

**Spring Boot 3
und Spring Framework 6**
Das umfassende Handbuch

**DAS
INHALTS-
VERZEICHNIS**

» Hier geht's
direkt
zum Buch

Auf einen Blick

1	Einleitung	35
2	Container für Spring-managed Beans	69
3	Ausgewählte Module des Spring Frameworks	213
4	Ausgewählte Proxys	353
5	Relationale Datenbanken anbinden	417
6	Jakarta Persistence mit Spring	497
7	Spring Data JPA	619
8	Spring Data für NoSQL-Datenbanken	731
9	Spring Web	759
10	Logging und Monitoring	921
11	Build und Deployment	945

Inhalt

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

1 Einleitung 35

1.1 Einleitung und ein erstes Spring-Projekt	35
1.1.1 Spring Boot	40
1.1.2 Ein Spring-Boot-Projekt aufsetzen	46
1.1.3 Spring-Projekte in Entwicklungsumgebungen aufbauen	51
1.2 Dependencys und Starter eines Spring-Boot-Projekts	57
1.2.1 POM mit Parent-POM	57
1.2.2 Dependencys als Import	60
1.2.3 Milestones und das Snapshots-Repository	61
1.2.4 Starter: Dependencys des Spring Initializr	62
1.3 Einstieg in die Konfigurationen und das Logging	64
1.3.1 Banner abschalten	65
1.3.2 Die Logging-API und SL4J	65

2 Container für Spring-managed Beans 69

2.1 Spring-Container	69
2.1.1 Container starten	69
2.1.2 SpringApplication instanziiieren	70
2.1.3 SpringApplicationBuilder *	71
2.1.4 Was main(...) macht und nicht macht	72
2.1.5 Die run(...) -Methode liefert ConfigurableApplicationContext	73
2.1.6 Kontextmethoden	74
2.2 Spring-managed Beans durch Classpath-Scanning aufnehmen	78
2.2.1 Container mit Beans füllen	78
2.2.2 @Component	79
2.2.3 @Repository, @Service, @Controller	82
2.2.4 Classpath-Scanning mit @ComponentScan präziser steuern *	84

2.3	Am Anfang und Ende	92
2.3.1	CommandLineRunner und ApplicationRunner	92
2.3.2	Am Ende der Anwendung	95
2.3.3	Java-Programme mit Exit-Code beenden *	96
2.4	Interaktive Anwendungen mit der Spring Shell	102
2.4.1	Eine Spring-Shell-Dependency einbinden	102
2.4.2	Eine erste interaktive Anwendung	103
2.4.3	Shell-Komponente schreiben: @ShellComponent, @ShellMethod ...	104
2.5	Die Verweisspritze	108
2.5.1	Objektgraphen aufbauen	108
2.5.2	Inversion of Control (IoC) und Dependency Injection	110
2.5.3	Injektionsarten	110
2.5.4	PhotoService und PhotoCommands	116
2.5.5	Mehrere Abhängigkeiten	119
2.5.6	Verhalten bei einer fehlenden Komponente	119
2.5.7	Optionale Abhängigkeit	121
2.5.8	Zyklische Abhängigkeiten *	123
2.5.9	Andere Dinge injizieren	124
2.6	Konfigurationsklassen und Fabrikmethode	125
2.6.1	@Configuration und @Bean	125
2.6.2	Parameter-Injektion einer @Bean-Methode	134
2.6.3	@Configuration-Bean und Lite-Bean	135
2.6.4	InjectionPoint *	137
2.6.5	Statische @Bean-Methoden *	140
2.6.6	@Import und @ImportSelector *	140
2.7	Abstraktion und Qualifizierungen	143
2.7.1	Der Name einer Bean und ihr Alias	143
2.7.2	AwtBicubicThumbnail für Vorschaubilder	144
2.7.3	Basistypen	148
2.7.4	ObjectProvider	157
2.7.5	@Order und @AutoConfigurationOrder *	160
2.7.6	Verhalten bei ausgewählten Vererbungsbeziehungen *	162
2.8	Lebenszyklus der Beans	165
2.8.1	@DependsOn	165
2.8.2	Verzögerte Initialisierung (lazy initialization)	166
2.8.3	Bean-initialisierung traditionell	168
2.8.4	InitializingBean und DisposableBean	174
2.8.5	Vererbung der Lebenszyklus-Methoden	175
2.8.6	*Aware-Schnittstellen *	176
2.8.7	BeanPostProcessor *	179

