

Auf einen Blick

1	Einleitung	33
2	Container für Spring-managed Beans	67
3	Ausgewählte Module des Spring Frameworks	207
4	Ausgewählte Proxies	365
5	Relationale Datenbanken anbinden	427
6	Jakarta Persistence mit Spring	511
7	Spring Data JPA	635
8	Spring Data für NoSQL-Datenbanken	749
9	Spring Web	777
10	Spring AI	983
11	Logging und Monitoring	995
12	Build und Deployment	1021

Inhalt

Vorwort	23
---------------	----

1	Einleitung	33
----------	-------------------	-----------

1.1	Einleitung und ein erstes Spring-Projekt	33
1.1.1	Spring Boot	38
1.1.2	Ein Spring-Boot-Projekt aufsetzen	43
1.1.3	Spring-Projekte in Entwicklungsumgebungen aufbauen	48
1.2	Dependencies und Starter eines Spring-Boot-Projekts	54
1.2.1	POM mit Parent-POM	54
1.2.2	Dependencies als Import	57
1.2.3	Milestones und das Snapshots-Repository	58
1.2.4	Annotationsprozessoren konfigurieren	59
1.2.5	Starter: Dependencies des Spring Initializrs	60
1.3	Einstieg in die Konfigurationen und das Logging	62
1.3.1	Banner abschalten	62
1.3.2	Die Logging-API und SLFJ	63

2	Container für Spring-managed Beans	67
----------	---	-----------

2.1	Spring-Container	67
2.1.1	Container starten	68
2.1.2	SpringApplication instanziiieren	69
2.1.3	SpringApplicationBuilder *	70
2.1.4	Was main(...) macht und nicht macht	71
2.1.5	Die run(...) -Methode liefert ConfigurableApplicationContext	71
2.1.6	Kontextmethoden	72
2.2	Design und Strukturierung einer Anwendung	75
2.2.1	Domain-driven Design und hexagonale Architektur	75
2.2.2	Hexagonale Architektur mit Domain-driven Design	77
2.2.3	Paketstruktur der Date4u-Anwendung	78
2.3	Spring-managed Beans durch Classpath-Scanning aufnehmen	81
2.3.1	Container mit Beans füllen	81

2.3.2	@Component	82
2.3.3	@Repository, @Service, @Controller	85
2.3.4	Classpath-Scanning mit @ComponentScan präziser steuern *	87
2.4	Interaktive Anwendungen mit der Spring Shell	95
2.4.1	Eine Spring-Shell-Dependency einbinden	95
2.4.2	Eine erste interaktive Anwendung	97
2.4.3	Shell-Komponente schreiben: @ShellComponent, @ShellMethod ...	97
2.5	Die Verweisspritze	101
2.5.1	Objektgraphen aufbauen	101
2.5.2	Inversion of Control (IoC) und Dependency-Injection	103
2.5.3	Injektionsarten	103
2.5.4	PhotoService und PhotoCommands	109
2.5.5	Mehrere Abhängigkeiten	112
2.5.6	Verhalten bei einer fehlenden Komponente	113
2.5.7	Optionale Abhängigkeit	114
2.5.8	Zyklische Abhängigkeiten *	117
2.5.9	Andere Dinge injizieren	118
2.6	Konfigurationsklassen und Fabrikmethoden	119
2.6.1	@Configuration und @Bean	120
2.6.2	Parameter-Injektion einer @Bean-Methode	128
2.6.3	@Configuration-Bean und Lite-Bean	129
2.6.4	InjectionPoint *	131
2.6.5	Statische @Bean-Methoden *	134
2.6.6	@Import und @ImportSelector *	135
2.7	Abstraktion und Qualifizierungen	137
2.7.1	Der Name einer Bean und ihr Alias	137
2.7.2	AwtBicubicThumbnail für Vorschaubilder	139
2.7.3	Basistypen	142
2.7.4	ObjectProvider	151
2.7.5	@Order und @AutoConfigurationOrder *	154
2.7.6	Verhalten bei ausgewählten Vererbungsbeziehungen *	156
2.8	Lebenszyklus der Beans	158
2.8.1	@DependsOn	159
2.8.2	Verzögerte Initialisierung (lazy initialization)	160
2.8.3	Bean-Initialisierung traditionell	162
2.8.4	InitializingBean und DisposableBean *	168
2.8.5	Vererbung der Lebenszyklus-Methoden	169
2.8.6	*Aware-Schnittstellen *	169
2.8.7	BeanPostProcessor *	173
2.8.8	Spring-managed Beans woanders anmelden	177

