

Erik Hatcher,  
Steve Loughran

# Java-Entwicklung mit Ant

Übersetzung aus dem Amerikanischen  
von Reinhard Engel

Technische Universität Darmstadt FACHBEREICH INFORMATIK	
B I B L I O T H E K	
Inventar-Nr.:	<u>MO7-00201</u>
Sachgebiete:	<u>Software</u>
Standort:	<u>D. 4 / Hatc</u>



# Inhaltsverzeichnis

<b>Vorwort</b>	<b>19</b>
<b>Einleitung</b>	<b>21</b>
<b>Über dieses Buch</b>	<b>25</b>
Wer dieses Buch lesen sollte .....	25
Wie dieses Buch aufgebaut ist .....	26
Online-Ressourcen .....	28
Konventionen in diesem Buch .....	28
Ant-Versionen und andere Projekte .....	29
Author-Online .....	29
<b>Über die Autoren</b>	<b>30</b>
<b>Teil 1: Ant lernen</b>	<b>31</b>
<b>Kapitel 1: Eine Einführung in Ant</b>	<b>33</b>
1.1 Was ist Ant ? .....	33
1.1.1 Was ist ein Build-Prozess und warum brauchen Sie einen? .....	34
1.1.2 Warum halten wir Ant für ein großartiges Build-Werkzeug? .....	34
1.2 Die Kernkonzepte von Ant .....	35
1.2.1 Ein Beispielprojekt .....	36
1.3 Warum sollten Sie Ant verwenden? .....	40
1.3.1 Integrierte Development Environments .....	40
1.3.2 Make .....	40
1.3.3 Andere Build-Werkzeuge .....	43
1.3.4 Schnell produktiv arbeiten .....	43
1.4 Die Entwicklung von Ant .....	43
1.5 Ant und Methoden zur Software-Entwicklung .....	45
1.5.1 Extreme Programming .....	45
1.6 Unser Beispielprojekt .....	47
1.6.1 Dokumentations-Suchmaschine – Ant-Beispielprojekt .....	47

1.7	Schön und gut, aber kann Ant auch ...	48
1.8	Jenseits der Java-Entwicklung	49
1.8.1	Webpublishing-Engine	50
1.8.2	Einfache Workflow-Engine	50
1.8.3	Microsoft .Net und andere Sprachen	50
1.9	Zusammenfassung	50

**Kapitel 2: Der Einstieg in Ant 53**

2.1	Unser erstes Projekt definieren	53
2.2	Schritt 1: Prüfen, ob die Tools einsatzbereit sind	54
2.3	Schritt 2: Das erste Ant-Buildfile schreiben	54
2.3.1	Das Buildfile analysieren	55
2.4	Schritt 3: Den ersten Build ausführen	56
2.4.1	Wenn der Build scheitert	57
2.4.2	Den Build genauer analysieren	59
2.5	Schritt 4: Struktur schaffen	62
2.5.1	Die Source-Verzeichnisse strukturieren	63
2.5.2	Die Build-Verzeichnisse strukturieren	64
2.5.3	Die Dist-Verzeichnisse strukturieren	64
2.5.4	Das Buildfile erstellen	66
2.5.5	Target-Abhängigkeiten	66
2.5.6	Das neue Buildfile ausführen	68
2.5.7	Den Build erneut ausführen	68
2.5.8	Wie Ant mehrere Targets auf der Kommandozeile verarbeitet	69
2.6	Schritt 5: Das Programm ausführen	70
2.6.1	Warum ein Programm aus Ant heraus ausführen	71
2.6.2	Ein execute-Target hinzufügen	71
2.6.3	Das neue Target ausführen	72
2.7	Kommandozeilenoptionen von Ant	73
2.7.1	Das Buildfile angeben, das ausgeführt werden soll	74
2.7.2	Den Umfang der angezeigten Informationen steuern	74
2.7.3	Information über ein Projekt abfragen	76
2.8	Das endgültige Buildfile	77
2.9	Zusammenfassung	78

