

Inhaltsübersicht

1	Einleitung	1
Teil I	Grundlagen	5
2	Hallo, Spring Boot	7
3	Spring-Grundlagen	15
4	Spring-Boot-Grundlagen	27
5	Konfiguration	35
6	Build Management mit Spring Boot	53
7	Einführung ins Testen	61
8	Troubleshooting einer Spring-Boot-Anwendung	67
Teil II	Anwendungsfälle	85
9	Einen REST-Service entwickeln	87
10	Das Reactor-Framework verwenden	91
11	Eine GraphQL-API entwickeln	95
12	Integration einer SPA mit Spring Boot	105
13	Ein serverbasiertes Web-Frontend entwickeln	119
14	Eine Datenbank anbinden	129
15	Eine CLI-Anwendung entwickeln	133
16	Architektur-Governance mit Spring Boot	147

Teil III	Referenz	157
17	Testing	159
18	Spring Reactive	189
19	Spring Web MVC	211
20	HTTP-Clients mit Spring	225
21	GraphQL	231
22	Spring Boot Developer Tools	255
23	Events	259
24	Caching	269
25	Messaging	275
26	Spring Data	289
27	Spring Cloud Config	315
28	Spring Security	323
29	Observability	339
30	Docker-Images mit Spring Boot	377
31	Native Images mit Spring Boot	385
32	Spring Boot erweitern	393
33	Coordinated Restore at Checkpoint (CRaC)	407
34	Migration von Spring Boot 2 zu Spring Boot 3	413
35	Ausblick	425

Inhaltsverzeichnis

1	Einleitung	1
1.1	Warum Spring Boot?	1
1.2	Für wen ist dieses Buch?	2
1.3	Aufbau des Buchs	2
1.4	Codebeispiele	3
Teil I	Grundlagen	5
2	Hallo, Spring Boot	7
2.1	Einleitung	7
2.2	JDK installieren	7
2.3	Kickstart mit dem Spring Initializr	8
2.4	Projektstruktur	9
2.5	Die Anwendung bauen	10
2.6	Die Anwendung starten	11
2.7	Einen REST-Controller bauen	12
3	Spring-Grundlagen	15
3.1	Dependency Injection und Inversion of Control	15
3.1.1	Inversion of Control	15
3.1.2	Dependency Injection	16
3.2	Der Spring Application Context	19
3.3	Application Context mit XML konfigurieren	20
3.4	Java-Konfiguration im Detail	21
3.4.1	@Configuration und @Bean	21
3.4.2	@Component und @ComponentScan	22
3.4.3	@Configuration und @ComponentScan kombinieren ...	23
3.5	Was haben wir vom Spring Container?	25

4	Spring-Boot-Grundlagen .	27
4.1	Bootstrapping	27
4.2	Den Application Context beeinflussen	29
4.3	Embedded Webserver	31
4.4	Dependency Management	32
4.5	Integrationen	32
4.6	Produktionsbetrieb	33
5	Konfiguration	35
5.1	Warum Konfiguration?	35
5.2	Konfigurationsparameter	36
5.2.1	Konfigurationsparameter definieren	36
5.2.2	Parameter als String injizieren	38
5.2.3	Parameter typischer injizieren	39
5.2.4	Parameter validieren	41
5.2.5	Default-Werte definieren	41
5.3	Profile	42
5.3.1	Konfigurationsparameter für ein Profil definieren	42
5.3.2	Ein Profil aktivieren	44
5.3.3	Konfigurationswerte aus unterschiedlichen Quellen kombinieren	45
5.4	Das Environment-Bean	45
5.5	Best Practices für Konfigurationsmanagement	46
5.5.1	Konfigurationsdateien im JAR	46
5.5.2	Secrets als Umgebungsvariablen definieren	47
5.5.3	Umgebungsvariablen explizit definieren	47
5.5.4	Profile nur für Umgebungen	48
5.5.5	Konfiguration von Default-Werten	51
6	Build Management mit Spring Boot	53
6.1	Überblick	53
6.2	Gradle oder Maven?	53
6.3	Spring Boot »Fat JAR«	54
6.4	Das Gradle Plugin	56
6.5	Das Maven Plugin	57
6.6	Dependency Management mit Spring Boot	58
6.6.1	Spring-Boot-BOM mit Gradle konsumieren	59
6.6.2	Spring-Boot-BOM mit Maven konsumieren	59

