreereport-0.8.4_9-all.jar«, die alle Jar-Dateien des Pakets enthält und deshalb 1,2 MByte groß ist. Zum Platzsparen sollte sie beim fertigen Programm durch das notwendige Minimum an Modulen ersetzt werden: Wer ohne grafische Oberfläche einen Report erstellt, kann auch die GUI-Klassen weglassen.

Eine Dokumentation bringt das Paket nicht mit. Auf der Homepage des Projekts findet sich allerdings eine 38-seitige Einführung im PDF-Format. Sie beschreibt zwar die 0.8.3-Version, genügt aber, um loszulegen. Ein besseres Handbuch wird künftig die Firma Object-Refinery verkaufen – die Vorabversion ist zurzeit noch frei verfügbar [2], weist aber einige Lücken auf. Die API-Dokumentation findet sich ebenfalls auf der Website, alternativ erzeugt sie das Make-Tool »ant«:

```
ant -f ant/build.xml javadoc
```

Zusätzlich zum Quellcode enthält das Paket viele Beispiele, die fast alle Möglichkeiten der Bibliothek demonstrieren. Aus diesen Vorlagen lassen sich leicht eigene Programme entwickeln. Um das JFreereport-API zu verstehen und die Bi-

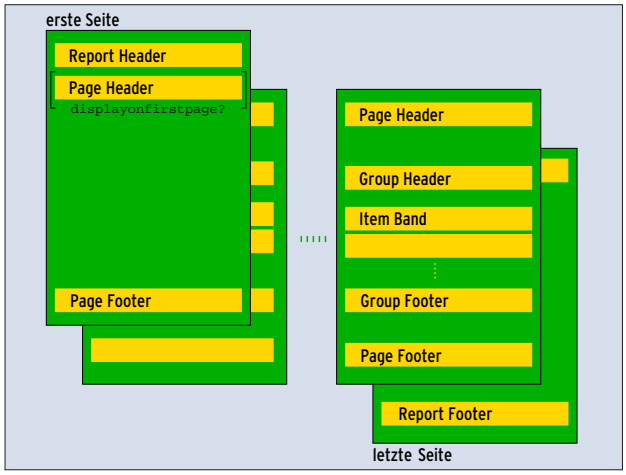


Abbildung 1: JFreereport baut die Seiten aus einzelnen Streifen (Bands) auf. Die eigentlichen Daten erscheinen im so genannten Item Band.

bliothek sinnvoll zu nutzen, ist ein Einblick in die Layoutkonzepte von JFreereport notwendig. Das API erlaubt es, das Layout direkt in Java-Code zu programmieren. Diese Lösung ist aber unflexibel, da sie die Gestaltung zur Compilezeit festlegt. Deshalb geht dieser Coffee-Shop nicht weiter darauf ein, sondern behandelt nur die XML-basierte Definition, mit der sich das Layout zur Laufzeit verändern lässt.

Layoutkonzepte

Ein Layoutelement heißt in der JFreereport-Sprache Band (Englisch für Streifen). Die äußere Klammer um den Report bilden der Report Header und der Report Footer. Der Report Header erscheint am Anfang des Dokuments und enthält typischerweise die Hauptüberschrift. Der Footer steht am Ende des gesamten Reports und dient für Anmerkungen, Quellenangaben und Ähnliches (Abbildung 1).

Jede Seite hat einen Page Header und einen Page Footer. Die Style-Option »display-on-first-page« legt fest, ob der Page Header auf der ersten Seite erscheint. Inhalt der Page Header und Footer könnten laufende Überschriften, Seitenzahlen oder Copyright-Vermerke sein.

Gruppen bilden

Eine wichtige Funktionalität jeder Reporting-Software ist es, zusammengehörige Daten zu gruppieren. Auch JFreereport unterstützt Gruppen, das Layout bestimmen dabei Group Header und Group Footer. Dafür erwartet JFreereport entsprechend sortierte Daten. Group Header und Footer dienen oft nicht nur als optische Gliederung, sondern auch zur Anzeige berechneter Felder: Sind Umsatzzahlen nach Produkten gruppiert, kann der Group Footer den Gesamtumsatz per Produkt ausgeben.