2.8.9	Hierarchische Kontexte *	179
2.8.10	Singleton und Prototyp zustandslos oder zustandsbehaftet	182
2.9	Annotationen aus JSR 330, »Dependency Injection for Java« *	183
2.10	Autokonfiguration	185
2.10.1	@Conditional und Condition	185
2.10.2	Wenn, dann: @ConditionalOn*	187
2.10.3	Die Debug-Protokollierung von Spring einschalten	195
2.10.4	Autokonfigurationen individuell steuern *	197
2.11	Spring Expression Language (SpEL)	200
2.11.1	ExpressionParser	200
2.11.2	SpEL in der Spring (Boot) API	202
3	Ausgewählte Module des Spring Frameworks	207
3.1	Hilfsklassen im Spring Framework	207
3.1.1	Bestandteile von org.springframework	207
3.2	Externe Konfiguration und das Environment	208
3.2.1	Konfiguration im Code und extern	209
3.2.2	Environment	210
3.2.3	Werte mit @Value injizieren	212
3.2.4	Environment-Belegungen über @Value und \${...} bekommen	213
3.2.5	@Value und Default-Werte	214
3.2.6	Zugriff auf Konfigurationen	217
3.2.7	Geschachtelte/Hierarchische Properties	223
3.2.8	Besondere Datentypen abbilden	228
3.2.9	Relaxed Bindings	234
3.2.10	Property-Sources	236
3.2.11	Spring-Profile definieren	251
3.2.12	Profile aktivieren	254
3.2.13	Spring-managed Beans je nach Profil	256
3.3	Am Anfang und Ende	258
3.3.1	CommandLineRunner und ApplicationRunner	258
3.3.2	Am Ende der Anwendung	261
3.3.3	Java-Programme mit Exit-Code beenden *	262
3.4	Ereignisbehandlung	268
3.4.1	Beteiligte Objekte	268
3.4.2	Context Events	268
3.4.3	ApplicationListener	270

3.4.4	@EventListener	271
3.4.5	Methoden der Ereignisklassen	272
3.4.6	Ereignisklassen schreiben und auf die Ereignisse reagieren	273
3.4.7	Ein Eventbus vom Typ ApplicationEventPublisher	273
3.4.8	Generische Ereignisse am Beispiel von PayloadApplicationEvent	275
3.4.9	Ereignis-Transformationen	277
3.4.10	Reihenfolgen durch @Order	278
3.4.11	Ereignisse nach Bedingungen filtern	278
3.4.12	Ereignisse synchron und asynchron	279
3.4.13	ApplicationEventMulticaster *	280
3.4.14	Ereignisse über ApplicationContext versenden	281
3.4.15	Listener an SpringApplication hängen *	281
3.4.16	Listener in spring.factories *	282
3.5	Ressourcen-Abstraktion mit Resource	283
3.5.1	InputStreamSource und Resource	284
3.5.2	Ressourcen laden	285
3.5.3	Ressourcen über @Value injizieren	286
3.6	Typkonvertierung mit ConversionService	288
3.6.1	ConversionService	288
3.6.2	DefaultConversionService und ApplicationConversionService	289
3.6.3	ConversionService als Spring-managed Bean	291
3.6.4	Bei der ConverterRegistry eigene Konverter anmelden	292
3.6.5	Printer und Parser	297
3.6.6	FormatterRegistry	299
3.6.7	DataBinder	302
3.7	Internationalisierung *	306
3.7.1	Möglichkeiten zur Internationalisierung mit der Java SE	306
3.7.2	MessageSource unter Untertypen	308
3.8	Testgetriebene Entwicklung mit Spring Boot	312
3.8.1	Testbezogene Einträge vom Spring Initializr	313
3.8.2	Annotation @Test	314
3.8.3	Testfall für die Klasse »FileSystem«	314
3.8.4	Mehrschichtige Anwendungen testen, Objekte austauschen	318
3.8.5	Das Mocking-Framework Mockito	322
3.8.6	@InjectMocks	326
3.8.7	Verhalten verifizieren	326
3.8.8	Testen mit ReflectionTestUtils	327
3.8.9	Mocken von Spring-Datentypen	330
3.8.10	Mit oder ohne Spring-Unterstützung	330
3.8.11	Test-Properties	333

