

Spring Boot 3 und Spring Framework 6

Das umfassende Handbuch

» Hier geht's
direkt
zum Buch

DAS VORWORT

Vorwort

Was ist Spring Boot?

Spring Boot ist aktuell das wichtigste Java-Enterprise-Framework und bietet der Entwicklergemeinschaft viele Vorteile: Es ist leicht zu lernen und zu verwenden und erleichtert die Entwicklung von Microservices. Richtig entwickelt, sind Spring-Boot-Anwendungen skalierbar und eignen sich daher perfekt für robuste Unternehmensanwendungen.

Vorkenntnisse und Zielgruppe

Spring ist ein sehr beliebtes Framework in der Java-Community und hat sich in den vergangenen Jahren zu einem Quasistandard entwickelt. Daher wurde dieses Buch für alle konzipiert, die Unternehmensanwendungen mit Datenbankanbindung und Webservices entwickeln möchten. Es ist sowohl für Anfänger als auch für Fortgeschrittene geeignet und liefert zahlreiche praktische Beispiele.

Voraussetzung sind robuste Java-Kenntnisse und der sichere Umgang mit Werkzeugen wie Maven. Mit relationalen Datenbanken sollte die Leserschaft grundlegend vertraut sein.

Die Demoanwendung Date4u, Aufgaben und Lösungen

Mit diesem Handbuch und einer Entwicklungsumgebung des Vertrauens lassen sich Spring-Programme entwickeln. Um ein neues Framework zu erlernen, reicht das Lesen aber nicht aus. Wer eine Programmiersprache oder ein Framework erlernen möchte, muss sie bzw. es wie eine Fremdsprache üben und »sprechen«, und neben der Grammatik der Sprache ist Spring das Vokabular.

Zur Demonstration wird in diesem Buch eine Dating-Anwendung für Einhörner implementiert. Dieses unterhaltsame Beispiel demonstriert alle zentralen Bestandteile einer Enterprise-Anwendung: das Zusammenspiel von Komponenten, die Speicherung der Profildaten in einer Datenbank und den Zugriff über Webservices. Die Leserschaft wird Schritt für Schritt durch die Anwendung geleitet, und im Text eingestreut finden sich immer wieder Aufgaben, die zur Übung einladen. Zu jeder Aufgabe folgt ein Lösungsvorschlag, sodass sich der Text linear lesen lässt. Die Anwendung *Date4u* ist per Download verfügbar (siehe <https://rheinwerk-verlag.de/5980>).

Organisation der Kapitel

Das Buch gliedert sich in folgende Kapitel:

Kapitel 1, »Einleitung«, stellt das *Spring Framework* und die »Erweiterung« *Spring Boot* vor. Es wird gezeigt, wie ein Projekt angelegt wird. Außerdem werden der Aufbau der Maven-POM-Datei sowie die Bedeutung der Starter und Dependencies erläutert.

In **Kapitel 2, »Container für Spring-managed Beans«**, steht der Spring-Kontext im Mittelpunkt. Schritt für Schritt wird aus einer einzigen Klasse ein Geflecht von Spring-managed Beans aufgebaut, um Bilder im Dateisystem abzulegen und zu laden.

Die Verwaltung von Spring-managed Beans ist eine Kernaufgabe des Spring Frameworks, aber es gibt viele weitere Features, die **Kapitel 3, »Ausgewählte Module des Spring Frameworks«**, anspricht. Dazu zählen die externe Konfiguration, die Ereignisbehandlung, Konvertierungsklassen und diverse Utility-Klassen.

In **Kapitel 4, »Ausgewählte Proxies«**, geht es mit besonderen »Ring« weiter, die sich um Komponenten legen und damit das Caching, asynchrone Aufrufe oder die Validierung übernehmen. Das bietet den Vorteil, dass bestimmte Aufgaben an das Framework verlagert werden können, sodass der eigene Code schlanker wird.

In den ersten vier Kapiteln stehen der Spring-Container und Hilfsklassen im Mittelpunkt, aber keine Speichertechnologien. Das ändert sich in **Kapitel 5, »Relationale Datenbanken anbinden«**. In diesem Kapitel wird ein Datenbankmanagementsystem vorbereitet und werden SQL-Anfragen an die Datenbank demonstriert.

