

Inhaltsverzeichnis

1	Einleitung	1
1.1	Grundsätzliche Aufgaben eines Build-Management-Werkzeugs	4
1.2	Gradle – Kurzübersicht	6
1.3	Migrationspfade von anderen Build-Management-Werkzeugen	8
1.4	Installation	9
1.4.1	Kurzanleitung	9
1.4.2	Windows	10
1.4.3	Unix-Varianten	14
1.4.4	Mac	15
1.5	Gradle aufrufen	16
1.6	Erste Schritte	19
1.6.1	Unser erstes Skript für die Konfigurationsphase	20
1.6.2	Unser erstes Skript für die Ausführungsphase	20
1.6.3	Definition von Abhängigkeiten	21
1.7	Einbindung in Entwicklungsumgebungen	23
1.8	Weitergehende Informationen	24
2	Der Einstieg	25
2.1	Unser erstes Projekt	25
2.2	Hinzufügen von Tests	28
2.2.1	Abhängigkeiten und Konfigurationen in Gradle	29
2.2.2	Manipulation der Jar-Datei und des zugehörigen Manifests ..	32
2.3	Das Application-Plug-in	34
2.4	Verwendung von TestNG	35
2.5	Spock	37
2.6	Teststufen	39

2.7	Qualitätssicherung des Quelltextes	41
2.7.1	Das PMD-Plug-in	41
2.7.2	Einstellen des Report-Typs	44
2.7.3	Das Checkstyle-Plug-in	45
2.7.4	Verwendung von Ant zur Erzeugung des HTML-Reports	47
2.7.5	Das FindBugs-Plug-in	48
2.7.6	Das JDepend-Plug-in	49
2.8	Testabdeckung	50
2.9	Sonar	52
2.10	Kopieren der Artefakte in ein Repository	55
2.10.1	Veröffentlichen in ein lokales Verzeichnis	56
2.10.2	Versionierung unserer Artefakte	56
2.10.3	Veröffentlichen mit Maven	57
2.10.4	Signieren der Ergebnisse	58
2.11	Groovy-Projekte mit Gradle	60
2.12	Scala-Projekte mit Gradle	62
2.12.1	Unser erstes Scala-Programm	63
2.12.2	Tests für unser Scala-Programm	64
2.12.3	Weitergehende Konfiguration	66
2.13	Unsere erste Webapplikation	67
2.13.1	Die Struktur der Webapplikation	67
2.13.2	Weitere Dateien für die Webapplikation	68
2.13.3	Die Ausführung unserer Webapplikation	69
2.13.4	Ein einfacher Test unserer Webapplikation	70
3	Weitergehende Details	73
3.1	Groovy für Gradle	73
3.1.1	Allgemeine Syntax	74
3.1.2	Skripte	74
3.1.3	Operatorüberladung	75
3.1.4	Benannte Parameter für Methodenaufrufe	75
3.1.5	Closures	76
3.1.6	Meta Object Protocol	77
3.1.7	Erbauermuster (Builder-Pattern)	78
3.2	Das Build-Skript	80
3.3	Das Projekt	82
3.3.1	Deklaration von dynamischen Eigenschaften und Methoden	82
3.3.2	Externe Eigenschaften	84

3.4	Der Task	89
3.4.1	Der voreingestellte Task	91
3.4.2	Abhangigkeiten von Tasks	91
3.4.3	Beeinflussung der Ausfuhrung	92
3.4.4	Regelbasierte Tasks	95
3.4.5	Aktualitat der Task-Ergebnisse	96
3.5	Umgang mit Dateien	98
3.5.1	Einzelne Dateien	99
3.5.2	Mengen von Dateien	100
3.5.3	Hierarchien von Dateien	102
3.5.4	Operationen auf Dateien	106
3.6	Ausfuhrung externer Kommandos	107
3.7	Haufig benutzte Task-Typen	108
3.7.1	Umgang mit Dateien	108
3.7.2	Archivierung von Dateien	110
3.7.3	Benennung von Ergebnissen	111
3.7.4	Ausfuhrung externer Programme	112
3.7.5	Erzeugung initialer Installationsskripte	113
3.8	Logging mit Gradle	114
3.9	Umgang mit Quellen	116
3.9.1	Hinzufugen von Quellen	117
3.9.2	Integration von SourceSets in den Build	118
3.10	Testen im Detail	123
3.11	Verwendung von Plug-ins	125
3.12	Externe Abhangigkeiten	128
3.13	Gradle GUI	134
3.14	Konfiguration des Build-Environments	135
3.14.1	Grundsatzliche Konfigurationseigenschaften	135
3.14.2	Verwendung eines Web-Proxys	136
3.15	Gradle-Daemon	137
3.16	Laufzeiten des Builds	138
3.17	Online-Dokumentation	140
4	Migration zu Gradle	141
4.1	Ant	141
4.1.1	Gradle und Ant	141
4.1.2	Verwendung von Ant-Tasks	142