3.8.12	@TestPropertySource	334
3.8.13	@ActiveProfiles	336
3.8.14	Stellvertreter einsetzen	337
3.8.15	@DirtiesContext	340
3.9	Scheibchen testen am Beispiel von JSON *	340
3.9.1	JSON	341
3.9.2	Jackson	342
3.9.3	Ein Java-Objekt in JSON schreiben	342
3.9.4	Testen von JSON-Mappings: @JsonTest	345
3.9.5	Die Abbildung mit @JsonComponent	349
3.10	Scheduling *	352
3.10.1	Scheduling-Annotationen und @EnableScheduling	352
3.10.2	@Scheduled	353
3.10.3	Nachteile von @Scheduled und Alternativen	356
3.11	Typen aus org.springframework.*.[lang util]	356
3.11.1	org.springframework.lang.*Null*-Annotationen	357
3.11.2	Das Paket org.springframework.util	358
3.11.3	Das Paket org.springframework.data.util	359

4 **Ausgewählte Proxies** 365

4.1	Proxy-Pattern	366
4.1.1	Proxy-Einsatz in Spring	367
4.1.2	Proxies dynamisch generieren	369
4.2	Caching	372
4.2.1	Optimierung durch Caching	372
4.2.2	Caching in Spring	373
4.2.3	Komponente mit @Cacheable implementieren	374
4.2.4	@Cacheable-Proxy nutzen	375
4.2.5	@Cacheable + condition	378
4.2.6	@Cacheable + unless	379
4.2.7	@CachePut	380
4.2.8	@CacheEvict	380
4.2.9	Eigene Schlüsselgeneratoren angeben	381
4.2.10	@CacheConfig	388
4.2.11	Cache-Implementierungen	389
4.2.12	Caching mit Caffeine	389
4.2.13	Das Caching im Test abschalten	391

4.3	Asynchrone Aufrufe	392
4.3.1	@EnableAsync und @Async-Methoden	392
4.3.2	Beispiel mit @Async	393
4.3.3	Der Rückgabetyp CompletableFuture	396
4.4	TaskExecutor *	400
4.4.1	TaskExecutor-Implementierungen	401
4.4.2	Executor setzen und Ausnahmen behandeln	403
4.5	Spring und Bean Validation	406
4.5.1	Parameterprüfungen	406
4.5.2	Die Jakarta Bean Validation (JSR 303)	407
4.5.3	Dependency auf die Spring-Boot-Starter-Validation	407
4.5.4	»Photo« mit Jakarta-Bean-Validation-Annotationen	408
4.5.5	Einen Validator injizieren und prüfen	409
4.5.6	Spring und die Bean-Validation-Annotationen	412
4.5.7	Bean-Validation-Annotationen an Methoden	414
4.5.8	Validierung überall, am Beispiel einer Konfiguration	414
4.5.9	Refactoring der Photo-Klasse	416
4.6	Spring Retry *	417
4.6.1	Das Spring-Retry-Projekt	417
4.6.2	@Retryable	419
4.6.3	Fallback mit @Recover	423
4.6.4	RetryTemplate	424
5	Relationale Datenbanken anbinden	427
5.1	Eine H2-Datenbank aufsetzen	427
5.1.1	Kurzvorstellung der H2-Datenbank	427
5.1.2	H2 installieren und starten	428
5.1.3	Über die H2 Console eine Verbindung zur Datenbank aufbauen	431
5.1.4	Das Datenbankschema der Date4u-Datenbank	433
5.2	Datenbankzugriffe mit Spring realisieren	436
5.2.1	Spring-Helfer	436
5.3	Der Spring Boot Starter JDBC	438
5.3.1	Den JDBC-Starter in die POM aufnehmen	438
5.3.2	JDBC-Verbindungsdaten eintragen	439
5.3.3	DataSource oder JdbcTemplate injizieren	439
5.3.4	Connection-Pooling	442
5.3.5	Das Paket org.springframework.jdbc und seine Unterpakete	443

