

Inhaltsverzeichnis

| | |
|---|-----------|
| Einleitung | 15 |
| Für wen dieses Buch gedacht ist | 16 |
| Wie dieses Buch aufgebaut ist | 17 |
| Konventionen | 18 |
| Voraussetzungen | 18 |
| Den Code herunterladen | 18 |
| Kontakt mit den Autoren aufnehmen | 18 |
| Über die Autoren | 19 |
| Über die technischen Gutachter | 20 |
| Danksagungen | 21 |
| | |
| I Das Wesen von Grails | 23 |
| I.I Einfachheit und Power | 23 |
| I.2 Grails, die Plattform | 25 |
| I.3 Leben im Java-Ökosystem | 25 |
| I.4 Grails installieren | 26 |
| I.5 Ihre erste Anwendung erstellen | 27 |
| I.5.1 Schritt 1: Die Anwendung erstellen | 28 |
| I.5.2 Schritt 2: Einen Controller erstellen | 29 |
| I.5.3 Schritt 3: Eine Meldung ausgeben | 30 |
| I.5.4 Schritt 4: Den Code testen | 31 |
| I.5.5 Schritt 5: Die Tests ausführen | 32 |
| I.5.6 Schritt 6: Die Anwendung ausführen | 33 |
| I.6 Zusammenfassung | 34 |
| | |
| 2 Der Einstieg in Grails | 35 |
| 2.1 Was ist Scaffolding? | 35 |
| 2.2 Eine Domain erstellen | 35 |
| 2.3 Dynamisches Scaffolding | 37 |
| 2.3.1 Die Create-Operation | 39 |
| 2.3.2 Die Read-Operation | 40 |
| 2.3.3 Die Update-Operation | 42 |
| 2.3.4 Die Delete-Operation | 44 |

| | | |
|-------|--|----|
| 2.4 | Statisches Scaffolding | 44 |
| 2.4.1 | Einen Controller generieren | 44 |
| 2.4.2 | Views generieren | 48 |
| 2.5 | Eine freundliche Umgebung einrichten | 50 |
| 2.6 | Datenquellen konfigurieren | 51 |
| 2.6.1 | Die Datei »DataSource.groovy« | 51 |
| 2.6.2 | Eine MySQL-Datenbank konfigurieren | 54 |
| 2.6.3 | Eine JNDI-Datenquelle konfigurieren | 56 |
| 2.6.4 | Unterstützte Datenbanken | 56 |
| 2.7 | Die Anwendung deployen | 57 |
| 2.7.1 | Deployment mit run-war | 58 |
| 2.7.2 | Deployment mit einer WAR-Datei | 58 |
| 2.8 | Zusammenfassung | 59 |
| | | |
| 3 | Domain-Klassen | 61 |
| 3.1 | Felder in der Datenbank speichern | 61 |
| 3.2 | Domain-Klassen validieren | 62 |
| 3.3 | Anwendungsspezifische Validatoren | 65 |
| 3.4 | Transiente Properties | 66 |
| 3.5 | Das Datenbank-Mapping anpassen | 67 |
| 3.6 | Beziehungen definieren | 69 |
| 3.7 | Klassen per Vererbung erweitern | 72 |
| 3.8 | Objekte einbetten | 74 |
| 3.9 | Domain-Klassen testen | 76 |
| 3.10 | Zusammenfassung | 78 |
| | | |
| 4 | Controller verstehen | 79 |
| 4.1 | Controller definieren | 79 |
| 4.1.1 | Die Standardaction setzen | 80 |
| 4.1.2 | Logging | 81 |
| 4.1.3 | Exceptions protokollieren | 82 |
| 4.1.4 | Request-Attribute abfragen | 82 |
| 4.1.5 | Mit Controller-Scopes arbeiten | 83 |
| 4.1.6 | Den flash-Scope verstehen | 84 |
| 4.1.7 | Auf Request-Parameter zugreifen | 86 |
| 4.1.8 | Text darstellen | 86 |
| 4.1.9 | Eine Anfrage umlenken (Redirects) | 87 |
| 4.2 | Ein Model erstellen | 88 |
| 4.3 | Eine View darstellen | 89 |