Träger für die eigentlichen Daten ist das Item Band. Für jede Datenzeile im zugrunde liegenden »TableModel« erzeugt JFreereport ein Item Band. Funktionsmäßig etwas aus der Reihe tanzt das Watermark Band, das jede Seite mit einem Wasserzeichen versieht. Damit der eigentliche Inhalt nicht gestört wird, sollte es sich um ein sehr kontrastarmes, helles Bild handeln.

Die Architektur

Seit der Version 0.8.4 ist JFreereport modular aufgebaut. Die Datei »modules.txt« beschreibt die einzelnen Module genauer. Wichtiger als die modularisierte Implementation ist aber der logische Aufbau der Report-Generierung, in deren Mittelpunkt drei Komponenten stehen. Die erste ist die Datenquelle des Reports: eine Klasse, die das »TableModel«-Interface implementiert (siehe Kasten „Das »TableModel«-Interface“). Es folgt die Layoutdefinition des Reports mit den einzelnen Bands, eben die erwähnte XML-Datei. Sie wird in dem Beispiel weiter unten erläutert. Die dritte Komponente ist die Report Engine: ein Objekt der Klasse »JFreeReport«.

Ein Anwendungsprogramm, das die JFreereport-Bibliothek nutzt, muss diese drei Komponenten erzeugen und verknüpfen. Aus der Layoutdefinition erstellt ein Parser des Pakets die Report Engine. Anschließend verknüpft die Methode »setData()« diese mit der Datenquelle. Den Report selbst erzeugen dann so genannte Output Targets. JFreereport unterstützt drei Typen: Raw Targets im CSV-Format (Comma Separated Values)

Das »TableModel«-Interface

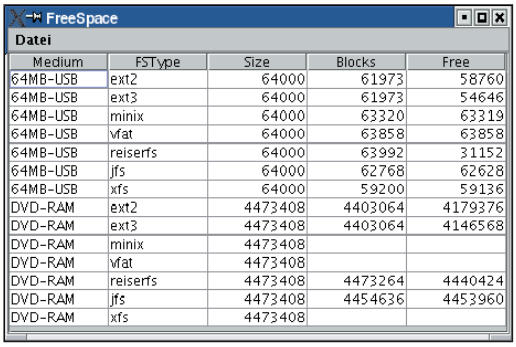
Die Klasse »javax.swing.JTable« ist ein Widget, das zweidimensionale Datentabellen anzeigt. Wie bei Swing üblich folgt diese Komponente dem MVC-Paradigma, das heißt, dass diese View-Komponente mit einer Model-Komponente für die Datenhaltung verknüpft ist. Die Model-Klasse muss dann das »javax.swing.table.TableModel«-Interface implementieren.

Die wichtigste Methode dieser Schnittstelle ist zweifellos »getValueAt(int row, int col)«. Weitere Methoden unterstützen die Controller-Komponente (»add...«- beziehungsweise »removeTableModelListener()«) oder fragen Metadaten des Model ab, beispielsweise mit »getColumnCount()« oder »getRowCount()«. Wie so oft bei AWT oder Swing existiert eine abstrakte Klasse, hier »AbstractTableModel«. Wenige Zeilen genügen, um die Schnittstelle zu implementieren:

```
TableModel simpleModel = new ➥
AbstractTableModel() {
  public int getColumnCount() {
    return 5;
  }
  public int getRowCount() {
    return 5;
  }
  public Object getValueAt(int row, ➥
      int col) {
    return new Integer(row*col);
  }
};
```

Eine so einfache Implementation reicht jedoch selten aus. Normalerweise ergeben sich stattdessen zwei Möglichkeiten: entweder eine vorhandene Datenstruktur mit Hilfe des obigen Interface zu kapseln oder die Daten in ein »DefaultTableModel« zu laden. Letztere Klasse erweitert ebenfalls das »AbstractTableModel« und speichert die Daten in einem Vektor von Vektoren.

Eine wichtige Quelle zweidimensionaler Tabellen sind Datenbanken, genauer die Ergebnisse von Datenbankabfragen. Das JFreereport-API enthält dafür eine »ResultSetTableModelFactory« mit einer Methode »createTableModel (ResultSet rs)«. Diese wandelt auf einfache Weise ein Query-Ergebnis »ResultSet« in ein »TableModel«.



