

# Inhaltsverzeichnis

<b>1</b>	<b>Vorwort</b>	<b>V</b>
<b>2</b>	<b>Automatisierte Integration mit Anthill &amp; Co.</b>	<b>1</b>
2.1	Überblick .....	1
2.2	Probleme bei der Softwareentwicklung im (virtuellen) Team .....	1
2.3	Lösungsideen und Lösungskonzepte .....	2
2.4	Werkzeuge .....	3
2.5	Einheitliche Modulstruktur .....	4
2.5.1	Modultypen .....	4
2.5.2	Beispiele Modulstruktur .....	5
2.6	Das Integrationstool Anthill .....	6
2.6.1	Das Konzept von Anthill .....	6
2.6.2	Arbeitsablauf mit Anthill .....	6
2.6.3	Unterstützung durch Urbancode .....	8
2.6.4	Die Zukunft von Anthill .....	8
2.7	Zusammenfassung .....	8
2.8	Links .....	9
2.9	Über den Autor .....	9
<b>3</b>	<b>Source-Code-Generatoren</b>	<b>11</b>
3.1	Definition .....	11
3.2	Motivation .....	11
3.3	Forderungen .....	12
3.3.1	Allgemein .....	12
3.3.2	Generatoren-Plattform .....	12
3.3.3	Generat .....	13
3.4	Prozess .....	13
3.5	Ansätze .....	15
3.5.1	Modell .....	15
3.5.2	Technisch .....	15
3.5.3	Architektur .....	16

3.6	Beispiel .....	18
3.6.1	Ziele .....	18
3.6.2	XML .....	19
3.6.3	Modelle .....	20
3.6.4	Generat .....	21
3.7	Java-Generator .....	23
3.7.1	Generierungsablauf .....	24
3.7.2	Methodenimplementierung .....	26
3.7.3	XSLT-Generator .....	27
3.7.4	XSLT-Facadenimplementierung .....	30
3.7.5	Jython-Generator .....	32
3.7.6	Analyse .....	38
3.7.7	Bewertung .....	38
3.8	Fazit .....	40
3.9	Andere Ansätze .....	40
3.10	Links .....	41
3.11	Über den Autor .....	41
<b>4</b>	<b>Das Dynamic Attribute Set Pattern</b>	<b>43</b>
4.1	Definition und Überblick über Patterns .....	43
4.2	Nutzen von Patterns .....	44
4.3	Arbeitsweise mit Patterns .....	44
4.4	Quellen für Patterns .....	45
4.5	Das Dynamic Attribute Set Pattern (DAS) .....	46
4.5.1	Problematik .....	46
4.5.2	Anwendungsfälle .....	46
4.5.3	Lösung mit dem Dynamic Attribute Set Pattern (DAS) .....	48
4.6	Fazit .....	54
4.7	Über die Autoren .....	54
<b>5</b>	<b>Pragmatic Programmer</b>	<b>55</b>
5.1	Projektrealität .....	55
5.2	Auswege .....	56
5.3	Das Muster .....	58
5.4	Die Bibliothek .....	69
5.4.1	Interfaces .....	69
5.4.2	Erweiterungen .....	70
5.4.3	Defaultimplementierungen .....	71
5.4.4	Hilfsklassen .....	71