2.8.8	Spring-managed Beans woanders anmelden	183
2.8.9	Hierarchische Kontexte	185
2.8.10	Singleton und Prototyp zustandslos oder zustandsbehaftet	189
2.9	Annotationen aus JSR 330, Dependency Injection for Java *	189
2.10	Autokonfiguration	191
2.10.1	@Conditional und Condition	191
2.10.2	Wenn, dann: @ConditionalOn*	193
2.10.3	Die Debug-Protokollierung von Spring einschalten	201
2.10.4	Autokonfigurationen individuell steuern *	203
2.11	Spring Expression Language (SpEL)	206
2.11.1	ExpressionParser	207
2.11.2	SpEL in der Spring (Boot) API	208
3	Ausgewählte Module des Spring Frameworks	213
3.1	Hilfsklassen im Spring Framework	213
3.1.1	Bestandteile von org.springframework	213
3.2	Externe Konfiguration und das Environment	214
3.2.1	Situationsabhängige Konfigurationen	215
3.2.2	Environment	216
3.2.3	Werte mit @Value injizieren	217
3.2.4	Environment-Belegungen über @Value und \${...} bekommen	219
3.2.5	@Value und Default-Werte	220
3.2.6	Zugriff auf Konfigurationen	223
3.2.7	Geschachtelte/hierarchische Propertyts	229
3.2.8	Besondere Datentypen abbilden	234
3.2.9	Relaxed Bindings	240
3.2.10	Property-Sources	242
3.2.11	Spring-Profile definieren	257
3.2.12	Profile aktivieren	260
3.2.13	Spring-managed Beans je nach Profil	262
3.3	Ereignisbehandlung	264
3.3.1	Ereignisbehandlung	264
3.3.2	Context Events	264
3.3.3	ApplicationListener	266
3.3.4	@EventListener	267
3.3.5	Methoden der Ereignisklassen	268
3.3.6	Ereignisklassen schreiben und auf die Ereignisse reagieren	269

3.3.7	Ein Eventbus vom Typ ApplicationEventPublisher	269
3.3.8	Generische Ereignisse am Beispiel von PayloadApplicationEvent	271
3.3.9	Ereignis-Transformationen	273
3.3.10	Reihenfolgen durch @Order	274
3.3.11	Ereignisse nach Bedingungen filtern	274
3.3.12	Ereignisse synchron und asynchron	275
3.3.13	ApplicationEventMulticaster *	276
3.3.14	Ereignisse über ApplicationContext versenden	277
3.3.15	Listener an SpringApplication hängen *	277
3.3.16	Listener in spring.factories *	278
3.4	Ressourcen-Abstraktion mit Resource	279
3.4.1	InputStreamSource und Resource	280
3.4.2	Ressourcen laden	282
3.4.3	Ressourcen über @Value injizieren	282
3.5	Typkonvertierung mit ConversionService	284
3.5.1	ConversionService	284
3.5.2	DefaultConversionService und ApplicationConversionService	285
3.5.3	ConversionService als Spring-managed Bean	287
3.5.4	Bei der ConverterRegistry eigene Konverter anmelden	289
3.5.5	Printer und Parser	294
3.5.6	FormatterRegistry	296
3.5.7	DataBinder	298
3.6	Testgetriebene Entwicklung mit Spring Boot	302
3.6.1	Testbezogene Einträge vom Spring Initializr	302
3.6.2	Annotation @Test	303
3.6.3	Testfall für die Klasse FileSystem	304
3.6.4	Mehrschichtige Anwendungen testen, Objekte austauschen	308
3.6.5	Das Mocking-Framework Mockito	312
3.6.6	@InjectMocks	315
3.6.7	Verhalten verifizieren	316
3.6.8	Testen mit ReflectionTestUtils	317
3.6.9	Mit oder ohne Spring-Unterstützung	319
3.6.10	Test-Propertyts	322
3.6.11	@TestPropertySource	323
3.6.12	@ActiveProfiles	325
3.6.13	Stellvertreter einsetzen	325
3.6.14	@DirtiesContext	328
3.7	Scheibchen testen am Beispiel von JSON *	329
3.7.1	JSON	329
3.7.2	Jackson	330