Medium	FSType	Size	Blocks	Free
64MB-USB	ext2	64000	61973	58760
64MB-USB	ext3	64000	61973	54646
64MB-USB	minix	64000	63320	63319
64MB-USB	vfat	64000	63858	63858
64MB-USB	reiserfs	64000	63992	31152
64MB-USB	jfs	64000	62768	62628
64MB-USB	xfs	64000	59200	59136
DVD-RAM	ext2	4473408	4403064	4179376
DVD-RAM	ext3	4473408	4403064	4146568
DVD-RAM	minix	4473408		
DVD-RAM	vfat	4473408		
DVD-RAM	reiserfs	4473408	4473264	4440424
DVD-RAM	jfs	4473408	4454636	4453960
DVD-RAM	xfs	4473408		

Abbildung 2: Das Rahmenprogramm zeigt die Rohdaten in einem Swing-»JTable« als einfache Tabelle.

oder in XML, Pageable Output Targets wie PDF oder Ascii-Text sowie Table Based Output Targets mit Formaten wie HTML, RTF oder Excel.

Zwischen der internen Report-Generierung und der Ausgabe lässt sich eine Preview-Komponente einbinden. Über ein Menü speichert oder exportiert der Nutzer von dort aus seinen Report. Die Methoden des Pakets schreiben das gewünschte Zielformat. Entsprechende Code-Schnipsel finden sich im Beispiel »source/org/jfree/report/demo/StraightToEverything.java«.

Die Qualität der Ausgabeformate ist sehr unterschiedlich. Der HTML-Export ist recht ansehnlich, verschluckt aber leider Felder im Group Header. Das Excel-Format versucht zwar, das Layout zu erhalten, erzeugt aber schlechte Excel-Dateien: zum Beispiel enthält eine Tabelle dann Leerzeilen – damit funktionieren die normalen Filterfunktionen nicht mehr.

Als Beispiel dient ein Report für dieselben Daten, die im letzten Coffee-Shop mit dem verwandten JFreechart-Paket als Diagramm dargestellt wurden [3]. Es geht also wieder um den freien Plattenplatz bei unterschiedlichen Linux-Dateisystemen frisch nach der Formatierung eines 64-MByte-USB-Sticks und einer DVD-RAM.

Die Praxis

Das Rahmenprogramm ist betont einfach gehalten, wie ein Blick auf Listing 1 und die zentralen Methoden zeigt. Es erzeugt ein einfaches Fenster, in dem es mit einem »JTable«-Objekt die Rohdaten darstellt (siehe Abbildung 2). Die Methode »createGUI()« in den Zeilen 77 bis 83 erzeugt das »JTable«-Objekt und setzt es in ein »JScrollPane«. Zuletzt baut die Methode eine Menüzeile auf, indem »setJMenuBar()« die Funktion »createMenuBar()« aufruft.

Über das Menü fordert der Anwender die Report-Vorschau an. Der dadurch ausgelöste Event wird von der Methode »processPreview()« (Listing 1, Zeilen 123 bis 139) verarbeitet: Sie holt durch den Aufruf von »parseReport()« ein »JFreeReport«-Objekt (Zeile 126), setzt die Datenquelle (Zeile 127) und erzeugt einen »PreviewFrame« (ab Zeile 128). Abbildung 3 zeigt diese leistungsstarke, grafische Komponente.

Die letzte abgedruckte Methode »parseReport()« (Zeile 149) kapselt jenen Teil, der die XML-Datei einliest sowie über den Parser das »JFreeReport«-Objekt erzeugt und dann zurückgibt. Normalerweise würde man nicht bei jedem Event das Objekt neu erzeugen. Während der Arbeit am Layout ist dies aber sehr praktisch, da auf diese Weise Änderungen ohne einen Neustart des Rahmenprogramms wirksam werden.