Das Schreiben in Datenbanken mit der JDBC-API ist einfach und direkt, aber unkomfortabel. Daher führt **Kapitel 6, »Jakarta Persistence mit Spring«**, in das objektrelationale Mapping ein.

Datenzugriffsschichten sind in jeder größeren Anwendung nötig. **Kapitel 7, »Spring Data JPA«**, stellt ihre Möglichkeiten exemplarisch an dem Familienmitglied *Spring Data JPA* vor. Repositories lassen sich damit einfach schreiben, und viele Abfragen und Datentransformationen werden vom Framework realisiert.

Relationale Datenbanken werden flankiert von nichtrelationalen Datenbanksystemen, die ebenfalls ausgezeichnet in die Spring-Data-Familie integriert sind. **Kapitel 8, »Spring Data für NoSQL-Datenbanken«**, zeigt anhand von MongoDB und Elasticsearch, wie die Spring-Data-Konzepte in die NoSQL-Welt übertragen werden.

Nachdem mit der Datenhaltung ein wichtiger Teil der Infrastruktur beschrieben wurde, geht es mit Anwendungen des HTTP-Protokolls in **Kapitel 9, »Spring Web«**, weiter. Dort stehen RESTful Webservices und HTTP-Clients im Mittelpunkt.

Das **Kapitel 10, »Spring AI«**, bietet einen Einblick in die neue API zur Nutzung von Large Language Models.

Mit dem bisher erworbenen Wissen lässt sich eine Anwendung betreiben, aber es ist wichtig, auch von außen in die Anwendung schauen zu können, um zu prüfen, ob die »Betriebstemperatur« stimmt. **Kapitel 11, »Logging und Monitoring«**, stellt mit dem Actuator-Projekt eine Möglichkeit vor, Daten einer Anwendung freizugeben, die dann von externen Lösungen abgeholt und visualisiert werden können.

Dass ein Programm in der Entwicklungsumgebung läuft und bei den Testfällen nicht versagt, ist gut, aber nur die halbe Miete: Ein Programm muss ausgespielt werden. Dazu stellt **Kapitel 12, »Build und Deployment«**, unter anderem OCI-Container (aka Docker) vor.

Die *-Abschnitte

Das Spring-Universum ist reich an Feinheiten und verwirrt Einsteiger und Einsteigerinnen oft. Um das nötige Wissen vom verzichtbaren Wissen zu trennen, enden einige Überschriften mit einem *, was bedeutet, dass dieser Abschnitt übersprungen werden kann, ohne dass der Leserschaft etwas Wesentliches für die späteren Kapitel fehlt.

Welche Java-Version wird im Buch genutzt?

Die Mindestanforderung für die Nutzung von Spring Boot 3 ist Java 17. Da Java 21 jedoch die neueste Long-Term-Support-(LTS-)Version darstellt – was bedeutet, dass Laufzeitumgebungen von den Herstellern über viele Jahre hinweg mit Updates und Support versorgt werden – basiert das Buch auf Java 21. Spring Boot 3 ist auch mit aktuelleren Java-Versionen wie Java 23 kompatibel.

Benötigte Software

Wollen wir Java-Programme laufen lassen, benötigen wir eine JVM. In der Anfangszeit war das einfach. Die Laufzeitumgebung stammte erst von Sun Microsystems, später von der Firma Oracle, die Sun übernommen hat. Heute ist das deutlich unübersichtlicher. Diverse Institutionen kompilieren das *OpenJDK* (die Originalquellen enthalten Hunderttausende Zeilen C[++] und Java-Quellcode) und bündeln es in eine Distribution, die Java-Laufzeitumgebung. Die bekanntesten Laufzeitumgebungen sind:

- ▶ *Eclipse Adoptium* (<https://adoptium.net>),
- ▶ *Amazon Corretto* (<https://aws.amazon.com/de/corretto>),
- ▶ *Red Hat OpenJDK* (<https://developers.redhat.com/products/openjdk/overview>)

Hinzu kommen weitere, wie die von *Azul Systems* oder *Bellsoft*. Zum Lernen mit diesem Buch kann man sich frei für eine Distribution entscheiden; mit Adoptium liegt man gut.