3.7.3	Ein Java-Objekt in JSON schreiben	331
3.7.4	Testen von JSON-Mappings: @JsonTest	333
3.7.5	Die Abbildung mit @JsonComponent	337
3.8	Scheduling *	340
3.8.1	Scheduling-Annotationen und @EnableScheduling	341
3.8.2	@Scheduled	341
3.8.3	Nachteile von @Scheduled und Alternativen	344
3.9	Typen aus org.springframework.*.[lang util]	345
3.9.1	org.springframework.lang.*Null*-Annotationen	345
3.9.2	Paket org.springframework.util	346
3.9.3	Das Paket org.springframework.data.util	348
4	Ausgewählte Proxys	353
4.1	Proxy-Pattern	353
4.1.1	Proxy-Einsatz in Spring	355
4.1.2	Proxys dynamisch generieren	357
4.2	Caching	360
4.2.1	Optimierung durch Caching	360
4.2.2	Caching in Spring	361
4.2.3	Komponente mit @Cacheable implementieren	362
4.2.4	@Cacheable-Proxy nutzen	363
4.2.5	@Cacheable + condition	365
4.2.6	@Cacheable + unless	366
4.2.7	@CachePut	368
4.2.8	@CacheEvict	368
4.2.9	Eigene Schlüsselgeneratoren angeben	369
4.2.10	@CacheConfig	376
4.2.11	Cache-Implementierungen	376
4.2.12	Caching mit Caffeine	377
4.2.13	Das Caching im Test abschalten	378
4.3	Asynchrone Aufrufe	379
4.3.1	@EnableAsync und @Async-Methoden	380
4.3.2	Beispiel mit @Async	380
4.3.3	Der Rückgabetypp CompletableFuture	383
4.4	TaskExecutor *	388
4.4.1	TaskExecutor-Implementierungen	389
4.4.2	Executor setzen und Ausnahmen behandeln	391

4.5	Spring und Bean Validation	393
4.5.1	Parameterprüfungen	393
4.5.2	Die Jakarta Bean Validation (JSR 303)	394
4.5.3	Dependency auf die Spring-Boot-Starter-Validation	395
4.5.4	»Photo« mit Jakarta-Bean-Validation-Annotationen	395
4.5.5	Einen Validator injizieren und prüfen	397
4.5.6	Spring und die Bean-Validation-Annotationen	400
4.5.7	Bean-Validation-Annotationen an Methoden	401
4.5.8	Validierung überall, am Beispiel einer Konfiguration	402
4.5.9	Validierung testen *	403
4.6	Spring Retry *	407
4.6.1	Das Spring-Retry-Projekt	407
4.6.2	@Retryable	409
4.6.3	Fallback mit @Recover	413
4.6.4	RetryTemplate	414
5	Relationale Datenbanken anbinden	417
<hr/>		
5.1	Eine H2-Datenbank aufsetzen	417
5.1.1	Kurzvorstellung der H2-Datenbank	417
5.1.2	H2 installieren und starten	418
5.1.3	Über die H2 Console eine Verbindung zur Datenbank aufbauen	422
5.1.4	Das Datenbankschema der Date4u-Datenbank	423
5.2	Datenbankzugriffe mit Spring realisieren	426
5.2.1	Spring-Helfer	427
5.3	Der Spring Boot Starter JDBC	428
5.3.1	Den JDBC-Starter in die POM aufnehmen	428
5.3.2	JDBC-Verbindungsdaten eintragen	429
5.3.3	DataSource oder JdbcTemplate injizieren	430
5.3.4	Connection-Pooling	432
5.3.5	JDBC-Zugriffe loggen	433
5.3.6	Das Paket org.springframework.jdbc und seine Unterpakete	434
5.3.7	DataAccessException	435
5.3.8	Autokonfiguration für DataSource *	437
5.3.9	Mehrere Datenbanken ansprechen	438
5.3.10	DataSourceUtils *	440