| | | |
|-------|--|-----|
| 4.3.1 | Die Standard-View ermitteln | 89 |
| 4.3.2 | Eine anwendungsspezifische View auswählen | 89 |
| 4.3.3 | Templates darstellen | 90 |
| 4.4 | Datenbindung | 90 |
| 4.4.1 | Eingabedaten validieren | 91 |
| 4.4.2 | Das Errors-API und Controller | 92 |
| 4.4.3 | Datenbindung an mehrere Domain-Objekte | 93 |
| 4.4.4 | Datenbindung mit der »bindData«-Methode | 94 |
| 4.4.5 | Datenbindung und Beziehungen | 94 |
| 4.5 | Mit Befehlsobjekten arbeiten (Command Objects) | 96 |
| 4.5.1 | Befehlsobjekte definieren | 96 |
| 4.5.2 | Befehlsobjekte anwenden | 97 |
| 4.5.3 | Eine imperative Lösung implementieren | 98 |
| 4.5.4 | Die Vorteile einer deklarativen Syntax nutzen | 98 |
| 4.6 | Controller-IO | 99 |
| 4.6.1 | Datei-Uplands | 99 |
| 4.6.2 | Den InputStream der Anfrage lesen | 102 |
| 4.6.3 | Eine binäre Antwort schreiben | 102 |
| 4.7 | Mit einfachen Interzeptoren arbeiten | 102 |
| 4.7.1 | Before Advice | 103 |
| 4.7.2 | After Advice | 104 |
| 4.8 | Controller testen | 104 |
| 4.9 | Controller in Aktion | 106 |
| 4.9.1 | Die gTunes-Homepage erstellen | 106 |
| 4.9.2 | Die User-Domain-Klasse hinzufügen | 107 |
| 4.9.3 | Ein Login-Formular hinzufügen | 108 |
| 4.9.4 | Die Registrierung implementieren | 110 |
| 4.9.5 | Den Registrierungscode testen | 113 |
| 4.9.6 | Den Login-Prozess des Benutzers implementieren | 114 |
| 4.9.7 | Den Login-Prozess testen | 116 |
| 4.10 | Zusammenfassung | 118 |
| 5 | Views verstehen | 119 |
| 5.1 | Die Grundlagen | 119 |
| 5.1.1 | Das Model verstehen | 120 |
| 5.1.2 | Seitendirektiven | 121 |
| 5.1.3 | Groovy-Scriptlets | 121 |
| 5.1.4 | GSP als GStrings | 122 |

| | | |
|----------|--|------------|
| 5.2 | Eingebaute Grails-Tags | 123 |
| 5.2.1 | Variablen mit Tags setzen | 123 |
| 5.2.2 | Logische Tags | 124 |
| 5.2.3 | Iterative Tags | 125 |
| 5.2.4 | Filter und Iterationen | 126 |
| 5.3 | Dynamische Tags | 128 |
| 5.3.1 | Linking-Tags | 129 |
| 5.3.2 | Formulare und Felder erstellen | 131 |
| 5.3.3 | Validierung und Fehlerbehandlung | 135 |
| 5.3.4 | Views paginieren | 136 |
| 5.3.5 | GSP-Templates darstellen | 143 |
| 5.4 | Anwendungsspezifische Tags erstellen | 147 |
| 5.4.1 | Eine Tag Library erstellen | 147 |
| 5.4.2 | Grundlagen anwendungsspezifischer Tags | 148 |
| 5.4.3 | Anwendungsspezifische Tags testen | 149 |
| 5.5 | Zusammenfassung | 151 |
| 6 | URLs mappen | 153 |
| 6.1 | Das standardmäßige URL-Mapping | 153 |
| 6.2 | Statischen Text in ein URL-Mapping einfügen | 154 |
| 6.3 | Controller und Action separat angeben | 154 |
| 6.4 | Parameter in ein Mapping einbetten | 155 |
| 6.5 | Zusätzliche Parameter spezifizieren | 157 |
| 6.6 | URL-Mappings für Views | 157 |
| 6.7 | Constraints auf URL-Mappings anwenden | 158 |
| 6.8 | Platzhalter in Mappings | 159 |
| 6.9 | Mappings von HTTP-Anfragemethoden | 160 |
| 6.10 | Mappings von HTTP-Antwortcodes | 161 |
| 6.11 | Das umgekehrte URL-Mapping nutzen (Rewrites) | 163 |
| 6.12 | Klassen mit mehreren URL-Mappings definieren | 164 |
| 6.13 | Beliebige URLs umschreiben | 164 |
| 6.14 | Benannte URL-Mappings definieren | 165 |
| 6.15 | URL-Mappings testen | 165 |
| 6.16 | Zusammenfassung | 168 |