**Kapitel 3: Ant-Datentypen und Properties verstehen 81**

3.1	Grundkonzepte	81
3.1.1	Ein Überblick über die Datentypen	81
3.1.2	Property-Überblick	82
3.2	Die Datentypen und Properties von <javac>	82
3.3	Pfade	85
3.4	Filesets	85
3.4.1	Fileset-Beispiele	86
3.4.2	Default Excludes	86
3.5	Patternsets	87

3.6	Selektoren	89
3.7	Die Benennung von Datentyp-Elementen	90
3.8	Filterset	91
3.8.1	Zeitstempel zur Build-Zeit in Dateien einfügen	91
3.9	FilterChains und FilterReader	92
3.10	Mapper	95
3.10.1	Identity-Mapper	95
3.10.2	Flatten-Mapper	96
3.10.3	Merge-Mapper	96
3.10.4	Glob-Mapper	97
3.10.5	Regexp-Mapper	97
3.10.6	Package-Mapper	98
3.11	Zusätzliche Ant-Datentypen	99
3.11.1	ZipFileset	99
3.11.2	Dirset	99
3.11.3	Filelist	99
3.11.4	ClassFileset	100
3.12	Properties	100
3.12.1	Properties mit dem <code>&lt;property&gt;</code> -Task setzen	103
3.12.2	Warum der <code>&lt;property&gt;</code> -Task etwas Spezielles ist	106
3.12.3	Die Verfügbarkeit von Ressourcen prüfen: <code>&lt;available&gt;</code>	106
3.12.4	Zeit durch Überspringen unnötiger Schritte sparen: <code>&lt;uptodate&gt;</code>	108
3.12.5	Bedingungen testen: <code>&lt;condition&gt;</code>	108
3.12.6	Properties von der Kommandozeile aus setzen	109
3.12.7	Einen Build-Zeitstempel erstellen: <code>&lt;tstamp&gt;</code>	110
3.12.8	Properties aus einer XML-Datei laden	112
3.13	Ant mit Properties steuern	113
3.13.1	Targets bedingt ausführen	113
3.13.2	Patternsets bedingt ein- oder ausschließen	114
3.13.3	Builds bedingt abbrechen	114
3.14	Referenzen	115
3.14.1	Properties und Referenzen	116
3.14.2	Eingebettete Patternsets referenzieren	117
3.15	Best Practices	118
3.16	Zusammenfassung	119

## Kapitel 4: Mit JUnit testen 121

4.1	Refactoring	121
4.2	Die Java-main()-Methode testen	122
4.3	Eine Einführung in JUnit	123
4.3.1	Einen Testfall schreiben	123
4.3.2	Einen Testfall ausführen	124
4.3.3	Gewünschte Ergebnisse zusichern	125
4.3.4	Der Lebenszyklus eines Testfalls	126
4.3.5	Eine Testsuite schreiben	126
4.3.6	JUnit beschaffen und installieren	127
4.3.7	Erweiterungen von JUnit	127

4.4	Unit-Tests auf unsere Anwendung anwenden	128
4.4.1	Den Test zuerst schreiben	128
4.4.2	Beim Testen externe Ressourcen nutzen	129
4.5	Der JUnit-Task: <junit>	130
4.5.1	Die Verzeichnisstruktur an das Testen anpassen	130
4.5.2	JUnit in den Build-Prozess einfügen	131
4.6	Scheitern eines Tests ist ein Scheitern des Builds	133
4.6.1	Testergebnisse sammeln	134
4.6.2	Mehrere Tests ausführen	136
4.6.3	Einen eigenen Ergebnis-Formatter erstellen	136
4.7	Testergebnisberichte erstellen	137
4.7.1	Berichte erzeugen und bei Testmisserfolgen ein Scheitern des Builds zulassen	138
4.7.2	Einen einzelnen Testfall von der Kommandozeile aus ausführen	140
4.7.3	Die Testumgebung initialisieren	140
4.7.4	Andere Testprobleme	141
4.8	Tests abkürzen	142
4.8.1	Eine große Anzahl von Tests bewältigen	145
4.9	Best Practices	146
4.10	Zusammenfassung	147