5.4	JdbcTemplate	442
5.4.1	Beliebiges SQL ausführen: execute(...)	442
5.4.2	SQL-Aktualisierungen: update(...)	443
5.4.3	Einzelne Werte erfragen: queryForObject(...)	443
5.4.4	Einen Platzhalter für ein PreparedStatement definieren	444
5.4.5	Ganze Zeile erfragen: queryForMap(...)	446
5.4.6	Mehrere Zeilen mit einem Element erfragen: queryForList(...)	446
5.4.7	Mehrere Zeilen und Spalten einlesen: queryForList(...)	447
5.5	Datentypen zum Mapping auf Ergebnisse	448
5.5.1	RowMapper	449
5.5.2	RowCallbackHandler	454
5.5.3	ResultSetExtractor	455
5.6	NamedParameterJdbcTemplate	459
5.6.1	Typbeziehungen von *JdbcTemplate	460
5.6.2	Methoden vom NamedParameterJdbcTemplate	460
5.6.3	Werte von NamedParameterJdbcTemplate mit Map übergeben	461
5.6.4	SqlParameterSource	461
5.6.5	Rückgriff auf tiefer liegende Objekte *	464
5.7	Batch-Operationen *	465
5.7.1	Batch-Methoden beim JdbcTemplate	465
5.7.2	BatchPreparedStatementSetter	466
5.7.3	Batch-Methoden beim NamedParameterJdbcTemplate	471
5.7.4	SqlParameterSourceUtils	472
5.7.5	Konfigurations-Property	473
5.8	Es wird groß mit BLOB und CLOB *	474
5.8.1	SqlLobValue	474
5.8.2	LobHandler und DefaultLobHandler	475
5.8.3	LOBs über AbstractLobStreamingResultSetExtractor lesen	477
5.9	Das Paket org.springframework.jdbc.core.simple *	478
5.9.1	SimpleJdbcInsert	478
5.10	Das Paket org.springframework.jdbc.object *	480
5.10.1	MappingSqlQuery	481
5.11	Transaktionen	483
5.11.1	Das ACID-Prinzip	483
5.11.2	Lokal oder global/verteilt	484
5.11.3	JDBC-Transaktionen/Auto-Commit	484
5.11.4	PlatformTransactionManager	485
5.11.5	TransactionTemplate	490
5.11.6	@Transactional	494

6	Jakarta Persistence mit Spring	497
6.1	Welt der Objekte und der Datenbanken	497
6.1.1	Transient und persistent	497
6.1.2	Abbildung von Objekten auf Tabellen	498
6.1.3	Java-Bibliotheken für das O/R-Mapping	502
6.2	Jakarta Persistence	503
6.2.1	Persistence Provider	504
6.2.2	Jakarta Persistence Provider nutzen JDBC	505
6.2.3	Was deckt die Jakarta Persistence ab?	505
6.3	Spring Data JPA	506
6.3.1	Spring Boot Starter Data JPA einbinden	506
6.3.2	Konfigurationen	507
6.4	Die Jakarta Persistence Entity-Bean	509
6.4.1	Eine Entity-Bean-Klasse entwickeln	510
6.5	Die Jakarta Persistence API	517
6.5.1	Den EntityManager von Spring bekommen	517
6.5.2	Eine Entity-Bean über ihren Schlüssel suchen: find(...)	519
6.5.3	find(...) und getReference(...)	522
6.5.4	Abfragemöglichkeiten mit dem EntityManager	523
6.6	Die Jakarta Persistence Query Language (JPQL)	524
6.6.1	Ein JPQL-Beispiel mit SELECT und FROM	524
6.6.2	JPQL-Abfragen mit createQuery(...) aufbauen und absenden	526
6.6.3	Bedingungen in WHERE	529
6.6.4	JPQL-Aufrufe parametrisieren	529
6.6.5	JPQL-Operatoren und -Funktionen	533
6.6.6	Rückgaben ordnen mit ORDER BY	537
6.6.7	Projektion auf skalare Werte	538
6.6.8	Aggregatfunktionen	539
6.6.9	Projektion auf mehrere Werte	540
6.6.10	Benannte deklarative Abfragen (Named Queries)	544
6.7	Datenbankfunktionen aufrufen und native SQL-Anfragen senden	546
6.7.1	Datenbankfunktionen aufrufen: FUNCTION(...)	547
6.7.2	createNativeQuery(...) über EntityManager nutzen	547
6.7.3	@NamedNativeQuery	549
6.7.4	@NamedNativeQuery mit resultSetMapping	550
6.8	Schreibender Zugriff mit dem EntityManager in Transaktionen	553
6.8.1	Transaktionale Operationen	553
6.8.2	persist(...)	554