7	Einführung ins Testen	61
7.1	Testen – wieso, weshalb, warum?	61
7.2	Spring Boot Starter Test	62
7.3	Unit-Tests	62
7.4	SpringExtension (JUnit 5)	63
7.5	Integrationstests mit Spring	63
7.5.1	@SpringBootTest	65
7.5.2	Ausblick auf Slice-Annotationen	66
8	Troubleshooting einer Spring-Boot-Anwendung	67
8.1	Spring-Boot-Magie	67
8.2	Troubleshooting-Werkzeuge	67
8.2.1	Lokal reproduzieren	68
8.2.2	Debug-Modus	69
8.2.3	Actuator-Endpoints	69
8.2.4	Logging	69
8.2.5	Hooks	70
8.3	Troubleshooting-Anwendungsfälle	71
8.3.1	Welche Beans stehen mir zur Verfügung?	71
8.3.2	Wo kommt eine Bean her?	72
8.3.3	Warum ist meine Bean nicht im Application Context? ...	73
8.3.4	Warum ist eine Bean doppelt im Application Context? ...	75
8.3.5	Wie ist meine Anwendung konfiguriert?	76
8.3.6	Welche Konfigurationsparameter werden (nicht) genutzt?	77
8.3.7	Welche Endpoints stellt meine Anwendung zur Verfügung?	81
8.3.8	Warum startet meine Anwendung nicht?	83
Teil II	Anwendungsfälle	85
9	Einen REST-Service entwickeln	87
9.1	Was ist REST?	87
9.2	Codebeispiel	88
9.3	Endpoints	89
9.4	Request Body und Parameter	89
9.5	Fehlerbehandlung	90

10	Das Reactor-Framework verwenden	91
10.1	Was ist das Reactor-Framework?	91
10.2	Codebeispiel	92
10.3	Andere reaktive Operatoren	93
11	Eine GraphQL-API entwickeln	95
11.1	GraphQL in Kürze	95
11.2	Die Beispielanwendung	96
11.3	Erstellung der Anwendung	96
11.4	Abbildung des Datenmodells in GraphQL	98
11.5	Implementierung des Controllers	98
11.5.1	Query Mapping	99
11.5.2	Schema Mapping	100
11.5.3	Mutation Mapping	101
11.6	Implementierung einer Subscription	102
12	Integration einer SPA mit Spring Boot	105
12.1	Herausforderungen von SPAs	105
12.1.1	Handhabung von URLs	105
12.1.2	Integration unterschiedlicher Toolings	106
12.2	Mögliche Varianten für ein Deployment	107
12.2.1	Integration in das Spring-Boot-JAR	107
12.2.2	Unabhängige Deployments des Frontends und Backends	108
12.3	Ein Spring Boot Backend mit React Frontend	110
12.3.1	Verzeichnisstruktur	110
12.3.2	Konfiguration des Builds	111
12.3.3	Integration einer REST-API in das Frontend	114
12.3.4	Unterstützung von Deep-Links mit der History-API	117
13	Ein serverbasiertes Web-Frontend entwickeln	119
13.1	Warum serverseitig?	119
13.2	Die Beispielanwendung	120
13.3	»Hello World« mit Thymeleaf	120
13.4	Formulardaten verarbeiten	121
13.5	Interaktivität mit HTMX	124
13.6	Spring Boot Developer Tools	128
13.7	Weitere Ressourcen	128