5.3.6	DataSourceException	444
5.3.7	Autokonfiguration für DataSource *	446
5.3.8	Mehrere Datenbanken ansprechen	447
5.3.9	DataSourceUtils *	450
5.4	JdbcTemplate	451
5.4.1	Beliebiges SQL ausführen: execute(...)	451
5.4.2	SQL-Aktualisierungen: update(...)	452
5.4.3	Einzelne Werte erfragen: queryForObject(...)	452
5.4.4	Einen Platzhalter für ein PreparedStatement definieren	453
5.4.5	Ganze Zeile erfragen: queryForMap(...)	454
5.4.6	Mehrere Zeilen mit einem Element erfragen: queryForList(...)	455
5.4.7	Mehrere Zeilen und Spalten einlesen: queryForList(...)	456
5.5	Datentypen zum Mapping auf Ergebnisse	457
5.5.1	RowMapper	458
5.5.2	RowCallbackHandler	463
5.5.3	ResultSetExtractor	464
5.6	NamedParameterJdbcTemplate	468
5.6.1	Typbeziehungen von *JdbcTemplate	468
5.6.2	Methoden vom NamedParameterJdbcTemplate	469
5.6.3	Werte von NamedParameterJdbcTemplate mit Map übergeben	470
5.6.4	SqlParameterSource	470
5.6.5	Rückgriff auf tiefer liegende Objekte *	473
5.7	Batch-Operationen *	474
5.7.1	Batch-Methoden beim JdbcTemplate	474
5.7.2	BatchPreparedStatementSetter	475
5.7.3	Batch-Methoden beim NamedParameterJdbcTemplate	480
5.7.4	SqlParameterSourceUtils	481
5.7.5	Konfigurations-Properties	482
5.8	Es wird groß mit BLOB und CLOB *	483
5.8.1	SqlLobValue	483
5.8.2	LobHandler und DefaultLobHandler	484
5.8.3	LOBs über AbstractLobStreamingResultSetExtractor lesen	486
5.9	Das Paket org.springframework.jdbc.core.simple *	487
5.9.1	JdbcClient	487
5.9.2	SimpleJdbcInsert	492
5.10	Das Paket org.springframework.jdbc.object *	494
5.10.1	MappingSqlQuery	495

5.11	Transaktionen	497
5.11.1	Das ACID-Prinzip	497
5.11.2	Lokal oder global/verteilt	497
5.11.3	JDBC-Transaktionen/Auto-Commit	498
5.11.4	PlatformTransactionManager	499
5.11.5	TransactionTemplate	503
5.11.6	@Transactional	507

6	Jakarta Persistence mit Spring	511
----------	---------------------------------------	-----

6.1	Welt der Objekte und der Datenbanken	511
6.1.1	Transient und persistent	511
6.1.2	Abbildung von Objekten auf Tabellen	512
6.1.3	Java-Bibliotheken für das O/R-Mapping	516
6.2	Jakarta Persistence	517
6.2.1	Persistence Provider	518
6.2.2	Jakarta Persistence Provider nutzen JDBC	519
6.2.3	Was deckt die Jakarta Persistence ab?	519
6.3	Spring Data JPA	520
6.3.1	Spring Boot Starter Data JPA einbinden	520
6.3.2	Konfigurationen	521
6.4	Die Jakarta-Persistence-Entity-Bean	523
6.4.1	Eine Entity-Bean-Klasse entwickeln	524
6.5	Die Jakarta Persistence API	531
6.5.1	Den EntityManager von Spring bekommen	531
6.5.2	Eine Entity-Bean über ihren Schlüssel suchen: find(...)	533
6.5.3	find(...) und getReference(...)	536
6.5.4	Abfragemöglichkeiten mit dem EntityManager	537
6.6	Die Jakarta Persistence Query Language (JPQL)	537
6.6.1	Ein JPQL-Beispiel mit SELECT und FROM	538
6.6.2	JPQL-Abfragen mit createQuery(...) aufbauen und absenden	540
6.6.3	Bedingungen in WHERE	543
6.6.4	JPQL-Aufrufe parametrisieren	543
6.6.5	JPQL-Operatoren und -Funktionen	546
6.6.6	Rückgaben ordnen mit ORDER BY	551
6.6.7	Projektion auf skalare Werte	552
6.6.8	Aggregatfunktionen	553

6.6.9	Projektion auf mehrere Werte	554
6.6.10	Benannte deklarative Abfragen (Named Queries)	558
6.7	Datenbankfunktionen aufrufen und native SQL-Anfragen senden	560
6.7.1	Datenbankfunktionen aufrufen: FUNCTION(...)	561
6.7.2	createNativeQuery(...) über EntityManager nutzen	561
6.7.3	@NamedNativeQuery	563
6.7.4	@NamedNativeQuery mit resultSetMapping	564
6.8	Schreibender Zugriff mit dem EntityManager in Transaktionen	567
6.8.1	Transaktionale Operationen	567
6.8.2	persist(...)	568
6.8.3	EntityTransaction	568
6.8.4	PlatformTransactionManager → JpaTransactionManager	569
6.8.5	@Transactional	571
6.8.6	Speichern vs. Aktualisieren	572
6.8.7	remove(...)	572
6.8.8	Synchronisation bzw. Flush	573
6.8.9	Query mit UPDATE und DELETE	573
6.9	Persistence Context und weitere Transaktionssteuerungen	574
6.9.1	Zustände einer Entity-Bean	575
6.10	Weiterführende OR-Metadaten	579
6.10.1	Der Database-first- und der Code-first-Ansatz	579
6.10.2	Den Tabellennamen über @Table setzen	579
6.10.3	Den @Entity-Namen ändern	580
6.10.4	Persistente Attribute	581
6.10.5	@Basic und @Transient	582
6.10.6	Spaltenbeschreibung und @Column	583
6.10.7	Entity-Bean-Datentypen	586
6.10.8	Datentypen mit AttributeConverter mappen	588
6.10.9	Schlüsselkennzeichnung	591
6.10.10	Eingebettete Typen	595
6.10.11	Eine Entity-Bean erbt Eigenschaften von einer Oberklasse	598
6.11	Beziehungen zwischen Entitäten	600
6.11.1	Unterstützte Assoziationen und Beziehungstypen	600
6.11.2	Die 1:1-Beziehung	600
6.11.3	Bidirektionale Beziehungen	605
6.11.4	Die 1:n-Beziehung	606
6.11.5	n:m-Beziehungen *	612
6.12	FetchType: Lazy und Eager Loading	613
6.12.1	Aufzählung mit FetchType	614