## **Kapitel 5: Programme ausführen** **149**

5.1	Warum Sie externe Programme ausführen müssen	149
5.2	Java-Programme ausführen	150
5.2.1	Eine Einführung in den <java>-Task	151
5.2.2	Den Klassenpfad setzen	152
5.2.3	Argumente	153
5.2.4	System-Properties definieren	155
5.2.5	Das Programm in einer neuen JVM ausführen	155
5.2.6	Umgebungsvariablen setzen	156
5.2.7	Die neue JVM steuern	156
5.2.8	Fehler mit »failonerror« behandeln	157
5.2.9	JAR-Dateien ausführen	158
5.2.10	Programme von Drittanbietern aufrufen	159
5.2.11	Vor dem Aufruf eines Java-Programms seine Existenz prüfen	161
5.2.12	Ein Timeout setzen	162
5.3	Native Programme mit <exec> starten	163
5.3.1	Umgebungsvariablen setzen	164
5.3.2	Fehler behandeln	164
5.3.3	Timeouts bearbeiten	165
5.3.4	Shell-Befehle erstellen und ausführen	166
5.3.5	Die Existenz eines Programms vor seinem Aufruf prüfen	167
5.4	Mehrere Dateien auf einmal bearbeiten: <apply>	168
5.5	Ausgaben verarbeiten	169
5.6	Einschränkungen bei der Ausführung	170
5.7	Best Practices	170
5.8	Zusammenfassung	171

<b>Kapitel 6:</b>	<b>Projekt-Packaging</b>	<b>173</b>
6.1	Dateien verschieben, kopieren und löschen	174
6.1.1	Dateien löschen	174
6.1.2	Dateien kopieren	176
6.1.3	Dateien verschieben	176
6.1.4	Filtern	177
6.2	Das Packaging vorbereiten	178
6.2.1	Den Release-Code erstellen und dokumentieren	178
6.2.2	Datendateien hinzufügen	180
6.2.3	Die Dokumentation vorbereiten	180
6.2.4	Installationskripts und Dokumente vorbereiten	182
6.2.5	Libraries für die Distribution vorbereiten	184
6.3	Archivdateien erstellen	185
6.3.1	JAR-Dateien	187
6.3.2	Eine JAR-Datei erstellen	187
6.3.3	Die JAR-Datei testen	188
6.3.4	Ein JAR-Manifest erstellen	189
6.3.5	Zusätzliche Metadaten zu der JAR-Datei hinzufügen	190
6.3.6	Best Practices für JAR-Dateien	191
6.3.7	JAR-Dateien signieren	191
6.4	Zip-Dateien erstellen	192
6.4.1	Eine Binär-Distribution erstellen	193
6.4.2	Eine Source-Distribution erstellen	195
6.4.3	Zip-Dateien vereinigen	196
6.4.4	Best Practices für Zip-Dateien	196
6.5	Tar-Dateien erstellen	196
6.6	Webanwendungen mit WAR-Dateien erstellen	198
6.7	Ein Package testen	200
6.8	Zusammenfassung	201
<b>Kapitel 7:</b>	<b>Deployment</b>	<b>203</b>
7.1	Deployment-Beispielprobleme	203
7.1.1	Die Tasks überprüfen	204
7.1.2	Tools für das Deployment	205
7.2	Tasks für das Deployment	205
7.2.1	Dateiübertragung mit <ftp>	205
7.2.2	Die Verfügbarkeit des Servers testen	206
7.2.3	Mit <sleep> Pausen in den Build einfügen	208
7.2.4	Der E-Mail-Task von Ant	208
7.2.5	Remote-Dateien mit <get> abrufen	210
7.2.6	Die Deployment-Tasks anwenden	210
7.3	FTP-basierte Distribution eines Anwendungs-Packages	211
7.3.1	Information mit dem <input>-Task abfragen	212
7.4	E-Mail-basierte Distribution eines Anwendungs-Packages	213
7.5	Ein lokales Deployment auf Tomcat 4.x	214

7.5.1	Das Tomcat Management Servlet API	214
7.5.2	Mit Ant nach Tomcat deployen	216
7.6	Remote-Deployment auf Tomcat	220
7.6.1	Zwischenspiel – Targets mit <antcall> aufrufen	222
7.6.2	Den <antcall>-Task beim Deployment verwenden	225
7.7	Das Deployment testen	226
7.8	Zusammenfassung	227