14	Eine Datenbank anbinden	129
14.1	Datenbanken	129
14.2	Codebeispiel	130
14.3	Spring Data Repositories	131
14.4	Transaktionen	132
15	Eine CLI-Anwendung entwickeln	133
15.1	Eine ADR-Management-Anwendung	133
15.2	Erstellen des Projekts	134
15.2.1	Die ADR-API	134
15.3	Ein erstes Kommando	136
15.4	Registrieren von Kommandos	137
15.4.1	Programmatische Registrierung von Kommandos	138
15.4.2	Kommandos mit Annotationen definieren	140
15.4.3	AvailabilityProvider	142
15.4.4	Anzeige eines eigenen Prompts	143
15.5	Ausführung im interaktiven oder Kommando-Modus	145
15.6	Alternativen zu Spring Shell	146
16	Architektur-Governance mit Spring Boot	147
16.1	Warum Architektur-Governance?	147
16.2	Komponentenbasierte Architektur	148
16.3	Komponentenbasierte Architektur mit Spring Boot	149
16.4	Governance mit ArchUnit	152
16.5	Spring Modulith	153
Teil III	Referenz	157
17	Testing	159
17.1	Überblick	159
17.2	Unit Testing	161
17.2.1	Unit Testing ohne Spring	162
17.2.2	Unit Testing Utilities im Spring Framework	164
17.3	Integration Testing	166
17.3.1	Spring TestContext Framework	167
17.3.2	SpringExtension und SpringRunner	168
17.3.3	TestContextManager und ContextConfiguration	171

17.4	Wichtige Testfunktionalitäten im Überblick	173
17.4.1	@SpringBootTest	175
17.4.2	Context Initializers	176
17.4.3	@TestPropertySource	177
17.4.4	@ActiveProfiles	178
17.4.5	@TestConfiguration	179
17.4.6	@TestExecutionListeners	181
17.5	Testen von einzelnen Schichten mit Test Slices	182
17.5.1	Was sind Test Slices?	183
17.5.2	@WebMvcTest	184
17.5.3	@DataJpaTest	185
17.5.4	@JsonTest	186
17.5.5	Weitere Test Slices	187
18	Spring Reactive	189
18.1	Warum Reactive?	189
18.2	Grundlagen	192
18.2.1	Reactive Streams und Project Reactor	192
18.2.2	Reaktive Datentypen: Mono und Flux	193
18.2.3	Reactive Chaining	195
18.3	Spring WebFlux	196
18.3.1	Annotierte Controller	198
18.3.2	Funktionale Endpoints	201
18.3.3	WebClient statt RestTemplate	204
18.4	Testen von Reactive Streams	207
18.4.1	StepVerifier	207
18.4.2	WebFluxTest	209
19	Spring Web MVC	211
19.1	REST	211
19.1.1	RestController	211
19.1.2	Request-Daten auslesen	213
19.1.3	Reactive RestController	214
19.2	JSON-Mapping mit Jackson	214
19.2.1	Jackson Annotations	215
19.2.2	Custom Serializer	216
19.3	Error Handling	217

19.4	API-Dokumentation mit Spring REST Docs	219
19.4.1	Setup für REST Docs	220
19.4.2	JUnit-Test zur Generierung	220
19.4.3	Nutzen von Snippets	221
19.5	HATEOAS	221
19.5.1	Wieso überhaupt HATEOAS?	221
19.5.2	Repräsentation von Ressourcen	222
19.5.3	Hypermedia Links & Relations	222
19.5.4	Komfortable Links mit dem WebMvcLinkBuilder	223
19.5.5	Affordances	223
19.5.6	Integration mit RestTemplate und WebClient	224
20	HTTP-Clients mit Spring	225
20.1	RestTemplate	225
20.2	RestClient	225
20.3	WebClient	227
20.4	HTTP-Interface	228
21	GraphQL	231
21.1	Einführung	231
21.2	GraphQL in Kürze	231
21.2.1	Das GraphQL-Schema	232
21.2.2	Die Query Language	236
21.2.3	Ein GraphQL-Schema kann noch mehr	238
21.3	Ablauf eines GraphQL-Requests	238
21.3.1	Transportmethoden	238
21.4	Erstellung einer Spring-Boot-Anwendung mit GraphQL-API	239
21.4.1	Verwenden des Spring Initializr	239
21.4.2	Manuelles Hinzufügen der notwendigen Abhängigkeiten	240
21.4.3	Interaktives Testen mit GraphiQL	240
21.5	Implementierung einer GraphQL-API	241
21.5.1	Definieren eines GraphQL-Schemas	243
21.5.2	GraphQL Controller in Spring Boot	243
21.5.3	Optimierte Abfragen mit Batches	245
21.5.4	Definition eigener Skalare	250
21.6	Fehlerbehandlung	252