Die Entwicklungsumgebung

Java-Quellcode ist nur Text, sodass im Prinzip ein einfacher Texteditor reicht. Allerdings können wir mit einem Editor wie Notepad keine große Produktivität erwarten. Moderne Entwicklungsumgebungen unterstützen uns in vielerlei Hinsicht: durch farbige Hervorhebung von Schlüsselwörtern, automatische Codevervollständigung, intelligente Fehlerkorrektur, durch das Einfügen von Codeblöcken, die Visualisierung von Zuständen im Debugger und durch vieles mehr. Es ist daher ratsam, eine vollständige Entwicklungsumgebung zu verwenden. Drei populäre IDEs sind: *IntelliJ*, *Eclipse* und *Visual Studio Code*. Hingegen verliert (*Apache*) *NetBeans* immer weiter den Anschluss.

Wie bei den Java-Laufzeitumgebungen ist es jedem und jeder überlassen, sich eine Entwicklungsumgebung auszuwählen. Eclipse, NetBeans und Visual Studio Code sind frei und quelloffen. Die *IntelliJ Community Edition* ist ebenfalls frei und quelloffen, die mächtigere *IntelliJ Ultimate Edition* kostet hingegen Geld. Die Ultimate-Version der IntelliJ ist sicherlich die leistungsfähigste Java-IDE auf dem Markt, und ihre Unterstützung von Spring ist vorbildlich und von anderen IDEs unerreicht.

Konventionen

In diesem Buch wird eine Reihe von Konventionen verwendet.

Neu eingeführte Begriffe sind *kursiv* gesetzt, und der Index verweist genau auf diese Stelle. Des Weiteren erscheinen *Dateinamen*, *HTTP-Adressen* und *Dateiendungen* (*.txt*) in kursiver Schrift.

Begriffe der Benutzeroberfläche stehen in KAPITÄLCHEN.

Für Code und Listings gelten weitere Regeln:

- ▶ Listings, Methoden und sonstige Programmelemente sind in nichtproportionaler Schrift gesetzt.
- ▶ An einigen Stellen wurde hinter eine Listingzeile ein abgeknickter Pfeil (↵) als Sonderzeichen gesetzt, das den Zeilenumbruch markiert. Der Code aus der nächsten Zeile gehört also noch zur vorangehenden. Bei Methodennamen und Konstruktoren folgt immer ein Klammerpaar, um die Methoden/Konstruktoren von Objekt-/Klassenvariablen abgrenzen zu können. So ist bei der Schreibweise `System.out` und

`System.gc()` klar, dass Ersteres eine statische Variable ist und Letzteres eine statische Methode.

- ▶ Hat eine Methode bzw. ein Konstruktor Parameter, so stehen sie in Klammern: `run(String... args)`. In der Regel wird der Parametername ausgelassen, also heißt es kurz: `run(String...)`. Ist der Rückgabotyp im Kontext relevant, steht er mit dabei, etwa so: `ConfigurableApplicationContext run(String... args)` bzw. `ConfigurableApplicationContext run(String...)`. Hat eine Methode bzw. ein Konstruktor eine Parameterliste, ohne dass diese gerade relevant ist, wird sie mit Auslassungspunkten abgekürzt. Beispiel: »Eine `run(...)`-Methode startet den Container.« Ein leeres Klammerpaar bedeutet demnach, dass eine Methode bzw. ein Konstruktor wirklich keine Parameterliste hat.
- ▶ Um die Parameterliste kurz und dennoch präzise zu machen, gibt es im Text teilweise Angaben wie `getProperty(String key[, String def])`, was eine Abkürzung für »`getProperty(String key)` und `getProperty(String key, String def)`« ist. Auch an anderen Stellen finden sich die Auslassungspunkte ..., wenn die Implementierung oder Bildschirmausgaben für das Verständnis nicht weiter erforderlich sind.
- ▶ Um eine Gruppe von Methoden anzugeben, symbolisiert die Kennung * einen Platzhalter. So steht zum Beispiel `print*(...)` für die Methoden `println(...)`, `print(...)` und `printf(...)`. Aus dem Kontext geht hervor, welche Methoden gemeint sind.
- ▶ Lange Paketnamen werden teilweise abgekürzt. Zum Beispiel wird aus `org.springframework.data.jpa.repository` die Kurzform `o.s.d.j.r.`
- ▶ Kommen Annotationen im Text vor, ist es eigentlich doppelt gemoppelt, etwa »@Value-Annotation« zu schreiben, doch erlaubt das @-Zeichen im Text das schnellere Erfassen für die Leserschaft.