Das Layout

Mit dem JFreereport-API die Reportfunktionen programmieren ist nicht sehr schwierig, wie Listing 1 zeigt. Komple-

Listing 1: »FreeSpace.java«

```
022 import java.awt.*;
023 import java.awt.event.*;
024 import java.net.*;
025 import java.text.*;
026 import javax.swing.*;
027 import javax.swing.table.*;
028
029 import org.jfree.report.*;
030 import org.jfree.report.modules.gui.base.*;
031 import org.jfree.report.modules.gui.base.components.*;
032 import org.jfree.report.modules.parser.base.*;
033 import org.jfree.report.util.*;
034 import org.jfree.ui.*;
035
043 public class FreeSpace extends JFrame {
044
045   private static final String REPORT_DEFINITION="freespace-report.xml";
046
077   public void createGUI() {
078     iData = new FreeSpaceTableModel();
079     final JTable table = new JTable(iData);
080     final JScrollPane scrollPane = new JScrollPane(table);
081     getContentPane().add(scrollPane,BorderLayout.CENTER);
082     setJMenuBar(createMenuBar());
083   }
084
123   protected void processPreview() {
124     final JFreeReport report;
125     try {
126       report = parseReport();
127       report.setData(iData);
128       PreviewFrame frame = new PreviewFrame(report);
129       frame.getBase().setToolbarFloatable(true);
130       frame.pack();
131       RefineryUtilities.positionFrameRandomly(frame);
132       frame.setVisible(true);
133       frame.requestFocus();
134     } catch (Exception ex) {
135       JOptionPane.showMessageDialog(this,ex.toString(),
136         "Exception",JOptionPane.ERROR_MESSAGE);
137       return;
138     }
139   }
140
149   private JFreeReport parseReport() throws Exception {
150     JFreeReport result = null;
151     final ReportGenerator generator = ReportGenerator.getInstance();
152     result = generator.parseReport(REPORT_DEFINITION);
153     return result;
154   }
170 }
```

xer ist die Layoutgestaltung, denn bislang gibt es keinen grafischer Editor dafür. Eine Reihe von Projekten arbeitet daran, entsprechende Links nennt die JFreereport-Homepage. Bis auf weiteres müssen die einzelnen Bands in XML von Hand kodiert werden.

Der Beispielreport in Abbildung 3 demonstriert eine Reihe von Möglichkeiten. Er gibt die Daten gruppiert nach Medium (USB-Stick oder DVD) aus. Für jeden Datensatz berechnet der Report automatisch, wie viel Prozent des theoretisch vorhandenen Platzes verfügbar ist (letzte Spalte in Blau).

Das Beispiel verwendet auch die Gruppenfunktion von JFreereport. Für jede Gruppe (also für jedes Speichermedium) soll es die Anzahl der tatsächlich verwendeten Dateisysteme berechnen. So sind die Dateisysteme Minix, FAT und XFS für DVD-RAM nicht verfügbar (genau genommen gilt dies nicht für Minix, aber wer will schon ein maximal 64 MByte großes Dateisystem auf einem 4,3 GByte großen Medium anlegen).

Listing 2 zeigt in Ausschnitten die Definition des Report Headers. Die restlichen Bands sind aus Platzgründen nicht abgedruckt; das vollständige Listing und alle Programme stehen auf dem Server des Linux-Magazins [4] bereit.

Das »style«-Tag in den Zeilen 128 bis 133 definiert den allgemeinen Stil des Band, den Standardstil aller Elemente legt das »default-style«-Tag fest (Zeilen 135 bis 139). Für sehr einfache Werte genügen »basic-key«-