22	Spring Boot Developer Tools	255
22.1	Warum Developer Tools?	255
22.2	Developer Tools aktivieren	256
22.3	Restart	256
22.4	Live Reload	257
22.5	Einschränkungen der Developer Tools	258
23	Events	259
23.1	Lose Kopplung	259
23.2	Events versenden	260
23.3	Events empfangen	261
23.3.1	ApplicationListener	262
23.3.2	@EventListener	263
23.4	Synchron oder asynchron?	264
23.5	Spring Boot ApplicationEvents	266
24	Caching	269
24.1	Cache-Configuration	269
24.2	Caching-Annotationen	270
24.3	Cache-Implementierungen	272
24.3.1	EhCache	273
24.3.2	Caffeine	273
25	Messaging	275
25.1	Messaging im Überblick	275
25.2	JMS	277
25.2.1	Mit einer JMS-API verbinden	277
25.2.2	Nachrichten senden	278
25.2.3	Nachrichten empfangen	280
25.2.4	Message Converter	280
25.3	AMQP	281
25.3.1	Das AMQP-Protokoll	281
25.3.2	Konfiguration	281
25.3.3	Nachrichten versenden	282
25.3.4	Nachrichten empfangen	283
25.4	Kafka	283
25.4.1	Konfiguration	283
25.4.2	Nachrichten versenden und empfangen	284

25.5	Ausblick: Spring Cloud Stream	284
25.5.1	Konzepte	285
25.5.2	Nachrichten senden und empfangen	286
25.5.3	Vorteile	287
26	Spring Data	289
26.1	Überblick über Spring Data	289
26.2	Spring Data Repositories	290
26.2.1	Das Repository-Interface	291
26.2.2	Query-Methoden	294
26.3	Die DataSource Bean	296
26.4	JPA	298
26.4.1	Welchen Mehrwert bietet Spring Data JPA?	298
26.4.2	JPA-Repositories und -Entities	299
26.4.3	Datenbankinitialisierung mit JPA und Hibernate	302
26.5	R2DBC	303
26.5.1	Datenbankverbindung über ConnectionFactory	304
26.5.2	Datenbankzugriff	305
26.6	NoSQL mit Spring Data MongoDB	307
26.6.1	MongoDatabaseFactory und MongoTemplate	307
26.6.2	Spring Data Repository für MongoDB	309
26.6.3	Integrationstests mit @DataMongoTest	310
26.7	Schema-Migration mit Flyway	312
27	Spring Cloud Config	315
27.1	Warum Spring Cloud Config?	315
27.2	Der Spring Cloud Config-Server	316
27.3	Environment Repository	317
27.4	Spring Cloud Config-Client	319
28	Spring Security	323
28.1	Konzepte	323
28.1.1	Authentication	323
28.1.2	Authorization	324
28.1.3	Auto-Configuration	325
28.2	Absicherung von Methoden	327
28.3	Absicherung von HTTP-Pfaden	329
28.4	Benutzerverwaltung	331