**Kapitel 8: Alle Schritte zusammensetzen 229**

8.1	Unsere Anwendung bis jetzt	229
8.2	Unsere eigene Ant-Task-Library erstellen	230
8.3	Gemeinsame Properties in mehreren Projekten laden	235
8.4	Abhängigkeiten bei verschiedenen Versionen bearbeiten	238
8.4.1	Eine neue Library-Version installieren	240
8.5	Die Buildfile-Philosophie	242
8.5.1	Denken Sie bereits am Anfang an das Ende	242
8.5.2	Integrieren Sie Tests in den Build	242
8.5.3	Unterstützen Sie ein automatisiertes Deployment	242
8.5.4	Machen Sie Ihren Code portabel	242
8.5.5	Planen Sie Anpassungsmöglichkeiten ein	243
8.6	Zusammenfassung	243

**Teil 2: Ant anwenden 245**

**Kapitel 9: Ant für Entwicklungsprojekte einsetzen 247**

9.1	Einen Ant-basierten Build-Prozess entwerfen	247
9.1.1	Ein Projekt analysieren	247
9.1.2	Das Kern-Buildfile erstellen	250
9.1.3	Das Buildfile ausbauen	250
9.2	Auf Ant umsteigen	250
9.3	Die zehn Schritte einer Migration	251
9.3.1	Der Umstieg von Make-basierten Projekten	252
9.3.2	Der Umstieg von IDE-basierten Projekten	253
9.4	Große Projekte mit Master-Builds verwalten	254
9.4.1	Refactoring der Buildfiles	254
9.4.2	Der <ant>-Task – eine Einführung	254
9.4.3	Beispiel: ein grundlegendes Master-Buildfile	255
9.4.4	Ein skalierbares, flexibles Master-Buildfile erstellen	257
9.5	Die Builds von Unterprojekten verwalten	263
9.5.1	Die Properties von Unterprojekten steuern	263
9.5.2	Properties und Referenzen von einem Master-Buildfile erben	265
9.5.3	Properties und Referenzen in <ant> deklarieren	266
9.5.4	Properties mit XML-Dateifragmenten gemeinsam nutzen	266
9.5.5	Targets mit XML-Dateifragmenten gemeinsam nutzen	268
9.6	Wiederverwendbare Library-Buildfiles erstellen	269

9.7	Ein Blick voraus – wie wird die Entwicklung großer Projekte in Zukunft unterstützt? .....	272
9.8	Best Practices für Ant-Projekte .....	273
9.8.1	Libraries verwalten .....	273
9.8.2	Prozesse implementieren .....	274
9.9	Zusammenfassung .....	274

## **Kapitel 10: Jenseits der Core-Tasks von Ant 275**

10.1	Die Typen von Tasks verstehen .....	275
10.1.1	Was also ist ein »optionaler« Task? .....	276
10.1.2	Die hauptsächlichlichen optionalen Tasks von Ant .....	276
10.1.3	Warum Drittanbieter-Tasks? .....	277
10.2	Optionale Tasks in Aktion .....	277
10.2.1	Property-Dateien manipulieren .....	278
10.2.2	Audiovisuelles Feedback zu einem Build hinzufügen .....	280
10.2.3	Abhängigkeitsprüfungen hinzufügen .....	281
10.2.4	Grammatiken mit JavaCC parsen .....	283
10.2.5	Reguläre Ausdrücke ersetzen .....	284
10.3	Tasks zum Software Configuration Management .....	285
10.3.1	CVS .....	286
10.3.2	ClearCase .....	287
10.4	Drittanbieter-Tasks verwenden .....	287
10.4.1	Tasks mit <taskdef> definieren .....	287
10.5	Erwähnenswerte Drittanbieter-Tasks .....	288
10.5.1	Checkstyle .....	288
10.5.2	Torque – objektrelationales Mapping .....	290
10.6	Die »ant-contrib«-Tasks .....	293
10.7	Task-Definitionen projektübergreifend nutzen .....	298
10.8	Best Practices .....	299
10.9	Zusammenfassung .....	299