4.1.3	Verwendung zusätzlicher Bibliotheken	143
4.1.4	Verwendung eigener Ant-Tasks	145
4.1.5	Verwendung vollständiger Ant-Build-Dateien	148
4.2	Maven	149
4.2.1	Vergleich der Build-Skripte	150
4.2.2	Konvertierung eines Maven-Build-Skripts	150
5	Multiprojekt-Builds	157
5.1	Die Ausgangsbasis	158
5.1.1	Das Teilprojekt <code>service</code>	158
5.1.2	Das Teilprojekt <code>client</code>	161
5.2	Der Umbau in einen Multiprojekt-Build	162
5.2.1	Die Datei <code>settings.gradle</code>	162
5.2.2	Hinzufügen des ersten Teilprojekts	164
5.2.3	Konfiguration durch das Elternprojekt	166
5.2.4	Der vernünftige Mittelweg	167
5.2.5	Hinzufügen des zweiten Teilprojekts	168
5.3	Weitere Funktionalität für Multiprojekte	170
5.3.1	Abhängigkeit von Tasks anderer Teilprojekte	170
5.3.2	Abhängigkeit in der Konfigurationsphase	170
5.3.3	Projektübergreifende Definition von Tasks	171
5.4	Verwendung eines Initialisierungsskripts	172
6	Integration mit dem Build-Server	173
6.1	Integration mit Jenkins	173
6.2	Integration mit JetBrains TeamCity	181
7	Erweiterung von Gradle	187
7.1	Arten von Erweiterungen	187
7.2	Orte für Erweiterungen	188
7.2.1	Das Build-Skript	188
7.2.2	Externe Skripte	188
7.2.3	Das Verzeichnis <code>buildSrc</code>	189
7.2.4	Eigene Projekte	190
7.2.5	Wahl der Variante	190
7.3	Eigene Tasks	191
7.3.1	Annotationen für unsere eigenen Task-Klassen	191
7.3.2	Ein einfaches Beispiel	192
7.3.3	Ein komplettes Beispiel	193

7.3.4	Unser Task im Verzeichnis <code>buildSrc</code>	195
7.3.5	Unser Task im eigenen Projekt	197
7.3.6	Verwendung von Tasks aus Bibliotheken	197
7.4	Eigene Plug-ins	199
7.4.1	Ein einfaches Beispiel	199
7.4.2	Ein komplexeres Beispiel	200
7.4.3	Integration mit dem Build-Skript	202
7.4.4	Ein komplettes Beispiel mit Konventionsobjekten	208
7.4.5	Das komplette Beispiel mit Erweiterungsobjekten	212
7.4.6	Unser Plug-in im Verzeichnis <code>buildSrc</code>	215
7.4.7	Unser Plug-in im eigenen Projekt	216
8	Praktische Vorgehensweisen	219
8.1	Explizite Vorgaben	219
8.2	Multiprojektumgebungen	220
8.3	Einsatz von Ant-Tasks	221
8.4	Quelltext im Gradle-Skript	221
8.5	Konfiguration der Tests	222
8.6	Ausgaben Ihrer Skripte	222
9	Weitere Plug-ins	223
9.1	Verwendung des Tomcat als Webcontainer	223
9.2	Integration mit Git	225
9.2.1	Die Verwendung von <code>GitClone</code>	226
9.2.2	Erzeugen eines Entwicklungs-Branch	227
9.2.3	Verwendung von Feature-Branche	227
9.2.4	Übernahme von Daten	228
9.2.5	Zusammenführen von Änderungen	229
9.2.6	Setzen von Tags	229
9.2.7	Weitergehende Funktionalität	230
9.3	Erzeugung von Versionsnummern	230
9.3.1	Verwendung des Quelltextverwaltungssystems	231
9.4	Weitere Plug-ins	233
10	Zusammenfassung	235
11	Literatur	237
Index		241