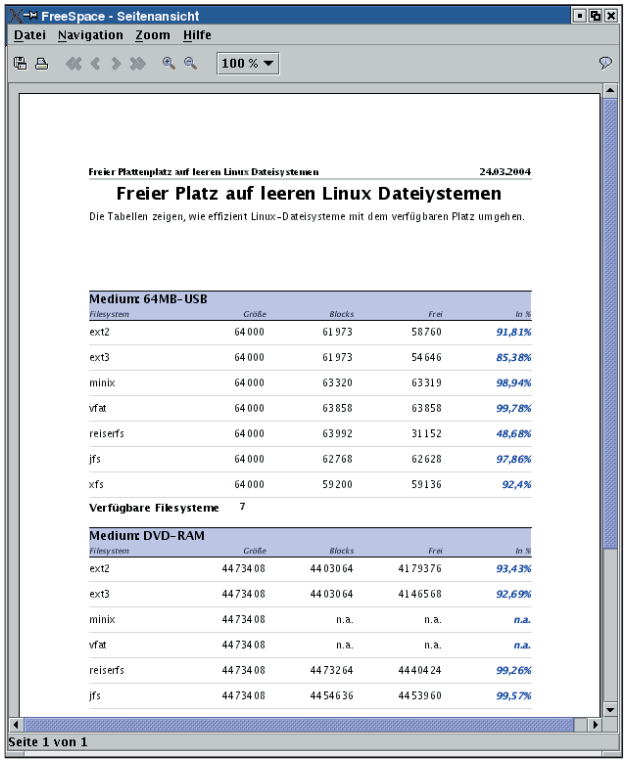


Abbildung 3: Die Preview-Komponente ordnet die Daten übersichtlicher als der einfache Rahmen, sie besitzt zudem ein Menü für Druck und Export.

Listing 2: Definition des Report Headers

```
001 <?xml version="1.0" encoding="ISO-8859-1"?>
002 <!DOCTYPE report-definition
003    PUBLIC "-//JFreeReport//DTD report definition//EN//extended"
004           "http://jfreereport.sourceforge.net/extreport.dtd">
005
099   <report-description>
100
127     <report-header name="report-header-band">
128       <style>
129         <compound-key name="min-size">
130           <basic-object name="height">112.0</basic-object>
131           <basic-object name="width">0.0</basic-object>
132         </compound-key>
133       </style>
134
135       <default-style>
136         <basic-key name="font">SansSerif</basic-key>
137         <basic-key name="font-size">10</basic-key>
138         <basic-key name="valignment">bottom</basic-key>
139       </default-style>
140
141       <element name="title1" type="text/plain">
142         <style>
143           <compound-key name="absolute_pos">
144             <basic-object name="x">0.0</basic-object>
145             <basic-object name="y">4.0</basic-object>
146           </compound-key>
147           <compound-key name="min-size">
148             <basic-object name="height">18.0</basic-object>
149             <basic-object name="width">-100.0</basic-object>
150           </compound-key>
151           <basic-key name="alignment">center</basic-key>
152           <basic-key name="font-size">18</basic-key>
153           <basic-key name="font-bold">true</basic-key>
154         </style>
155         <template references="label">
156           <basic-object name="content">Freier Platz auf
                leeren Linux-Dateiystemen</basic-object>
157         </template>
158       </element>
159
160       <element name="description1" type="text/plain">
161         <style>
162           <compound-key name="absolute_pos">
163             <basic-object name="x">0.0</basic-object>
164             <basic-object name="y">32.0</basic-object>
165           </compound-key>
166           <compound-key name="min-size">
167             <basic-object name="height">10.0</basic-object>
168             <basic-object name="width">-100.0</basic-object>
169           </compound-key>
170         </style>
171
172         <template references="label">
173           <basic-object name="content">Platzverbrauch von
                Linux-Dateisystemen.</basic-object>
174         </template>
175       </element>
176
177     </report-header>
710   </report-description>
711 </report-definition>
```

Tags (Zeile 136), zusammengesetzte erfordern dagegen die Kombination von »compound-key« und »basic-object« (ab Zeile 129).

Den Inhalt selbst beschreiben »element«-Tags. Das Attribut »type« legt den Typ fest, die Daten sind als »basic-object« mit dem Attribut »name = "content"« ausgezeichnet (Zeile 156). Hier lässt sich für das Element auch der Default-Style überschreiben (Zeilen 142 bis 154 und 161 bis 170). So verändert die Zeile 152 die Schriftgröße der ersten Titelzeile.

Felder und Funktionen

Referenzen auf Templates steuern das Aussehen der Elemente, in Listing 2 sind es Labels. Das Item Band referenziert dagegen typischerweise Spalten aus »TableModel« oder Funktionen, die ein vorgegebenes Interface implementieren. Im Layout definiert man Namen für die Funktionen (Listing 3, Zeilen 72 bis 94). Diese Namen referenziert das entsprechende Feld im Item Band (Zeile 705). Die Zeilen 702 bis 706 beschreiben das »number-field«-Template näher. So legt Zeile 703 fest, dass JFreereport »n.a.« ausgibt, wenn ein Wert nicht vorhanden, also »NULL« ist.