## **Kapitel 11: XDoclet 301**

11.1	XDoclet installieren .....	301
11.2	To-do-Listen generieren .....	302
11.3	Die XDoclet-Architektur .....	304
11.3.1	Die Ant-Tasks von XDoclet .....	304
11.3.2	Templating .....	305
11.3.3	Wie XDoclet funktioniert .....	306
11.4	Ein eigenes XDoclet-Template schreiben .....	306
11.4.1	Code-Generierung .....	308
11.4.2	Klassenweise oder dateiweise Generierung .....	314
11.4.3	Verarbeitete Klassen filtern .....	314
11.5	Fortgeschrittene Aufgaben mit XDoclet lösen .....	315
11.5.1	Eigene Subtasks .....	315
11.5.2	Einen eigenen Tag-Handler erstellen .....	316

11.6 Die Richtung von XDoclet .....	316
11.6.1 XDoclet im Vergleich zu C# .....	317
11.6.2 Ein Blick in Javas Zukunft: jsr 175 und 181 .....	317
11.7 Best Practices für XDoclet .....	317
11.7.1 Abhängigkeitsprüfungen .....	317
11.8 Zusammenfassung .....	318

**Kapitel 12: Für das Web entwickeln** **319**

12.1 Welche speziellen Eigenschaften haben Webanwendungen? .....	319
12.2 Mit Tag-Libraries arbeiten .....	321
12.2.1 Eine Tag-Library erstellen .....	321
12.2.2 Tag-Libraries integrieren .....	326
12.2.3 Zusammenfassung der Taglib-Entwicklung mit Ant .....	328
12.3 JSP-Seiten kompilieren .....	328
12.3.1 Den <jspc>-Task installieren .....	329
12.3.2 Den <jspc>-Task anwenden .....	330
12.3.3 JSP-Seiten für das Deployment kompilieren .....	332
12.3.4 Andere Tasks für das Kompilieren von JSP-Seiten .....	332
12.4 Webanwendungen anpassen .....	332
12.4.1 Anpassung auf der Basis eines Filtersets .....	332
12.4.2 Deployment-Deskriptoren mit XDoclet anpassen .....	334
12.4.3 Libraries in der WAR-Datei anpassen .....	336
12.5 Statische Inhalte erzeugen .....	337
12.5.1 Neue Inhalte erzeugen .....	337
12.5.2 Neue Dateien erstellen .....	338
12.5.3 Vorhandene Dateien ändern .....	339
12.6 Webanwendungen mit HttpUnit testen .....	340
12.6.1 HttpUnit-Tests schreiben .....	340
12.6.2 Die Tests kompilieren .....	342
12.6.3 Die Ausführung von HttpUnit-Tests von Ant aus vorbereiten .....	343
12.6.4 Die HttpUnit-Tests ausführen .....	343
12.6.5 Die Tests integrieren .....	344
12.6.6 Die Grenzen von HttpUnit .....	346
12.6.7 Canoo WebTest .....	346
12.7 Serverseitiges Testen mit Cactus .....	350
12.7.1 Cactus aus der Sicht von Ant .....	351
12.7.2 Wie Cactus funktioniert .....	352
12.7.3 Und jetzt unser Testfall .....	353
12.7.4 Cactus: Zusammenfassung .....	354
12.8 Zusammenfassung .....	355

**Kapitel 13: Mit XML arbeiten** **357**

13.1 Präambel: Alles über XML-Libraries .....	358
13.2 XML validieren .....	358
13.2.1 Wenn eine Datei nicht validiert wird .....	360

13.2.2 XML-DTDs auflösen .....	361
13.2.3 Alternative XML-Validierungsmechanismen unterstützen .....	362
13.3 XML mit XSLT transformieren .....	362
13.3.1 Den XMLCatalog-Datentyp verwenden .....	365
13.3.2 PDF-Dateien aus XML-Source generieren .....	367
13.3.3 Styler – ein Drittanbieter-Task für Transformationen .....	367
13.4 Ein XML-Buildlog erzeugen .....	367
13.4.1 Stylesheets .....	368
13.4.2 Ausgabedateien .....	369
13.4.3 Das Buildlog nachbearbeiten .....	370
13.5 XML-Daten in Ant-Properties laden .....	371
13.6 JAXB und Castor .....	372
13.7 Zusammenfassung .....	372