| | | |
|----------|---|-----|
| 7 | Internationalisierung | 169 |
| 7.1 | Meldungen lokalisieren | 169 |
| 7.1.1 | Benutzermeldungen definieren | 169 |
| 7.1.2 | Meldungswerte abrufen | 170 |
| 7.1.3 | URL-Mappings für die Internationalisierung | 173 |
| 7.2 | Mit parametrisierten Meldungen arbeiten | 173 |
| 7.2.1 | Mit »java.text.MessageFormat« arbeiten | 173 |
| 7.2.2 | Parametrisierte Meldungen mit dem »message«-Tag | 174 |
| 7.2.3 | Mit parametrisierten Meldungen validieren | 175 |
| 7.3 | Die »messageSource«-Bean | 177 |
| 7.4 | Zusammenfassung | 179 |
| 8 | Ajax | 181 |
| 8.1 | Die Grundlagen von Ajax | 181 |
| 8.2 | Ajax in Aktion | 183 |
| 8.3 | Den Ajax-Provider ändern | 184 |
| 8.4 | Formulare asynchron übermitteln | 184 |
| 8.5 | Code vor und nach einem Aufruf ausführen | 187 |
| 8.6 | Events verarbeiten | 188 |
| 8.7 | Ajax Remote Linking | 189 |
| 8.8 | Effekte und Animation hinzufügen | 201 |
| 8.9 | Ajax-fähige Formularfelder | 202 |
| 8.10 | Ajax und die Performance | 206 |
| 8.11 | Zusammenfassung | 207 |
| 9 | GORM | 209 |
| 9.1 | Grundlagen der Persistenz | 209 |
| 9.1.1 | Objekte lesen | 210 |
| 9.1.2 | Auflisten, sortieren und zählen | 210 |
| 9.1.3 | Speichern, aktualisieren und löschen | 211 |
| 9.2 | Beziehungen | 212 |
| 9.2.1 | Methoden zur Beziehungsverwaltung | 214 |
| 9.2.2 | Transitive Persistenz | 214 |
| 9.3 | Abfragen | 215 |
| 9.3.1 | Dynamische Finder | 215 |
| 9.3.2 | Kriterienabfragen (Criteria Queries) | 217 |
| 9.3.3 | Query by Example (QBE) | 222 |
| 9.3.4 | HQL und SQL | 222 |

Inhaltsverzeichnis

| | | |
|--------|--|-----|
| 9.3.5 | Benannte Abfragen (Named Query) | 223 |
| 9.3.6 | Paginierung | 225 |
| 9.4 | GORM konfigurieren | 226 |
| 9.4.1 | SQL-Logging | 226 |
| 9.4.2 | Einen speziellen Dialekt einstellen | 226 |
| 9.5 | Die Semantik von GORM | 228 |
| 9.5.1 | Die Hibernate-Session | 228 |
| 9.5.2 | Session-Management und Flushing | 229 |
| 9.5.3 | Die Session erhalten | 231 |
| 9.5.4 | Automatisches Session-Flushing | 233 |
| 9.6 | Transaktionen in GORM | 234 |
| 9.7 | Detached Objekte | 236 |
| 9.7.1 | Der Persistenz-Lebenszyklus | 236 |
| 9.7.2 | Detached Objekte wieder ankoppeln | 237 |
| 9.7.3 | Zustandsänderungen kombinieren | 239 |
| 9.8 | Die Performance optimieren | 239 |
| 9.8.1 | Eager- vs. Lazy-Beziehungen | 240 |
| 9.8.2 | Batch-Abrufe | 242 |
| 9.8.3 | Caching | 243 |
| 9.8.4 | Vererbungsstrategien | 246 |
| 9.9 | Locking-Strategien | 246 |
| 9.10 | GORM-Events und automatische Zeitstempel | 248 |
| 9.11 | Zusammenfassung | 249 |
| | | |
| 10 | Services | 251 |
| 10.1 | Service-Grundlagen | 251 |
| 10.2 | Services und Dependency Injection | 252 |
| 10.3 | Services in Aktion | 253 |
| 10.3.1 | Einen Service definieren | 254 |
| 10.3.2 | Einen Service benutzen | 255 |
| 10.4 | Transaktionen | 256 |
| 10.5 | Den Scope von Services festsetzen | 258 |
| 10.6 | Services testen | 258 |
| 10.7 | Services veröffentlichen | 259 |
| 10.8 | Zusammenfassung | 263 |