Um im Programmcode Compilerfehler oder Laufzeitfehler anzuzeigen, steht in der Zeile ein ☠. So ist auf den ersten Blick abzulesen, dass die Zeile entweder nicht kompiliert wird oder zur Laufzeit aufgrund eines Programmierfehlers eine Ausnahme auslöst. Beispiel:

```
SpringApplication.run( WebAppApplication.class, null )  
// java.lang.IllegalArgumentException: Args must not be null ☠
```

Teilweise werden von der Kommandozeile (synonym: *Befehlszeile, Konsole, Shell*) aus Programme aufgerufen. Da jedes Kommandozeilenprogramm eine eigene Prompt-Sequenz hat, wird diese hier im Buch generisch durch ein \$ symbolisiert. Unsere Eingaben sind fett gesetzt. Ein Beispiel:

```
$ java -version  
openjdk version "21.0.1" 2023-10-17  
OpenJDK Runtime Environment (build 21.0.1+12-29)  
OpenJDK 64-Bit Server VM (build 21.0.1+12-29, mixed mode, sharing)
```

An Stellen, an denen ausdrücklich die Windows-Kommandozeile gemeint ist, benutze ich das Prompt-Zeichen >:

```
> ver
```

```
Microsoft Windows [Version 10.0.19041.388]
```

Programmlistings

Bei den Listings finden sich in der Regel nur die relevanten Codeausschnitte, um den Umfang des Buches nicht zu sprengen. So tauchen Paketnamen und Importdeklarationen in der Regel nicht auf. Da der meiste Quellcode entweder als Snippet verlinkt oder Teil des Demoprojekts ist, wird der Name der Datei oder die URL zu dem Code-schnipselchen in der Listingunterschrift genannt, etwa so:

Listing 1 ABC.java

```
class ABC { }
```

Der abgebildete Quellcode befindet sich in der Datei *ABC.java*. Das Paket ist nicht angegeben, weil es in den Projekten keine zwei gleichen Typnamen in verschiedenen Paketen gibt. Für andere Ressourcen, wie XML- oder HTML-Dokumente, gilt das gleiche Muster:

Listing 2 pom.xml

```
<xml ...>
```

Der komplette Quellcode ist per Download verfügbar (siehe <https://rheinwerk-verlag.de/5980>).

Einige Listings sind nicht Teil des Date4u-Projekts und werden verlinkt, etwa so:

Listing 3 <https://gist.github.com/ullenboom/80cdb6c7435980c14b887a85d4b667ec>

```
# http://web.archive.org/web/20111224041840/http://ww...
```

Da sich oftmals ein Codeblock im Laufe der Zeit weiterentwickelt, macht die Ergänzung »Erweiterung« darauf aufmerksam; der neue Code ist in der Regel fett gesetzt:

Listing 4 Date4uApplicationTests, Erweiterung

```
class ABC {  
    public final static String INTRO = "Wann geht's endlich los?";  
}
```

Über die richtige Programmier-»Sprache«

Die Programmiersprache in diesem Buch ist Englisch, um ein Vorbild für »echte« Programme zu sein. Bezeichner wie Klassennamen, Methodennamen und auch eigene API-Dokumentationen sind hier auf Englisch, um eine Homogenität mit der englischsprachigen Java-Bibliothek und dem Spring Framework zu schaffen.

Download und Onlineinformationen

Die Downloaddateien zum Date4u-Projekt sowie Updates zum Buch finden sich auf der Webseite <https://rheinwerk-verlag.de/5980>. Alle Programmteile von Date4u sind frei von Rechten und können ungefragt in eigene Programme übernommen und modifiziert werden. Das ZIP-Archiv aus dem Download enthält folgende Bestandteile:

- ▶ **main:** Dies ist die Hauptanwendung Date4u mit Services zum Upload und Download von Bildern und zum Datenbankzugriff über Jakarta Persistence. Die Spring Shell ermöglicht die interaktive Anwendung mit Shell-Kommandos, etwa zur Auflistung der Profile. RESTful Webservices machen Profildaten von außen zugänglich. Das Projekt basiert auf Maven und lässt sich nahtlos in jede Java-IDE integrieren.