## **Kapitel 14: Enterprise JavaBeans 373**

14.1 EJB im Überblick .....	373
14.1.1 Die viele Arten von Enterprise JavaBeans .....	373
14.1.2 EJB JAR .....	374
14.1.3 Anbieterspezifische Situationen .....	375
14.2 Ein einfacher EJB-Build .....	375
14.3 Die EJB-Tasks von Ant anwenden .....	375
14.4 <ejbjar> verwenden .....	376
14.4.1 Anbieterspezifische <ejbjar>-Verarbeitung .....	379
14.5 XDoclet für die EJB-Entwicklung verwenden .....	380
14.5.1 XDoclet-Subtasks .....	381
14.5.2 Die @tags von XDoclet .....	382
14.5.3 Verschiedene Application Server mit XDoclet unterstützen .....	382
14.5.4 Ant-Properties ersetzen .....	383
14.6 Middlegen .....	385
14.7 Deployment auf einem J2EE Application Server .....	388
14.8 Ein vollständiges EJB-Beispiel .....	388
14.9 Best Practices für EJB-Projekte .....	394
14.10 Zusammenfassung .....	394

## **Kapitel 15: Mit Webservices arbeiten 395**

15.1 Was sind Webservices, und was ist SOAP? .....	395
15.1.1 Das SOAP-API .....	396
15.1.2 Webservices zu Java hinzufügen .....	397
15.2 Eine SOAP-Client-Anwendung mit Ant erstellen .....	397
15.2.1 Unser Buildfile vorbereiten .....	398
15.2.2 Die Proxy-Klassen erstellen .....	399
15.2.3 Die SOAP-Proxy-Klassen verwenden .....	401
15.2.4 Den SOAP-Client kompilieren .....	402
15.2.5 Den SOAP-Client ausführen .....	402
15.2.6 Die Erstellung des SOAP-Clients überprüfen .....	403

15.3	Einen SOAP-Service mit Axis und Ant erstellen .....	403
15.3.1	Die einfache Methode, um einen Webservice zu erstellen .....	404
15.4	Webservices zu einer vorhandenen Webanwendung hinzufügen .....	407
15.4.1	Die Webanwendung konfigurieren .....	407
15.4.2	Die Libraries hinzufügen .....	408
15.4.3	SOAP-Services in den Build einfügen .....	408
15.4.4	Die Existenz benötigter Klassen auf dem Server testen .....	409
15.4.5	Den SOAP-Endpoint implementieren .....	410
15.4.6	Unseren Webservice deployen .....	410
15.5	Einen Client für unseren SOAP-Service schreiben .....	411
15.5.1	Die WSDL importieren .....	412
15.5.2	Die Tests implementieren .....	412
15.5.3	Den Java-Client schreiben .....	415
15.6	Was ist Interoperabilität, und warum ist sie ein Problem? .....	416
15.7	Einen C#-Client erstellen .....	417
15.7.1	Die Existenz von Klassen prüfen .....	418
15.7.2	Die WSDL nach C# importieren .....	418
15.7.3	Die C#-Client-Klasse schreiben .....	419
15.7.4	Den C#-Client erstellen .....	420
15.7.5	Den C#-Client ausführen .....	421
15.7.6	Überblick über den C#-Client-Build-Prozess .....	421
15.8	Die rigorose Methode, um einen Webservice zu erstellen .....	422
15.9	Die Webservice-Entwicklung überprüfen .....	423
15.10	Ant per SOAP aufrufen .....	423
15.11	Zusammenfassung .....	425

**Kapitel 16: Fortlaufende Integration 427**

16.1	Ant-Builds mit dem Betriebssystem planen .....	428
16.1.1	Die Windows-Methode .....	428
16.1.2	Die Unix-Version .....	428
16.1.3	Mit Skripts arbeiten .....	429
16.2	CruiseControl .....	429
16.2.1	Wie CruiseControl funktioniert .....	429
16.2.2	Den Build Runner starten .....	430
16.2.3	Build-Log-Berichte .....	436
16.2.4	E-Mail-Benachrichtigungen und Build-Zähler .....	437
16.2.5	Zusammenfassung von CruiseControl .....	437
16.2.6	Tipps und Tricks .....	437
16.2.7	Vor- und Nachteile von CruiseControl .....	438
16.3	Anthill .....	438
16.3.1	Anthill einrichten .....	439
16.3.2	Wie Anthill funktioniert .....	440
16.3.3	Zusammenfassung: Anthill .....	441
16.4	Gump .....	441
16.4.1	Gump installieren und ausführen .....	442
16.4.2	Wie Gump funktioniert .....	444
16.4.3	Zusammenfassung: Gump .....	445