| | | |
|---------------|--|-----|
| II | Grails integrieren | 265 |
| II.1 | Konfiguration in Grails | 265 |
| II.1.1 | Grundlagen der Konfiguration | 265 |
| II.1.2 | Umgebungsspezifische Konfiguration | 266 |
| II.1.3 | Logging-Konfiguration | 266 |
| II.1.4 | Stacktrace-Filter | 269 |
| II.1.5 | Ausgelagerte Konfiguration | 269 |
| II.1.6 | Globale Standardwerte für GORM konfigurieren | 270 |
| II.2 | Das Buildsystem von Grails verstehen | 271 |
| II.2.1 | Gant-Skripts erstellen | 272 |
| II.2.2 | Befehlszeilenvariablen | 274 |
| II.2.3 | Befehlszeilenargumente parsen | 275 |
| II.2.4 | Skripts dokumentieren | 276 |
| II.2.5 | Mehr von Grails wiederverwenden | 277 |
| II.2.6 | Grails von der Befehlszeile aus bootstrappen | 277 |
| II.2.7 | Gant in Aktion | 278 |
| II.3 | Integration mit Apache Ant | 285 |
| II.4 | Dependencies mit Ivy auflösen | 287 |
| II.5 | Code Coverage mit Cobertura | 292 |
| II.6 | Continuous Integration mit Hudson | 293 |
| II.7 | Die IDE Ihrer Wahl unterstützen | 297 |
| II.7.1 | IntelliJ | 297 |
| II.7.2 | NetBeans | 298 |
| II.7.3 | Eclipse | 299 |
| II.7.4 | TextMate | 299 |
| II.7.5 | Remote Debugging mit einer IDE | 301 |
| II.8 | Integration mit E-Mail-Servfern | 302 |
| II.9 | Jobs planen | 306 |
| II.9.1 | Das Quartz-Plugin installieren | 306 |
| II.9.2 | Einfache Jobs | 307 |
| II.9.3 | Cron Jobs | 307 |
| II.9.4 | Mit dem Scheduler interagieren | 310 |
| II.9.5 | Jobs planen | 311 |
| II.9.6 | Jobs anhalten und fortsetzen | 311 |
| II.9.7 | Einen Job auslösen | 311 |
| II.9.8 | Jobs hinzufügen und entfernen | 312 |
| II.9.9 | Jobs in Aktion | 312 |

| | | |
|---------|---|-----|
| II.10 | Deployment | 316 |
| II.10.1 | Deployment mit Grails | 317 |
| II.10.2 | Deployment in einem Container | 317 |
| II.10.3 | Anwendungsversionierung und -metadaten | 318 |
| II.10.4 | Die WAR-Datei anpassen | 318 |
| II.10.5 | Daten mittels BootStrap-Klasse der Datenbank hinzufügen | 319 |
| II.11 | Zusammenfassung | 320 |
| | | |
| I2 | Plugins | 321 |
| I2.1 | Plugin-Grundlagen | 321 |
| I2.1.1 | Plugins entdecken | 321 |
| I2.1.2 | Plugins installieren | 323 |
| I2.1.3 | Lokale Plugins | 323 |
| I2.1.4 | Plugins erstellen | 324 |
| I2.1.5 | Plugin-Metadaten | 324 |
| I2.1.6 | Anwendungssartefakte bereitstellen | 326 |
| I2.1.7 | Plugin-Hooks | 326 |
| I2.1.8 | Plugin-Variablen | 327 |
| I2.1.9 | Anwendungsspezifische Artefakttypen | 329 |
| I2.1.10 | Spring-Beans zur Verfügung stellen | 331 |
| I2.1.11 | Konventionen für dynamische Spring-Beans | 334 |
| I2.1.12 | Verhalten per Metaprogrammierung erweitern | 335 |
| I2.1.13 | Plugin-Events und Neuladen der Anwendung | 336 |
| I2.1.14 | Den generierten WAR-Deskriptor modifizieren | 339 |
| I2.1.15 | Grails-Plugins paketieren und verteilen | 340 |
| I2.1.16 | Lokale Plugin-Repositories | 341 |
| I2.2 | Plugins in Aktion | 342 |
| I2.2.1 | Das Verhalten mit Plugins erweitern | 342 |
| I2.2.2 | Anwendungen mit Plugins modularisieren | 347 |
| I2.3 | Zusammenfassung | 356 |
| | | |
| I3 | Sicherheit | 357 |
| I3.1 | Schutz vor Angriffen | 357 |
| I3.1.1 | SQL- oder HQL-Injektion | 358 |
| I3.1.2 | Groovy-Injektion | 359 |
| I3.1.3 | Cross-Site Scripting (XSS) | 359 |
| I3.1.4 | XSS- und URL-Escaping | 361 |
| I3.1.5 | Denial of Service (DoS) | 362 |