6.12.2	Der Hibernate-Typ PersistentBag	614
6.12.3	Das 1 + N Query-Problem ist ein Performance-Antipattern	616
6.13	Cascading	619
6.13.1	Der Aufzählungstyp CascadeType	620
6.13.2	CascadeType setzen	622
6.13.3	cascade=REMOVE vs. orphanRemoval=true	623
6.14	Repositories	624
6.14.1	Datenzugriffsschicht	625
6.14.2	Methoden im Repository	627
6.14.3	SimpleJpaRepository	629
7	Spring Data JPA	635
7.1	Welche Aufgaben erfüllt Spring Data?	635
7.2	Spring Data Commons: CrudRepository	637
7.2.1	Der Typ CrudRepository	638
7.2.2	JPA-basierte Repositories	639
7.3	Untertypen von CrudRepository: JpaRepository etc.	642
7.3.1	ListCrudRepository	643
7.3.2	Technologiespezifische [List]CrudRepository-Untertypen	643
7.3.3	Ausgewählte Methoden über Repository	646
7.4	Blättern und Sortieren mit [List]PagingAndSortingRepository	647
7.4.1	Der Typ Sort	649
7.4.2	Sort.TypedSort<T>	650
7.4.3	Pageable und PageRequest	651
7.4.4	Sortieren von paginierten Seiten	657
7.4.5	Performanceüberlegungen	657
7.5	QueryByExampleExecutor *	658
7.5.1	Die Probe	659
7.5.2	QueryByExampleExecutor	659
7.5.3	Probe in das Example	660
7.5.4	ExampleMatcher aufbauen	661
7.5.5	Properties ignorieren	662
7.5.6	String-Vergleichs-Techniken setzen	663
7.5.7	Bewertung von Query-By-Example (QBE)	663
7.6	Eigene Abfragen mit @Query formulieren	664
7.6.1	Die @Query-Annotation	664
7.6.2	Verändernde @Query-Operationen mit @Modifying	666

7.6.3	IN-Parameter durch Array/Vararg/Collection füllen	667
7.6.4	@Query mit JPQL-Projektion	667
7.6.5	Sort- und Pageable-Parameter	668
7.6.6	Neue Query-Methoden ergänzen	670
7.6.7	Queries mit SpEL-Ausdrücken	671
7.6.8	Die @NamedQuery einer Entity-Bean verwenden	672
7.6.9	Die @Query-Annotation mit nativem SQL	672
7.7	Stored Procedures (gespeicherte Prozeduren) *	675
7.7.1	Eine gespeicherte Prozedur in H2 definieren	676
7.7.2	Eine gespeicherte Prozedur über eine native Query aufrufen	677
7.7.3	Eine gespeicherte Prozedur mit @Procedure aufrufen	678
7.8	Derived Query Methods	679
7.8.1	Individuelle CRUD-Operationen über Methodennamen	679
7.8.2	Aufbau der Derived Query Methods	680
7.8.3	Rückgaben von Derived Query Methods	684
7.8.4	Asynchrone Query-Methoden	685
7.8.5	Streamende Query-Methoden	685
7.8.6	Vorteile und Nachteile der Derived Query Methods	685
7.9	Die Criteria API und der JpaSpecificationExecutor	686
7.9.1	Die Criteria API	687
7.9.2	Die funktionale Schnittstelle Specification	688
7.9.3	JpaSpecificationExecutor	689
7.9.4	Methoden in JpaSpecificationExecutor	689
7.9.5	Specification-Implementierungen	690
7.9.6	Specification-Instanzen zusammensetzen	692
7.9.7	Interna *	693
7.9.8	Metamodellklassen	694
7.9.9	Nachteile der Criteria API	697
7.10	Alternativen zu JDBC Jakarta Persistence	698
7.10.1	Querydsl	699
7.10.2	Spring Data JDBC	706
7.11	Gutes Design mit Repositories	712
7.11.1	Abstraktionen durch die Zwiebelarchitektur	712
7.11.2	Denke an ISP und die Zwiebel!	713
7.11.3	Das Fragment-Interface	715
7.12	Projektionen	717
7.12.1	Projektionen in Spring Data	718
7.12.2	Die Interface-basierte Projektion	718
7.12.3	Projektionen mit SpEL-Ausdrücken	720