16.5 Ein Vergleich von Werkzeugen zur fortlaufenden Integration .....	445
16.6 Zusammenfassung .....	446

## **Kapitel 17: Nativen Code entwickeln** **447**

17.1 Spezielle Probleme bei nativem Code .....	447
17.2 Vorhandene Build-Werkzeuge nutzen .....	448
17.2.1 Delegation an eine IDE .....	448
17.2.2 Mit Make arbeiten .....	449
17.3 Eine Einführung in den <cc>-Task .....	449
17.3.1 Die Tasks installieren .....	450
17.3.2 Einen Compiler hinzufügen .....	450
17.3.3 Eine schnelle Einführung in den <cc>-Task .....	451
17.4 Eine JNI-Library in Ant erstellen .....	452
17.4.1 Die Schritte, um eine JNI-Library zu erstellen .....	452
17.4.2 Den Java-Stub schreiben .....	454
17.4.3 Die C++-Klasse schreiben .....	455
17.4.4 Den C++-Source kompilieren .....	456
17.4.5 Die Library testen und deployen .....	458
17.5 Plattformübergreifend arbeiten .....	462
17.5.1 Den C++-Source migrieren .....	462
17.5.2 Das Buildfile erweitern .....	462
17.5.3 Die Migration testen .....	463
17.5.4 Den Code portieren .....	464
17.6 Den <cc>-Task ausführlicher studieren .....	465
17.6.1 Präprozessor-Makros definieren .....	465
17.6.2 Libraries beim Linken mit <libset> einbinden .....	466
17.6.3 Compiler und Linker konfigurieren .....	467
17.6.4 Linker anpassen .....	468
17.7 Native Libraries distribuieren .....	469
17.8 Zusammenfassung .....	469

## **Kapitel 18: Produktiver Einsatz (Deployment)** **471**

18.1 Das Problem der verschiedenen Application Server .....	472
18.1.1 Grundsätzlich verschiedenes Basisverhalten .....	472
18.1.2 Unterschiedliches Java Runtime-Verhalten .....	473
18.1.3 Mit verschiedenen API-Implementierungen umgehen .....	474
18.1.4 Anbieterspezifische Libraries .....	475
18.1.5 Deployment-Deskriptoren .....	476
18.1.6 Serverspezifische Deployment-Prozesse .....	476
18.1.7 Serverspezifische Verwaltung .....	476
18.2 Mit den Operatoren arbeiten .....	476
18.2.1 Anwendungsfälle für den laufenden Betrieb .....	476
18.2.2 Operative Tests .....	477
18.2.3 Defect Tracking beim operativen Einsatz .....	477
18.2.4 Den operativen Betrieb in den Build-Prozess integrieren .....	477

18.3	Deployment-Aufgaben mit Ant lösen	479
18.3.1	Arbeiten Sie mit einem einzigen Source-Baum	479
18.3.2	Erstellen Sie die Archivdateien mit einem vereinheitlichten Target	479
18.3.3	Ant serverseitig für ein Deployment ausführen	480
18.3.4	Den Upload- und Deployment-Prozess automatisieren	481
18.4	Eine Einführung in Deployment-Powertools von Ant	481
18.4.1	Der <copy>-Task	481
18.4.2	Der <serverdeploy>-Task	482
18.4.3	Remote-Kontrolle mit <telnet>	482
18.5	Einen Deployment-Prozess für den produktiven Einsatz erstellen	485
18.5.1	Der Plan	485
18.5.2	Die Verzeichnisstruktur	486
18.5.3	Die Konfigurationsdateien	486
18.5.4	Die Buildfiles	487
18.5.5	Das Remote-Buildfile build.xml	487
18.5.6	Das Buildfile für die Installation auf einem Server schreiben	488
18.5.7	Auf den Remote-Server hinaufladen	489
18.5.8	Das Remote-Deployment in Aktion	494
18.5.9	Der Deployment-Prozess im Überblick	495
18.6	Das Deployment auf speziellen Application Servern	496
18.6.1	Tomcat 4.0 und 4.1	496
18.6.2	BEA WebLogic	497
18.6.3	HP Bluestone Application Server	497
18.6.4	Andere Server	498
18.7	Das Deployment verifizieren	499
18.7.1	Die Zeitstempeldatei erstellen	499
18.7.2	Die Zeitstempeldatei in die Anwendung einfügen	500
18.7.3	Den Zeitstempel testen	501
18.8	Best Practices	502
18.9	Zusammenfassung	503