6.8.3	EntityTransaction	554
6.8.4	PlatformTransactionManager → JpaTransactionManager	555
6.8.5	@Transactional	557
6.8.6	Speichern vs. Aktualisieren	558
6.8.7	remove(...)	558
6.8.8	Synchronisation bzw. Flush	559
6.8.9	Query mit UPDATE und DELETE	559
6.9	Persistence Context und weitere Transaktionssteuerungen	560
6.9.1	Zustände einer Entity-Bean	561
6.10	Weiterführende OR-Metadaten	565
6.10.1	Der Database-first- und der Code-first-Ansatz	565
6.10.2	Den Tabellennamen über @Table setzen	565
6.10.3	Den @Entity-Namen ändern	566
6.10.4	Persistente Attribute	567
6.10.5	@Basic und @Transient	568
6.10.6	Spaltenbeschreibung und @Colum	569
6.10.7	Entity-Bean-Datentypen	571
6.10.8	Datentypen mit AttributeConverter mappen	574
6.10.9	Schlüsselkennzeichnung	576
6.10.10	Eingebettete Typen	581
6.10.11	Eine Entity-Bean erbt Eigenschaften von einer Oberklasse	583
6.11	Beziehungen zwischen Entitäten	585
6.11.1	Unterstützte Assoziationen und Beziehungstypen	585
6.11.2	Die 1:1-Beziehung	586
6.11.3	Bidirektionale Beziehungen	590
6.11.4	Die 1:n-Beziehung	591
6.11.5	n:m-Beziehungen *	598
6.12	FetchType: Lazy und Eager Loading	599
6.12.1	Aufzählung mit FetchType	599
6.12.2	Der Hibernate-Typ PersistentBag	600
6.12.3	Das 1 + N Query-Problem ist ein Performance-Antipattern	602
6.13	Cascading	605
6.13.1	Der Aufzählungstyp CascadeType	606
6.13.2	CascadeType setzen	608
6.13.3	cascade=REMOVE vs. orphanRemoval=true	609
6.14	Repositorys	610
6.14.1	Datenzugriffsschicht	610
6.14.2	Methoden im Repository	612
6.14.3	SimpleJpaRepository	614

7	Spring Data JPA	619
7.1	Welche Aufgaben erfüllt Spring Data?	619
7.2	Spring Data Commons: CrudRepository	621
7.2.1	Der Typ CrudRepository	622
7.2.2	JPA-basierte Repositories	624
7.3	Untertypen von CrudRepository: JpaRepository etc.	627
7.3.1	ListCrudRepository	627
7.3.2	Technologiespezifische [List]CrudRepository-Untertypen	628
7.3.3	Ausgewählte Methoden über Repository	630
7.4	Blättern und Sortieren mit [List]PagingAndSortingRepository	632
7.4.1	Der Typ Sort	633
7.4.2	Sort.TypedSort<T>	635
7.4.3	Pageable und PageRequest	636
7.4.4	Sortieren von paginierten Seiten	641
7.5	QueryByExampleExecutor *	642
7.5.1	Die Probe	643
7.5.2	QueryByExampleExecutor	643
7.5.3	Probe in das Example	644
7.5.4	ExampleMatcher aufbauen	645
7.5.5	Property ignorieren	646
7.5.6	String-Vergleichstechniken setzen	647
7.5.7	Individuelle Regeln mit GenericPropertyMatcher setzen	648
7.5.8	PropertyValueTransformer	648
7.6	Eigene Abfragen mit @Query formulieren	650
7.6.1	Die @Query-Annotation	650
7.6.2	Verändernde @Query-Operationen mit @Modifying	652
7.6.3	IN-Parameter durch Array/Vararg/Collection füllen	653
7.6.4	@Query mit JPQL-Projektion	653
7.6.5	Sort- und Pageable-Parameter	654
7.6.6	Neue Query-Methoden ergänzen	656
7.6.7	Querys mit SpEL-Ausdrücken	656
7.6.8	Die @NamedQuery einer Entity-Bean verwenden	657
7.6.9	Die @Query-Annotation mit nativem SQL	658
7.7	Stored Procedures (gespeicherte Prozeduren) *	661
7.7.1	Eine gespeicherte Prozedur in H2 definieren	662
7.7.2	Eine gespeicherte Prozedur über eine native Query aufrufen	663
7.7.3	Eine gespeicherte Prozedur mit @Procedure aufrufen	664