7.12.4	Projektionen mit Default-Methoden	721
7.12.5	Klassenbasierte Projektionen	721
7.12.6	Dynamische Projektionen	722
7.13	[Fetchable]FluentQuery *	723
7.14	Auditing *	725
7.14.1	Auditing mit Spring Data	725
7.14.2	Auditing mit Spring Data JPA	726
7.14.3	AuditorAware für User-Informationen	727
7.14.4	Ausblick: Spring Data Envers	728
7.15	Incremental Data Migration	729
7.15.1	Lang leben die Daten	730
7.15.2	Evolutionäres Datenbankdesign	730
7.15.3	Incremental Data Migration mit Flyway	731
7.15.4	Flyway in Spring Boot: Migrationsskripte	732
7.15.5	Migrations in Java-Code	734
7.15.6	Flyway-Migrations außerhalb von Spring	736
7.16	Die Datenzugriffsschicht testen	736
7.16.1	Was wollen wir testen?	737
7.16.2	Testscheibchen	737
7.16.3	Eine In-Memory-Testdatenbank einsetzen	739
7.16.4	Eigene Verbindungsdaten zur Testdatenbank zuweisen	740
7.16.5	Tabellen aufbauen mit Initialisierungsskripten	741
7.16.6	Das Projekt Testcontainers	742
7.16.7	Demodaten	745
7.16.8	@Sql und @SqlGroup	745
7.16.9	TestEntityManager	746

8

Spring Data für NoSQL-Datenbanken

749

8.1	Not only SQL	749
8.2	MongoDB	750
8.2.1	MongoDB: Dokumente und Collections	750
8.2.2	Über MongoDB	751
8.2.3	Den MongoDB Community Server installieren und starten	752
8.2.4	GUI-Werkzeuge für MongoDB	753
8.2.5	Spring Data MongoDB	754
8.2.6	MongoDB APIs	756
8.2.7	MongoDB-Dokumente	756

8.2.8	Die Klasse MongoTemplate	758
8.2.9	MongoDB-Repositories	760
8.2.10	MongoDB-Programme testen	763
8.3	Elasticsearch	764
8.3.1	Textsuche ist anders	764
8.3.2	Apache Lucene	765
8.3.3	Dokument und Feld	765
8.3.4	Index	765
8.3.5	Die Schwächen von Apache Lucene	767
8.3.6	Lucene-Aufsätze: Elasticsearch und Apache Solr	767
8.3.7	Elasticsearch installieren und starten	768
8.3.8	Spring Data Elasticsearch	769
8.3.9	Dokumente	770
8.3.10	ElasticsearchRepository	773
8.3.11	@DataElasticsearchTest	776
9	Spring Web	777
9.1	Webserver	777
9.1.1	Java-Webserver	778
9.1.2	Spring Boot Starter Web	779
9.1.3	Andere Webserver einsetzen *	780
9.1.4	Port anpassen über server.port	781
9.1.5	Statische Ressourcen servieren	781
9.1.6	WebJars	782
9.1.7	TLS-Verschlüsselung	783
9.2	Dynamische Inhalte generieren	784
9.2.1	Steinzeit: Das Common Gateway Interface (CGI)	784
9.2.2	Der Servlet-Standard	786
9.2.3	@WebServlet programmieren	787
9.2.4	Schwächen von Servlets	788
9.3	Spring Web MVC	789
9.3.1	Spring-Container in Webanwendungen	789
9.3.2	@Controller und @ResponseBody	791
9.3.3	@RestController	792
9.3.4	Controller → [Service →] Repository	795
9.4	Hot Code Swapping	796
9.4.1	Hot Swapping der JVM	796
9.4.2	Spring Developer Tools	797