28.5	Security Testing	332
28.6	OAuth	334
28.6.1	Was ist OAuth?	334
28.6.2	OAuth Resource-Server	335
28.6.3	OAuth-Client	336
29	Observability	339
29.1	Warum Observability?	339
29.2	Admin-Endpoints mit Actuator	339
29.2.1	Anwendungs-Metadaten ausgeben	342
29.2.2	Einen eigenen Actuator-Endpoint entwickeln	347
29.3	Logging	349
29.3.1	SLF4J	349
29.3.2	Logging via application.yml konfigurieren	350
29.3.3	Logback und Log4J direkt konfigurieren	351
29.3.4	Kombination der Logging-Konfigurationen	352
29.4	Metriken	353
29.4.1	Standardmetriken exportieren	353
29.4.2	Metriken an ein Observability-Produkt exportieren	355
29.4.3	Eigene Metriken exportieren	356
29.4.4	Tags	359
29.4.5	Metriken programmatisch anpassen	360
29.4.6	Histogramme und Perzentile	363
29.5	Tracing	366
29.5.1	Traces programmatisch auswerten	368
29.5.2	Traces in Log-Events ausgeben	369
29.5.3	Traces exportieren	370
29.5.4	Clients instrumentieren	372
29.5.5	Baggage	374
30	Docker-Images mit Spring Boot	377
30.1	Warum Docker?	377
30.2	Einfaches Docker-Image mit Spring Boot	377
30.3	Optimierte Docker-Images	379
30.4	Optimiertes Docker-Image mit Spring Boot	381
30.5	Docker abstrahieren mit Buildpacks	383

31	Native Images mit Spring Boot	385
31.1	Warum Native?	385
31.2	Was ist ein Native Image?	386
31.3	Anwendungsfälle für Native Images	387
31.4	Ahead-of-Time-Optimierung mit Spring Boot	388
31.5	Ein natives Image erstellen	389
31.6	Ein natives Image testen	390
31.7	Reachability-Metadaten erstellen	390
32	Spring Boot erweitern	393
32.1	Cross-Cutting Concerns	393
32.2	@Configuration und @Import	394
32.3	@Enable...-Annotationen	394
32.4	@AutoConfiguration	395
32.5	Bedingte @Configuration	396
32.6	Testen von @AutoConfigurations	399
32.7	Starter	400
32.8	Fortgeschrittene Erweiterungspunkte	401
32.8.1	FactoryBean	401
32.8.2	BeanPostProcessor	402
32.8.3	BeanDefinitionRegistryPostProcessor	403
32.8.4	EnvironmentPostProcessor	405
33	Coordinated Restore at Checkpoint (CRaC)	407
33.1	Warum CRaC?	407
33.2	Checkpoint und Restore	408
33.3	Checkpoint und Restore mit Spring Boot	409
33.4	Automatische Checkpoints	410
33.5	Checkpoints in Docker	410
33.6	CRaC vs. GraalVM	411
34	Migration von Spring Boot 2 zu Spring Boot 3	413
34.1	Überblick	413
34.2	Schritt 1: Bibliotheken analysieren und aktualisieren	414
34.3	Schritt 2: Auf Java 17 aktualisieren	415

- 34.4 Schritt 3: Das Spring-Boot-Upgrade vorbereiten 416
 - 34.4.1 WebSecurityConfigurerAdapter durch
WebSecurityFilterChain ersetzen 417
 - 34.4.2 @AutoConfiguration 418
 - 34.4.3 @LocalServerPort 419
 - 34.4.4 @EnableWebFluxSecurity 419
- 34.5 Schritt 4: Spring Boot aktualisieren 420
 - 34.5.1 Spring Boot auf 3.x aktualisieren 420
 - 34.5.2 Bibliotheken aktualisieren 421
 - 34.5.3 javax durch jakarta ersetzen 421
 - 34.5.4 Spring Cloud Sleuth durch Micrometer ersetzen 422
 - 34.5.5 @ConstructorBinding 423
 - 34.5.6 HttpStatusCode 424
 - 34.5.7 Konfigurationsparameter 424
- 35 Ausblick 425**
- Index 427**