Dank des frei verfügbaren Quellcodes lassen sich recht einfach eigene Funktionen zu JFreereport hinzufügen. Das Beispiel nimmt nur triviale Erweiterungen bestehender Funktionen vor. Zuletzt sollen noch zwei Alternativen erwähnt werden. Jasper Reports [5] verfolgt einen ähnlichen Ansatz wie JFreereport. Für die Ausgabe verwendet es dieselben Bibliotheken, beispielsweise I-Text, um PDFs zu schreiben. Auch Jasper Reports definiert das Layout mit XML, ist aber mehr datenorientiert.

Die zweite Alternative gibt es nicht als fertiges Paket, sollte aber ohne zu großen Aufwand implementierbar sein. Die Idee ist es, den Layoutprofi Tex zu verwenden. Eine Template-Engine wie Velocity aus dem Jakarta-Projekt könnte Tex-Fragmente mit Daten füllen und anschließend das Layout dem Tex-Programm überlassen. Das wäre zwar keine reine Java-Lösung, aber da Tex auf vielen Plattformen zur Verfügung steht, ist dies keine prinzipielle Einschränkung.

finally{}

JFreereport ist mächtig und erzeugt schöne Reports – eine saubere Layoutdefinition vorausgesetzt. Das Java-API überzeugt durch seine Struktur und die klaren Abstraktionsebenen. Wenige Zeilen Code ergeben ein leistungsfähiges Programm. Es wäre schön, wenn es beim Layout ebenso wäre: ein automatisch erzeugtes Default-Layout aus einer minimalen Beschreibung. *(ofr)* ■

Infos

[1] JFreereport-Homepage: [http://jfree.org/jfreereport]
[2] Dokumentation bei Object-Refinery: [http://www.object-refinery.com/jfreereport/]
[3] Bernhard Bablok, „Malen nach Zahlen, Diagramme mit JFreechart": Linux-Magazin 5/04, S. 116
[4] Listings zum Artikel: [http://www.linux-magazin.de/Service/Listings/2004/06/Coffeeshop]
[5] Jasper Reports: [http://jasperreports.sf.net]

Der Autor

Bernhard Bablok arbeitet bei der AGIS mbH als Anwendungsentwickler. Wenn er nicht Musik hört, mit dem Radl oder zu Fuß unterwegs ist, beschäftigt er sich mit Themen rund um Objektorientierung. Er ist unter der Adresse [coffee-shop@bablokb.de] zu erreichen.

Listing 3: Ein Feld des Item Band

```
072  <functions>
075
076    <expression name="PerCent"
077                class="PerCentExpression">
078      <properties>
079        <property name="dividend">Free</property>
080        <property name="divisor">Size</property>
081      </properties>
082    </expression>
083
087    <function name="AvailableFS" class="ItemNotNullCountFunction">
088      <properties>
089        <property name="field">Free</property>
090        <property name="group">Medium</property>
091      </properties>
092    </function>
093
094  </functions>
098
099  <report-description>
580
581    <!-- ============= -->
582    <!-- = ITEM BAND = -->
583    <!-- ============= -->
584    <itemband>
585
687      <element type="text/plain">
688        <style>
689          <compound-key name="absolute_pos">
690            <basic-object name="x">-80.0</basic-object>
691            <basic-object name="y">8.0</basic-object>
692          </compound-key>
693          <compound-key name="min-size">
694            <basic-object name="height">10.0</basic-object>
695            <basic-object name="width">-20.0</basic-object>
696          </compound-key>
697          <basic-key name="alignment">right</basic-key>
698          <basic-key name="paint">blue</basic-key>
699          <basic-key name="font-italic">true</basic-key>
700          <basic-key name="font-bold">true</basic-key>
701        </style>
702        <template references="number-field">
703          <basic-object name="nullValue">n.a.</basic-object>
704          <basic-object name="format">#####.##%</basic-object>
705          <basic-object name="field">PerCent</basic-object>
706        </template>
707      </element>
708    </itemband>
709
710  </report-description>
711 </report-definition>
```