9.5	Das Hypertext Transfer Protocol (HTTP)	797
9.5.1	HTTP-Request und -Response	798
9.5.2	Einen HTTP-Client zum Testen von Anwendungen aufsetzen	799
9.6	Request Matching	800
9.6.1	@RequestMapping	800
9.6.2	@*Mapping	801
9.6.3	Allgemeinere Pfadausdrücke und Pfad-Matcher	801
9.6.4	@RequestMapping am Typ	802
9.7	Response senden	803
9.7.1	HttpMessageConverter in der Anwendung	803
9.7.2	Formatkonvertierungen der Rückgaben von Handler-Methoden	804
9.7.3	Abbildung auf JSON-Dokumente	805
9.7.4	HTTP-Statuscode und @ResponseStatus	811
9.7.5	ResponseEntity = Statuscode + Header + Body	814
9.8	Request auswerten	815
9.8.1	Handler-Methoden mit Parametern	815
9.8.2	Datenübertragung vom Client zum Controller	816
9.8.3	Datenannahme über Parameter	817
9.8.4	Query-Parameter auswerten	817
9.8.5	Optionale Query-Parameter	819
9.8.6	Alle Query-Parameter mappen	820
9.8.7	Eine Pfadvariable auswerten	821
9.8.8	REST-Endpunkte für Profile implementieren	825
9.8.9	Data Transfer Objects (DTO)	828
9.8.10	POST und PUT mit @RequestBody	832
9.8.11	UriComponents	836
9.8.12	MultipartFile	837
9.8.13	Header auswerten	839
9.8.14	HttpEntity mit ihren Unterklassen RequestEntity und ResponseEntity *	840
9.9	Typkonvertierung der Parameter	841
9.9.1	Beispiel für YearMonth-Converter	841
9.9.2	@DateTimeFormat und @NumberFormat	842
9.9.3	Query-Parameter und Formulardaten auf eine Bean mappen	843
9.9.4	Eigene Typkonverter anmelden	846
9.9.5	URI-Template-Pattern mit regulären Ausdrücken	849
9.10	Ausnahmebehandlung und Fehlermeldung	850
9.10.1	Ausnahmen selbst auf Statuscodes mappen	850
9.10.2	Eskaliert	850
9.10.3	Die Konfigurations-Properties server.error.*	852

9.10.4	Exception-Resolver	855
9.10.5	Exception auf Statuscode mit @ResponseStatus mappen	855
9.10.6	ResponseStatusException	857
9.10.7	Lokales Controller-Exception-Handling mit @ExceptionHandler	858
9.10.8	RFC 7807: »Problem Details for HTTP APIs«	861
9.10.9	Globales Controller-Exception-Handling mit Controller-Advice	863
9.11	Webservices und RESTful API	865
9.11.1	Die Prinzipien hinter REST	865
9.12	GraphQL	868
9.12.1	Das Problem mit traditionellen REST-APIs	869
9.12.2	GraphQL-Schema und Datentypen	870
9.12.3	Spring GraphQL	872
9.12.4	Erstellen eines einfachen GraphQL-Schemas	872
9.12.5	Implementierung eines Data-Fetchers	873
9.12.6	GraphQL-Abfrage	873
9.12.7	Testumgebung GraphQL	874
9.12.8	GraphQL-Ergebnis	875
9.12.9	Data Fetcher über Handler-Methoden	876
9.12.10	Ein komplexeres GraphQL-Schema umsetzen	877
9.12.11	Assoziationen und Schema-Mappings	887
9.12.12	CORS (Cross-Origin Resource Sharing)	891
9.12.13	GraphQL JavaScript-Client	891
9.12.14	Spring GraphQL Client	892
9.12.15	Ausblick und Fazit	895
9.13	Asynchrone Web-Requests *	897
9.13.1	Lange Abfragen blockieren den Worker-Thread	897
9.13.2	Asynchron in die Ausgabe schreiben	898
9.13.3	Eine Handler-Methode liefert Callable	898
9.13.4	WebAsyncTask	899
9.13.5	Eine Handler-Methode liefert DeferredResult	899
9.13.6	StreamingResponseBody	900
9.14	Spring Data Web Support	901
9.14.1	Laden aus dem Repository	901
9.14.2	Pageable und Sort als Parametertypen	902
9.14.3	Die Rückgabe Page und PagedModel	905
9.14.4	Payload-Verarbeitung mit Projektionen	906
9.14.5	Querydsl Predicate als Parametertyp	906
9.15	Dokumentation einer RESTful API mit OpenAPI	908
9.15.1	Beschreibung einer RESTful API	909
9.15.2	Die OpenAPI-Spezifikation	909