7.8	Derived Query Methods	665
7.8.1	Individuelle CRUD-Operationen über Methodennamen	665
7.8.2	Der Query-Builder-Mechanismus	666
7.8.3	Rückgaben von Derived Query Methods	670
7.8.4	Asynchrone Query-Methoden	671
7.8.5	Streamende Query-Methoden	671
7.8.6	Vorteile und Nachteile der Derived Query Methods	671
7.9	Die Criteria API und der JpaSpecificationExecutor	672
7.9.1	Die Criteria API	673
7.9.2	Die funktionale Schnittstelle Specification	674
7.9.3	JpaSpecificationExecutor	675
7.9.4	Methoden in JpaSpecificationExecutor	676
7.9.5	Specification-Implementierungen	676
7.9.6	Specification-Instanzen zusammensetzen	678
7.9.7	Interna *	679
7.9.8	Metamodell-Klassen	680
7.9.9	Alles chic?	683
7.10	Alternativen zu JDBC Jakarta Persistence	684
7.10.1	Querydsl	685
7.10.2	Spring Data JDBC	692
7.11	Gutes Design mit Repositorys	697
7.11.1	Das Gegenteil von wünschenswert	697
7.11.2	Denke an ISP und die Zwiebel!	699
7.11.3	Das Fragment-Interface	701
7.12	Projektionen	702
7.12.1	Projektionen in Spring Data	703
7.12.2	Die Interface-basierte Projektion	703
7.12.3	Projektionen mit SpEL-Ausdrücken	705
7.12.4	Projektionen mit Default-Methoden	706
7.12.5	Klassenbasierte Projektionen	707
7.12.6	Dynamische Projektionen	707
7.13	Auditing *	709
7.13.1	Auditing mit Spring Data	709
7.13.2	Auditing mit Spring Data JPA	709
7.13.3	AuditorAware für User-Informationen	711
7.13.4	Ausblick: Spring Data Envers	712
7.14	Incremental Data Migration	713
7.14.1	Lang leben die Daten	713
7.14.2	Evolutionäres Datenbankdesign	714

7.14.3	Incremental Data Migration mit Flyway	715
7.14.4	Flyway in Spring Boot: Migrationskripte	715
7.14.5	Migrations in Java-Code	718
7.14.6	Flyway-Migrations außerhalb von Spring	719
7.15	Die Datenzugriffsschicht testen	720
7.15.1	Was wollen wir testen?	720
7.15.2	Testscheibchen	721
7.15.3	Eine In-Memory-Testdatenbank einsetzen	722
7.15.4	Eigene Verbindungsdaten zur Testdatenbank zuweisen	723
7.15.5	Tabellen aufbauen mit Initialisierungsskripten	724
7.15.6	Das Projekt Testcontainers	726
7.15.7	Demodaten	728
7.15.8	@Sql und @SqlGroup	729
7.15.9	TestEntityManager	729
8	Spring Data für NoSQL-Datenbanken	731
8.1	Not only SQL	731
8.2	MongoDB	732
8.2.1	MongoDB: Dokumente und Collections	732
8.2.2	Über MongoDB	733
8.2.3	Den MongoDB Community Server installieren und starten	734
8.2.4	GUI-Werkzeuge für MongoDB	735
8.2.5	Spring Data MongoDB	736
8.2.6	MongoDB APIs	737
8.2.7	MongoDB-Dokumente	738
8.2.8	Die Klasse MongoTemplate	739
8.2.9	MongoDB-Repositoryys	742
8.2.10	MongoDB-Programme testen	744
8.3	Elasticsearch	745
8.3.1	Textsuche ist anders	745
8.3.2	Apache Lucene	746
8.3.3	Dokument und Feld	747
8.3.4	Index	747
8.3.5	Die Schwächen von Apache Lucene	748
8.3.6	Lucene-Aufsätze: Elasticsearch und Apache Solr	748
8.3.7	Elasticsearch installieren und starten	749
8.3.8	Spring Data Elasticsearch	751
8.3.9	Dokumente	751