- ▶ **unicorns:** Das ist ein Verzeichnis mit 120 verschiedenen Bildern von Einhörnern. Die Dateinamen lauten *unicorn001.jpg* bis *unicorn120.jpg*. Ein ZIP-Archiv mit den Bildern liegt auch unter <https://github.com/ullenboom/120-unicorn-photos>.
- ▶ **h2-2.3.232-bin:** Dieses Verzeichnis enthält das *bin*-Verzeichnis des Datenbankmanagementsystems H2. Im *bin*-Verzeichnis befinden sich die Shell-Skripte *h2.bat* (für Windows) und *h2.sh* (für Unix/Linux). Mit diesen Skripten lässt sich die Datenbank direkt starten, vorausgesetzt, die JVM ist korrekt im Systempfad konfiguriert.
- ▶ **unicorn-database:** Dieses Verzeichnis enthält die Datei *unicorns.sql* mit einem SQL-Skript für das Datenbankmanagementsystem H2 zur Initialisierung der Beispieldatenbank. Sie finden es auch unter <https://tinyurl.com/4fu3hwu4>.
- ▶ **product-database:** Dies ist ein Spring-Boot-Programm mit einem Beispiel für den Zugriff auf die NoSQL-Datenbank MongoDB. Es ist ein Maven-Projekt.
- ▶ **chat-messages:** Dieses Spring-Boot-Programm beinhaltet ein Beispiel für Elasticsearch. Es ist ein Maven-Projekt.

Über den Autor

Christian Ullenboom tippte im Alter von 10 Jahren seine ersten Zeilen Code in den C64. Nach Jahren intensiver Assembler-Programmierung führte seine Reise nach dem Studium der Informatik und der Psychologie auf die Insel Java. Urlaubsreisen nach Python, JavaScript, TypeScript und Kotlin konnten ihn bisher nicht von der Inselbegabung befreien.

Seit über 25 Jahren ist Christian Ullenboom begeisterter Softwarearchitekt, Java-Trainer (<http://www.tutego.de>) und Ausbilder für Fachinformatiker. Aus seiner Schultätigkeit heraus sind mehrere Fachbücher entstanden, unter anderem *Java ist auch eine Insel. Einführung, Ausbildung, Praxis* (17. Auflage Rheinwerk 2024) und *Captain CiaoCiao erobert Java: Das Trainingsbuch für besseres Java*. Die Bücher bringen seit vielen Jahren Lesern die Programmiersprache Java näher. Für sein besonderes Engagement hat Sun (heute Oracle) Christian Ullenboom im Jahr 2005 als eine Persönlichkeit, die sich besonders um Java verdient gemacht hat, mit dem Status eines *Java-Champions* ausgezeichnet.

Der Autor dieses Buches teilt sein Wissen und Erfahrungen auch über Lernvideos unter <https://tutego.learnworlds.com>. Diese Videos behandeln auch eine Einführung in Java und Spring Boot. Gelegentlich postet er Developer-News und weiteren Unsinn auf der Mastodon-Instanz <https://mas.to/@ullenboom>.

Christian Ullenboom ist in Sonsbeck am Niederrhein verwurzelt.

Danksagungen

An dieser Stelle möchte ich mich bei allen Personen bedanken, die auf unterschiedliche Art und Weise zum Gelingen dieses Buches beigetragen haben. Besonderer Dank gilt meiner Frau für ihre Liebe und Geduld und meinen Eltern, ohne die es das Buch nicht geben würde.

Feedback

Auch wenn wir die Kapitel noch so sorgfältig durchgegangen sind, sind bei über 1000 Seiten einige Unstimmigkeiten wahrscheinlich, so wie auch jede Software rein statistisch Fehler aufweist.¹ Wer Anmerkungen, Hinweise, Korrekturen oder Fragen zu bestimmten Punkten oder zur allgemeinen Didaktik hat, sollte sich nicht scheuen, mir eine E-Mail unter der Adresse *ullenboom@gmail.com* zu senden. Ich bin für Anregung, Lob und Tadel stets empfänglich. Auch für Käse und Lakritz.

Vorwort zur 2. Auflage

In dieser neuen Auflage wurden viele kleinere Fehler der ersten Ausgabe behoben, und das Buch wurde auf den neuesten Stand von Spring Boot Version 3.4 und Spring Framework 6.2 gebracht.