9.15.3	Woher kommt das OpenAPI-Dokument?	910
9.15.4	OpenAPI mit Spring	911
9.15.5	springdoc-openapi	911
9.15.6	Bessere Dokumentation mit OpenAPI-Annotationen	913
9.15.7	Java-Code aus einem OpenAPI-Dokument generieren	914
9.15.8	Spring REST Docs	918
9.16	Testen der Webschicht	919
9.16.1	QuoteRestController testen	919
9.16.2	Die Annotation @WebMvcTest	920
9.16.3	Eine Testmethode mit MockMvc schreiben	921
9.16.4	REST-Endpunkte mit dem Server testen	926
9.16.5	WebTestClient	926
9.17	Best Practices bei der Nutzung einer RESTful API	928
9.17.1	Jakarta Bean Validation	928
9.17.2	Ressourcen-ID	929
9.17.3	Data Transfer Objects (DTO) mit MapStruct mappen	930
9.17.4	Versionierung von RESTful Webservices	932
9.18	Webanwendungen mit Spring Security absichern	934
9.18.1	Die Bedeutung von Spring Security	934
9.18.2	Dependency auf Spring Boot Starter Security	934
9.18.3	Authentication	935
9.18.4	SecurityContext und SecurityContextHolder	936
9.18.5	AuthenticationManager	938
9.18.6	SpringBootWebSecurityConfiguration	940
9.18.7	AuthenticationManager und ProviderManager	942
9.18.8	Die Schnittstelle UserDetailsService	943
9.18.9	PasswordEncoder	949
9.18.10	BasicAuthenticationFilter	953
9.18.11	Zugriff auf den Benutzer	953
9.18.12	Authorization und das Cookie	955
9.18.13	Token-basierte Authentifizierung mit JWT	955
9.19	RESTful Webservices konsumieren	964
9.19.1	Klassen zum Ansprechen von HTTP-Endpunkten	964
9.19.2	RestClient API	965
9.19.3	WebClient API	971
9.19.4	Deklarative Webservice-Clients	977

10 Spring AI 983

10.1	Was ist künstliche Intelligenz?	983
10.2	Ollama	985
10.2.1	Ollama installieren	985
10.2.2	Modelle installieren	985
10.2.3	Sprachmodelle nutzen	986
10.2.4	KI-Modelle ansprechen und integrieren	986
10.2.5	KI-Modelle in Java-Programmen integrieren	987
10.3	Spring AI nutzen	987
10.3.1	Spring AI Ollama Spring Boot Starter	988
10.3.2	ChatClient.Builder	989
10.3.3	ChatClient ohne Autokonfiguration	989
10.4	Prompt aufbauen	990
10.4.1	Ergebnis abrufen	990
10.4.2	ChatResponse	991
10.4.3	Generation	992
10.4.4	Prompt-Template	992
10.5	Ausblick	993

11 Logging und Monitoring 995

11.1	Logging	995
11.1.1	Warum ein Protokoll erstellen?	995
11.1.2	Log-Group	996
11.2	Logging-Implementierung	996
11.2.1	Das Logging-Pattern-Layout	998
11.2.2	Die Logging-Konfiguration ändern	999
11.2.3	JSON-Logging in Spring Boot	999
11.2.4	MDC (Mapped Diagnostic Context)	1001
11.2.5	Banner	1002
11.2.6	Logging zur Startzeit	1003
11.2.7	Testen von geschriebener Log-Meldung	1003
11.3	Anwendungen mit dem Spring Boot Actuator überwachen	1004
11.3.1	Den Gesundheitszustand über den Actuator ermitteln	1004
11.3.2	Aktivierung der Endpunkte	1006
11.3.3	Info-Ergänzungen	1008

11.3.4	Parameter und JSON-Rückgaben	1008
11.3.5	Neue Actuator-Endpunkte	1010
11.3.6	Bin ich gesund?	1010
11.3.7	HealthIndicator	1012
11.3.8	Metriken	1012
11.4	Micrometer und Prometheus	1013
11.4.1	Micrometer	1014
11.4.2	Prometheus: Die Software	1017
12	Build und Deployment	1021
12.1	Spring-Boot-Programme verpacken und ausführen	1021
12.1.1	Deployment-Optionen	1021
12.1.2	Ein Spring-Boot-Programm über Maven starten	1022
12.1.3	Ein Spring-Boot-Programm in JAR packen	1022
12.1.4	Das spring-boot-maven-plugin	1023
12.2	Spring-Anwendungen im OCI-Container	1024
12.2.1	Container	1024
12.2.2	Docker installieren und nutzen	1026
12.2.3	H2 starten, stoppen, Port-Weiterleitung und Data-Volumes	1028
12.2.4	Eine Spring-Boot-Docker-Anwendung vorbereiten	1031
12.2.5	Docker Compose	1033
12.2.6	Anwendungen beenden mit einem Actuator-Endpunkt	1035
Index		1037