8.3.10	ElasticsearchRepository	754
8.3.11	@DataElasticsearchTest	757

9 Spring Web 759

9.1	Webserver	759
9.1.1	Java-Webserver	760
9.1.2	Spring Boot Starter Web	761
9.1.3	Andere Webserver einsetzen *	762
9.1.4	Port anpassen über server.port	762
9.1.5	Statische Ressourcen servieren	763
9.1.6	WebJars	764
9.1.7	TLS-Verschlüsselung	765
9.2	Dynamische Inhalte generieren	766
9.2.1	Steinzeit: Das Common Gateway Interface (CGI)	766
9.2.2	Der Servlet-Standard	767
9.2.3	@WebServlet programmieren	768
9.2.4	Schwächen von Servlets	770
9.3	Spring Web MVC	771
9.3.1	Spring-Container in Web-Anwendungen	771
9.3.2	@Controller und @ResponseBody	772
9.3.3	@RestController	774
9.3.4	Controller → [Service →] Repository	776
9.4	Hot Code Swapping	777
9.4.1	Hot Swapping der JVM	778
9.4.2	Spring Developer Tools	778
9.5	Das Hypertext Transfer Protocol (HTTP)	779
9.5.1	HTTP-Request und -Response	779
9.5.2	Einen HTTP-Client zum Testen von Anwendungen aufsetzen	780
9.6	Request Matching	781
9.6.1	@RequestMapping	781
9.6.2	@*Mapping	782
9.6.3	Allgemeinere Pfadausdrücke und Pfad-Matcher	782
9.6.4	@RequestMapping am Typ	782
9.7	Response senden	784
9.7.1	HttpMessageConverter in der Anwendung	784
9.7.2	Formatkonvertierungen der Rückgaben von Handler-Methoden	785

9.7.3	Abbildung auf JSON-Dokumente	785
9.7.4	ResponseEntity = Statuscode + Header + Body	792
9.8	Request auswerten	794
9.8.1	Handler-Methoden mit Parametern	794
9.8.2	Datenübertragung vom Client zum Controller	795
9.8.3	Datenannahme über Parameter	796
9.8.4	Query-Parameter auswerten	796
9.8.5	Optionale Query-Parameter	797
9.8.6	Alle Query-Parameter mappen	798
9.8.7	Eine Pfadvariable auswerten	800
9.8.8	MultipartFile	803
9.8.9	Header auswerten	805
9.8.10	HttpEntity mit ihren Unterklassen RequestEntity und ResponseEntity *	806
9.9	Typkonvertierung der Parameter	806
9.9.1	Beispiel für YearMonth-Converter	807
9.9.2	@DateTimeFormat und @NumberFormat	808
9.9.3	Query-Parameter und Formulardaten auf eine Bean mappen	809
9.9.4	Eigene Typkonverter anmelden	812
9.9.5	URI-Template-Pattern mit regulären Ausdrücken	815
9.10	Ausnahmebehandlung und Fehlermeldung	816
9.10.1	Ausnahmen selbst auf Statuscodes mappen	816
9.10.2	Eskaliert	816
9.10.3	Die Konfigurations-Propertys server.error.*	818
9.10.4	Exception-Resolver	821
9.10.5	Exception auf Statuscode mit @ResponseStatus mappen	821
9.10.6	ResponseStatusException	823
9.10.7	Lokales Controller-Exception-Handling mit @ExceptionHandler	825
9.10.8	RFC 7807: »Problem Details for HTTP APIs«	827
9.10.9	Globales Controller-Exception-Handling mit Controller-Advice	829
9.11	RESTful API	830
9.11.1	Die Prinzipien hinter REST	830
9.11.2	REST-Endpunkte für Profile implementieren	836
9.11.3	Data Transfer Objects (DTO)	839
9.11.4	Best Practice: Einmal bitte alles? Nein!	841
9.11.5	GET und DELETE auf individuelle Ressourcen	842
9.11.6	POST und PUT mit @RequestBody	843
9.11.7	UriComponents	846