5.4.5	Introspection .....	71
5.4.6	Abgrenzung .....	73
5.5	Der Prozess .....	74
5.6	Schlussbemerkung .....	78
5.7	Weitergehende Literatur .....	78
5.8	Über den Autor .....	79
<b>6</b>	<b>KGB-Programmierung</b>	<b>81</b>
6.1	Motivation .....	81
6.1.1	Null-Check .....	81
6.1.2	Exception Container .....	84
6.1.3	Klassenspezifische Validatoren .....	85
6.1.4	Starke Typisierung nutzen .....	87
6.1.5	Zusammenspiel mit Logging und Unittests .....	89
6.2	Zusammenfassung .....	89
6.3	Über die Autoren .....	89
<b>7</b>	<b>Java und relationale Datenbanken</b>	<b>91</b>
7.1	Ein SQL-Wrapper macht Applikationen flexibel .....	91
7.2	Grundlagen .....	91
7.2.1	Relationale Modell .....	91
7.2.2	SQL .....	91
7.2.3	JDBC .....	92
7.2.4	Das Problem .....	92
7.2.5	Nachfolgend verwendete Beispiel .....	92
7.3	Die Fallen .....	93
7.3.1	Falle Join .....	93
7.3.2	Falle Position .....	94
7.3.3	Falle Sortierung .....	94
7.3.4	Falle Alias-Namen .....	94
7.3.5	Falle Group .....	95
7.3.6	Falle Funktion .....	95
7.3.7	Falle Datentypen .....	95
7.3.8	Falle Sommerzeit .....	96
7.3.9	Falle Borland-Save .....	96
7.3.10	Falle Memory Leak 1 .....	96
7.3.11	Falle Memory Leak 2 .....	96
7.3.12	Falle Memory Leak 3 .....	97
7.3.13	Falle Datenbank-Performance .....	97
7.3.14	Falle SQL-Bugs .....	97
7.3.15	Falle SQL-Umfang und SQL-Version .....	97
7.3.16	Falle Syntax .....	97

7.3.17	Falle strukturelle Limits .....	98
7.3.18	Falle Installation .....	98
7.3.19	Falle Bestellung .....	98
7.4	Das Kochbuch für den SQL-Wrapper .....	98
7.4.1	Rezept: Datenbank Verbindung .....	98
7.4.2	Rezept: Jede Tabelle in eine Klasse packen .....	99
7.4.3	Rezept: Datenbank-Manipulation kapseln .....	100
7.4.4	Rezept: Datenbank-Select kapseln .....	100
7.4.5	Rezept: Kapseln von ResultSet-Datentypen .....	101
7.4.6	Rezept: Java-Datentypen kapseln .....	102
7.4.7	Rezept: Datenbank-Funktionen kapseln .....	102
7.4.8	Rezept: Kommandoabschlusszeichen „;“ kapseln .....	102
7.4.9	Rezept: Datenmigration .....	102
7.4.10	Zusammenfassung .....	102
7.5	Alternativen zum SQL-Wrapper .....	103
7.5.1	Visuelle Programmierung .....	103
7.5.2	Model Driven Architektur (MDA) .....	103
7.5.3	Java-Framework .....	103
7.5.4	SQLJ .....	103
7.5.5	Java Data Objects (JDO) .....	104
7.6	Fazit .....	104
7.7	Links .....	105
7.8	Über den Autor .....	105
<b>8</b>	<b>SWT - Plattformübergreifend natives Look &amp; Feel</b>	<b>107</b>
8.1	Introducing SWT .....	107
8.2	Installation .....	108
8.2.1	Bibliotheken .....	108
8.2.2	Dokumentation .....	108
8.3	Struktur einer SWT-Anwendung .....	109
8.4	Typisch SWT .....	109
8.4.1	Erzeugen von Objekten .....	110
8.4.2	readAndDispatch-Schleife .....	110
8.4.3	Style Bits .....	110
8.4.4	Ressourcenverwaltung .....	111
8.5	Widgets .....	112
8.6	Events .....	114
8.7	Grafik .....	115
8.8	Layouts .....	115
8.9	Distribution .....	116