| | | |
|-----------|--|------------|
| 13.1.6 | Verwundbarkeit durch Datenbindung | 362 |
| 13.1.7 | Cross-Site Request Forgery | 364 |
| 13.2 | Mit dynamischen Codecs arbeiten | 365 |
| 13.3 | Authentifizierung und Autorisierung | 367 |
| 13.4 | Grails-Filter | 367 |
| 13.5 | Das Shiro-Plugin (früher JSecurity) | 370 |
| 13.5.1 | Authentifizierungs-Realms | 370 |
| 13.5.2 | Subjekte und Principals | 371 |
| 13.5.3 | Rollen und Berechtigungen | 371 |
| 13.5.4 | Shiro in Aktion | 372 |
| 13.6 | Den Zugriff mit URL-Mappings einschränken | 395 |
| 13.7 | Zusammenfassung | 397 |
| 14 | Webservices | 399 |
| 14.1 | REST | 400 |
| 14.1.1 | RESTful URL-Mappings | 400 |
| 14.1.2 | Content-Negotiation | 402 |
| 14.1.3 | Content-Negotiation mit dem ACCEPT-Header | 402 |
| 14.1.4 | Der ACCEPT-Header und ältere Browser | 406 |
| 14.1.5 | Content-Negotiation mit dem CONTENT_TYPE-Header .. | 407 |
| 14.1.6 | Content-Negotiation mit Dateinamenserweiterungen | 408 |
| 14.1.7 | Content-Negotiation mit einem Request-Parameter | 408 |
| 14.1.8 | Content-Negotiation und die View | 409 |
| 14.1.9 | Objekte in XML-Code packen | 409 |
| 14.1.10 | Objekte in JSON packen | 412 |
| 14.1.11 | XML oder JSON entpacken | 414 |
| 14.1.12 | REST und Sicherheit | 420 |
| 14.2 | Atom und RSS | 420 |
| 14.2.1 | RSS- und Atom-Feeds erstellen | 421 |
| 14.2.2 | RSS- und Atom-Feeds erkennen | 423 |
| 14.3 | SOAP | 424 |
| 14.3.1 | SOAP-Webservices über Plugins installieren | 426 |
| 14.3.2 | SOAP vom Client aus aufrufen | 429 |
| 14.4 | Zusammenfassung | 431 |
| 15 | Mit Spring arbeiten | 433 |
| 15.1 | Spring-Grundlagen | 433 |
| 15.2 | Spring und Grails | 435 |

Inhaltsverzeichnis

| | | |
|-----------|---|------------|
| 15.2.1 | Dependency Injection und Grails | 435 |
| 15.2.2 | Die BeanBuilder-DSL | 435 |
| 15.2.3 | Nutzung der Spring-Annotationen | 444 |
| 15.3 | Spring in Aktion | 445 |
| 15.3.1 | JMS mit Spring JMS integrieren | 445 |
| 15.3.2 | Groovy und Java mit Spring kombinieren | 459 |
| 15.4 | Zusammenfassung | 462 |
| 16 | Legacy-Integration mit Hibernate | 463 |
| 16.1 | Legacy-Mapping mit der ORM-DSL | 463 |
| 16.1.1 | Tabellen- und Spalten-Namen mappen | 464 |
| 16.1.2 | Beziehungen auf eine Datenbank abbilden | 465 |
| 16.1.3 | Hibernate-Typen | 468 |
| 16.1.4 | Den Datenbank-Identity-Generator ändern | 472 |
| 16.1.5 | Zusammengesetzte IDs | 474 |
| 16.2 | Mapping mit Hibernate XML | 475 |
| 16.3 | EJB-3-konformes Mapping | 478 |
| 16.4 | Constraints bei POJO-Entities verwenden | 484 |
| 16.5 | Zusammenfassung | 485 |
| A | Die Groovy-Sprache | 487 |
| A.1 | Groovy und Java: ein Vergleich | 487 |
| A.1.1 | Übereinstimmungen | 488 |
| A.1.2 | Unterschiede | 488 |
| A.2 | Die Grundlagen | 489 |
| A.2.1 | Klassen deklarieren | 490 |
| A.2.2 | Assertions auf Sprachebene | 490 |
| A.2.3 | Groovy-Strings | 491 |
| A.2.4 | Closures | 493 |
| A.2.5 | Listen, Maps und Ranges | 494 |
| A.2.6 | Expando-Objekte | 496 |
| A.2.7 | Ranges | 497 |
| A.3 | Power-Funktionen von Groovy | 498 |
| A.3.1 | Alles ist ein Objekt | 498 |
| A.3.2 | Metaprogrammierung | 502 |
| A.3.3 | Builder verstehen | 507 |
| A.4 | Zusammenfassung | 509 |
| | Stichwortverzeichnis | 511 |