**Teil 3: Ant erweitern 505**

**Kapitel 19: Ant-Tasks schreiben 507**

19.1	Was genau ist ein Ant-Task?	508
19.1.1	Der einfachste Ant-Task der Welt	508
19.1.2	Einen Task in demselben Build kompilieren und anwenden	508
19.1.3	Der Lebenszyklus eines Tasks	509
19.2	Ein Leitfaden für das Ant-API	510
19.2.1	Task	510
19.2.2	Project	511
19.2.3	Path	512
19.2.4	Fileset	512
19.2.5	DirectoryScanner	512

19.2.6 EnumeratedAttribute .....	512
19.2.7 FileUtils .....	513
19.3 Wie Tasks Daten erhalten .....	513
19.3.1 Attribute setzen .....	514
19.3.2 Eingebettete Elemente unterstützen .....	519
19.3.3 Datentypen unterstützen .....	521
19.3.4 Formatfreien Body-Text zulassen .....	521
19.4 Eine grundlegende Unterklasse der Ant-Task erstellen .....	522
19.4.1 Ein Attribut zu einem Task hinzufügen .....	523
19.4.2 Elementtext verarbeiten .....	524
19.5 Mit einem Fileset arbeiten .....	525
19.6 Fehlerbehandlung .....	526
19.7 Ant-Tasks testen .....	527
19.8 Externe Programme ausführen .....	527
19.8.1 Prozessausgaben verarbeiten .....	530
19.8.2 Zusammenfassung der nativen Ausführung .....	530
19.9 Ein Java-Programm innerhalb eines Tasks ausführen .....	530
19.9.1 Ein Beispiel-Task, der ein Java-Programm per Forking ausführt .....	530
19.10 Beliebig benannte Elemente und Attribute unterstützen .....	533
19.11 Eine Task-Library erstellen .....	536
19.12 Mehrere Versionen von Ant unterstützen .....	537
19.13 Zusammenfassung .....	538

## **Kapitel 20: Weitere Möglichkeiten, Ant zu erweitern** **539**

20.1 Scripting innerhalb von Ant .....	539
20.1.1 Implizite Objekte, die für <script> bereitgestellt werden .....	540
20.1.2 Skripting-Zusammenfassung .....	542
20.2 Listener und Logger .....	542
20.2.1 Einen eigenen Listener schreiben .....	544
20.2.2 Die Logging-Funktionen von Log4j verwenden .....	547
20.2.3 Einen eigenen Logger schreiben .....	551
20.2.4 MailLogger verwenden .....	555
20.3 Einen eigenen Mapper entwickeln .....	556
20.4 Custom-Selektoren erstellen .....	557
20.4.1 Einen Custom-Selektor in einem Build verwenden .....	558
20.5 Einen Custom-Filter implementieren .....	560
20.5.1 Einen Custom-FilterReader codieren .....	561
20.6 Zusammenfassung .....	563

<b>Anhang A: Installation</b>	<b>565</b>
<b>Anhang B: XML für Ant – eine kurze Einführung</b>	<b>573</b>
<b>Anhang C: IDE-Integration</b>	<b>577</b>
<b>Anhang D: Die Elemente des Ant-Stils</b>	<b>583</b>
<b>Anhang E: Die Ant-Task-Referenz</b>	<b>601</b>
<b>Anhang F: Quellen</b>	<b>655</b>
<b>Anhang G: Stichwortverzeichnis</b>	<b>659</b>