Maven kann (und wird) in einigen Fällen Code generieren. Dieser Prozess wird vollständig über Annotationsprozessoren gesteuert, und die dafür notwendige Konfiguration des Maven-Compiler-Plugins wird früh im Buch erklärt, da es kapitelübergreifend ist.

Die KI-generierten Einhornbilder wurden komplett neu erstellt, da die Bildgenerierung innerhalb eines Jahres enorme Fortschritte gemacht hat. Die Qualität der Bilder ist nun deutlich verbessert, und die ZIP-Datei enthält auch Vorschaubilder (Thumbnails).

Im Grundlagenteil wurde die Behandlung optionaler Abhängigkeiten im Konstruktor verbessert sowie das *MessageSource*-Konzept aufgenommen. Das UML-Diagramm zum *TaskExecutor* wurde korrigiert und um virtuelle Threads ergänzt. Der Abschnitt »Am Anfang und Ende« hat sich von Kapitel 2 in das Kapitel 3 verschoben. Der Abschnitt »Validierung testen *« sowie die Details zum *QueryByExampleExecutor* wurden entfernt, da diese Themen sehr speziell sind und ich Raum für neue Inhalte schaffen wollte. Die Annotationen *@MockBean* und *@SpyBean* sind mittlerweile deprecated und

¹ Statistisch gibt es bei der Entwicklung etwa 1 bis 25 Fehler pro 1000 Zeilen Code. Für Referenzen siehe <https://stackoverflow.com/questions/2898571>.

wurden durch `@MockitoBean` und `@MockitoSpyBean` ersetzt. Dass Spring Boot auch Konfigurationen validieren kann, zeigt ein neues Beispiel und verzichtet dabei auf Lombok.

Obwohl Jakarta Persistence 3.2 erst mit Hibernate 7.0 vollständig umgesetzt wird und somit kein Bestandteil von Spring Boot 3.4 ist, so wurden doch schon die veralteten Datentypen (`Date`, `Calendar` usw.) aus dem Text genommen.

Der Datenbankabschnitt finden sich nun Informationen zu `[Fetchable]FluentQuery`. Die Neuerungen in Spring Data werden ebenfalls berücksichtigt: Die Scroll API findet Erwähnung, und Updates, die sich aus Spring Framework 6.1 ergeben, werden vorgestellt. Dazu gehört der `JdbcClient` mit diversen CRUD-Beispielen. Auch die `@ServiceConnection`-Annotation bei Testcontainers wird behandelt.

Im Webteil wurde das Beispiel für `@RequestParam` ausgetauscht und der `QuoteRestController` früher eingeführt. Neu hinzugekommen sind Erläuterungen zu `@JsonMixin` und zur Verwendung von Java-Records für REST-Rückgaben. Während POST und PUT sowie HTTP-Statuscodes bisher bei REST aufgeführt waren, ist dies nun getrennt, und das Kapitel über REST beschreibt nur noch die Besonderheiten von REST selbst; `@RequestBody` wird auch ohne REST benötigt. `PagedModel` wird als alternative Rückgabe zu `Page` empfohlen. Beim Testen von HTTP-Anfragen ist `MockMvc` durch `MockMvcTester` ersetzt worden.

Die Dokumentation der Klasse `RestClient` wurde deutlich ausgebaut, und der Spring-HTTP-Client wurde vom reaktiven `WebClient` auf den `RestClient` umgeschrieben. Hinzugekommen ist ein Abschnitt zur Verwendung von `QuerydslPredicate` in der Parameterliste einer Handler-Methode. Außerdem gibt es jetzt einen umfassenden Abschnitt zu Spring GraphQL, der die Implementierung sowohl auf der Server- als auch auf der Clientseite beleuchtet. Ein konkretes Beispiel zur Nutzung von `MapStruct` rundet die Neuerungen im Kapitel über Spring Web MVC ab.

Im Bereich Logging und Konfiguration wurde das JSON-Logging aufgenommen.

Ein ganz neues Kapitel stellt das spannende neue Modul Spring AI vor, mit dem Sprachmodelle angesprochen werden.

Und jetzt wünsche ich viele Frühlingsgefühle mit Spring Boot.