8.10	Zusammenfassung .....	118
8.11	Links .....	118
8.12	Über den Autor .....	119
<b>9</b>	<b>Internationalisierung</b>	<b>121</b>
9.1	Mehrsprachige Anwendungen .....	121
9.2	Ermittlung der Länder- bzw. Spracheinstellungen .....	121
9.2.1	Full Client/Applikation .....	122
9.2.2	Web Client .....	122
9.2.3	Benutzerdefiniert .....	122
9.2.4	Server .....	122
9.3	Ressourcen verwenden .....	123
9.3.1	ResourceBundles .....	123
9.3.2	Verwendung der Ressourcen .....	124
9.3.3	Formatierung von Zahlen und Nachrichten .....	125
9.4	Datum und Zeit .....	127
9.4.1	Kalender .....	127
9.4.2	Zeitzonen .....	127
9.4.3	Ein-/Ausgabe .....	128
9.5	I18N in Apache Jakarta und JSTL .....	128
9.5.1	Apache Jakarta i18n Tag Library .....	128
9.5.2	JSP Standard Tag Library .....	129
9.5.3	Apache Struts .....	130
9.6	Fortgeschrittene Themen .....	130
9.6.1	Kodierung und Zeichensätze .....	130
9.6.2	Schriftarten .....	131
9.6.3	Grafische Benutzungsoberflächen .....	131
9.6.4	Sprachsensitive Sortierung .....	132
9.7	Links .....	132
9.8	Über den Autor .....	133
<b>10</b>	<b>Java Native Interface</b>	<b>135</b>
10.1	Motivation .....	136
10.1.1	Erweiterung einer vorhandenen Anwendung mit Java .....	137
10.1.2	Wiederverwendung vorhandener C/C++-Programmteile .....	137
10.2	Grundlagen .....	138
10.2.1	Das „Hello World“-Beispiel .....	138
10.2.2	Erweitern einer bestehenden C/C++-Anwendung mit Java .....	143
10.3	Toolunterstützung .....	148
10.3.1	cxxwrap .....	149

10.3.2	SWIG .....	154
10.3.3	Weitere Tools .....	161
10.4	Links .....	162
10.5	Über die Autoren .....	162
<b>11</b>	<b>Mobile Datenkommunikation</b>	<b>163</b>
11.1	Was ist „Mobile Datenkommunikation“? .....	163
11.2	Wie macht man Mobile Datenkommunikation? .....	163
11.2.1	Socket Kommunikation TCP .....	163
11.2.2	Socket Kommunikation UDP .....	164
11.2.3	RMI .....	165
11.2.4	CORBA .....	165
11.2.5	JMS .....	166
11.2.6	SOAP .....	166
11.3	Herausforderungen der Mobilen Datenkommunikation .....	168
11.3.1	TCP .....	168
11.3.2	RMI, CORBA, JMS, SOAP .....	170
11.3.3	UDP .....	171
11.4	Optimierungsmöglichkeiten .....	172
11.4.1	Verwendung von Wireless TCP .....	172
11.4.2	Lösung bei Verwendung von Standard-TCP .....	172
11.4.3	Lösung mit UDP .....	172
11.4.4	Optimale Lösung .....	173
11.5	Datenkommunikation in der Praxis .....	173
11.5.1	RMI über GPRS .....	174
11.5.2	JMS über GPRS .....	175
11.5.3	HTTP Kommunikation im J2ME Midlet .....	177
11.6	Nebenbei erwähnt .....	179
11.7	Über den Autor .....	179
<b>12</b>	<b>Server-Konfiguration und -Betrieb</b>	<b>181</b>
12.1	Überblick .....	181
12.2	Entwicklung .....	181
12.2.1	Besondere Anforderungen .....	183
12.2.2	Lesen von Konfigurationsparametern .....	184
12.2.3	Kontrollierte Initialisierung von Web-Anwendungen .....	185
12.2.4	Dynamische Änderungen über das Preferences-API .....	186
12.2.5	Logging mittels Log4j .....	188
12.3	Betrieb .....	191
12.3.1	Schnelles Logging und Log-Level .....	192
12.3.2	Mail .....	193

12.3.3	Dynamisches Ausführen von Java-Code .....	194
12.4	Zusammenfassung .....	195
12.5	Links .....	195
12.6	Über die Autoren .....	195