9.12 Asynchrone Web-Requests *	848
9.12.1 Lange Abfragen blockieren den Worker-Thread	848
9.12.2 Asynchron in die Ausgabe schreiben	849
9.12.3 Eine Handler-Methode liefert Callable	849
9.12.4 WebAsyncTask	850
9.12.5 Eine Handler-Methode liefert DeferredResult	850
9.12.6 StreamingResponseBody	851
9.13 Spring Data Web Support	852
9.13.1 Laden aus dem Repository	852
9.13.2 Pageable und Sort als Parametertypen	853
9.13.3 Die Rückgabe Page	856
9.14 Dokumentation einer RESTful API mit OpenAPI	856
9.14.1 Beschreibung einer RESTful API	857
9.14.2 Die OpenAPI-Spezifikation	857
9.14.3 Woher kommt das OpenAPI-Dokument?	858
9.14.4 OpenAPI mit Spring	859
9.14.5 springdoc-openapi	859
9.14.6 Bessere Dokumentation mit OpenAPI-Annotationen	861
9.14.7 Java-Code aus einem OpenAPI-Dokument generieren	862
9.14.8 Spring REST Docs	864
9.15 Testen der Webschicht	865
9.15.1 QuoteRestController testen	865
9.15.2 Die Annotation @ WebMvcTest	866
9.15.3 Eine Testmethode mit MockMvc schreiben	867
9.15.4 REST-Endpunkte mit dem Server testen	870
9.15.5 WebTestClient	871
9.16 Best Practices bei der Nutzung einer RESTful API	872
9.16.1 Jakarta Bean Validation	873
9.16.2 Ressourcen-ID	874
9.16.3 Data Transfer Objects (DTO) mappen	874
9.16.4 Versionierung von RESTful Webservices	877
9.17 Webanwendungen mit Spring Security absichern	878
9.17.1 Die Bedeutung von Spring Security	878
9.17.2 Dependency auf Spring Boot Starter Security	879
9.17.3 Authentication	879
9.17.4 SecurityContext und SecurityContextHolder	880
9.17.5 AuthenticationManager	882
9.17.6 SpringBootWebSecurityConfiguration	884

9.17.7	AuthenticationManager und ProviderManager	886
9.17.8	Die Schnittstelle UserDetailsService	887
9.17.9	Die Spring-managed Bean PasswordEncoder	892
9.17.10	BasicAuthenticationFilter	896
9.17.11	Zugriff auf den Benutzer	896
9.17.12	Authorization und das Cookie	898
9.17.13	Token-basierte Authentifizierung mit JWT	898
9.18	RESTful Webservices konsumieren	907
9.18.1	Klassen zum Ansprechen von HTTP-Endpunkten	907
9.18.2	Ein WebClient-Beispiel	908
9.18.3	Deklarative Webservice-Clients	914

10 Logging und Monitoring 921

10.1	Logging	921
10.1.1	Warum ein Protokoll erstellen?	921
10.1.2	Log-Group	922
10.2	Logging-Implementierung	922
10.2.1	Das Logging-Pattern-Layout	924
10.2.2	Die Logging-Konfiguration ändern	925
10.2.3	Banner	925
10.2.4	Logging zur Startzeit	927
10.2.5	Testen von geschriebener Log-Meldung	927
10.3	Anwendungen mit Spring Boot Actuator überwachen	928
10.3.1	Den Gesundheitszustand über den Actuator ermitteln	928
10.3.2	Aktivierung der Endpunkte	930
10.3.3	Info-Ergänzungen	932
10.3.4	Parameter und JSON-Rückgaben	932
10.3.5	Neue Actuator-Endpunkte	934
10.3.6	Bin ich gesund?	934
10.3.7	HealthIndicator	936
10.3.8	Metriken	936
10.4	Micrometer und Prometheus	937
10.4.1	Micrometer	937
10.4.2	Prometheus: Die Software	940

11 Build und Deployment 945

11.1	Spring-Boot-Programme verpacken und ausführen	945
11.1.1	Deployment-Optionen	945
11.1.2	Ein Spring-Boot-Programm über Maven starten	946
11.1.3	Ein Spring-Boot-Programm in JAR packen	946
11.1.4	Das spring-boot-maven-plugin	947
11.2	Spring-Anwendungen im OCI-Container	948
11.2.1	Container	948
11.2.2	Docker installieren und nutzen	950
11.2.3	H2 starten, stoppen, Port-Weiterleitung und Data-Volumes	952
11.2.4	Eine Spring-Boot-Docker-Anwendung vorbereiten	955
11.2.5	Docker Compose	957
11.2.6	Anwendungen beenden mit einem Actuator-Endpunkt	958

Anhang 961

A	Migration von Spring Boot 2 auf Spring Boot 3	961
A.1	Vorbereitung	961
A.2	Jakarta EE 9	961
A.3	Gibt es weitere Neuerungen?	962
A.4	Spring-boot-properties-migrator	963
A.5	Der Spring Boot Migrator (SBM)	963
A.6	Dependency-Upgrades	964

Index		967
-------------	--	-----