Geleitwort

15 Jahre Spring

Die Wurzeln von Spring liegen in einer Sammlung von Framework-Ideen im Buch »J2EE Design and Development« aus dem Jahre 2002 und deren Konkretisierung im Nachfolgebuch »J2EE without EJB« aus dem Jahre 2004. Bücher spielten immer schon eine wichtige Rolle für Spring, sowohl die Titel von meinem Mitgründer Rod Johnson und mir als auch jene aus der Community.

In den 15 Jahren seiner Geschichte entwickelte sich das Spring Framework konsequent weiter, wobei es im Kern des Frameworks nie zu einem radikalen Bruch kam, sondern immer zu einer Evolution im Rahmen des breiteren Java-Ökosystems. Natürlich musste sich das Framework einige Male im sprichwörtlichen Sinne neu erfinden, baute dabei aber jedes Mal wieder auf seinen Kernideen auf und erlaubte das schrittweise Upgrade von Bestandsanwendungen.

Einer der größten Meilensteine in diesen 15 Jahren war sicherlich die Einführung von Spring Boot im Jahre 2014. Ursprünglich gedacht als bequeme Konfigurationsvariante, entwickelte sich Boot schnell zum Standard-Einstiegspunkt ins Spring-Ökosystem und zum beliebtesten Framework für Microservice-Architekturen im Java-Umfeld. Dabei bietet Spring Boot nicht nur einen einfachen Einstieg für Entwickler, sondern auch viele Laufzeit-Features für den Betrieb.

Nach den Feature-Releases 1.1 bis 1.5 bringt Spring Boot 2.0 nun die erste große Überarbeitung, aufbauend auf Spring Framework 5.0. Die Schwerpunktthemen sind dabei breit gefächert: von reaktiver Programmierung bis hin zu Kotlin als neuer Programmiersprache auf der JVM, und nicht zu vergessen natürlich die neuen JDK-Releases der Generation 9+, die uns ab nun zwei Mal im Jahr bevorstehen. Spring Framework 5 und Spring Boot 2 stehen hierfür bereit.

Erfreulicherweise setzt sich auch die Tradition der begleitenden Bücher fort: Michael Simons liefert mit dem vorliegenden Titel nicht nur perfektes Timing, sondern vor allem einen umfassenden Einblick in viele aktuelle Bereiche des Spring-Ökosystems. Dieses Buch deckt alle Fa-

cetten der Entwicklung mit Spring Boot ab und bietet einen praktischen Leitfaden für moderne Java-Webanwendungen, der bereits jetzt auf die Herausforderungen der kommenden Jahre ausgerichtet ist.

Viel Spaß bei der Lektüre,

Jürgen Höller
Mitbegründer und Entwicklungsleiter des Spring Framework

Vorwort

Das Spring-Framework

Das Spring-Framework wurde 2002 erstmals als Idee vorgestellt und ein Jahr später unter dem Namen Spring-Framework als quelloffenes Projekt veröffentlicht. Das Ziel – damals wie heute – ist, die Entwicklung mit Java zu vereinfachen und gute Programmierpraktiken zu fördern.

Infrastruktur auf Anwendungsebene ist eines der Schlüsselelemente von Spring. Springs Fokus liegt ganz klar auf Bereitstellung und Konfiguration nichtfunktionaler Anforderungen, so dass Entwickler von Anwendungen sich auf ihre eigentliche Aufgabe, Implementierung der Geschäftslogik, konzentrieren können.

Kernfunktionen von Spring sind dabei:

- Dependency Injection
- MVC-basierte Webanwendungen und RESTful Webservices
- Grundlagen für JDBC, JPA und vieles mehr
- aspektorientierte Programmierung und deklarative Behandlung von Transaktionen

Der Umfang an Funktionalität ist seit 2003 kontinuierlich gewachsen, die Möglichkeiten, eine Spring-basierte Anwendung zu konfigurieren und in Betrieb zu nehmen, ebenso.

Spring Boot ist in diesem Kontext kein neues Framework, sondern eine Sicht auf die Spring-Platform, die es ermöglicht, eigenständige und produktionsreife Anwendungen auf Basis des beschriebenen Spring-Frameworks zu bauen, die unter anderem folgende Eigenschaften haben:

eigenständige Anwendungen, die keine externen Laufzeitabhängigkeiten mehr haben
eingebettete Container (zum Beispiel Tomcat, Jetty oder Undertow), so dass keine War-Dateien verteilt werden müssen
automatische Konfiguration soweit möglich

Bereitstellung von nicht funktionalen Eigenschaften, die zur Produktion in der Regel benötigt werden: Metriken, Health Checks und externe Konfiguration
keinerlei Generierung von Code oder Konfiguration

Warum Spring Boot?

Spring Boot basiert vollständig auf dem Spring-Framework. Es wurde mit dem Ziel erschaffen, die Entwicklung eigenständig lauffähiger Anwendungen mit dem Spring-Framework drastisch zu erleichtern. *Convention over configuration* sowie wenige, bekannte Annotationen reichen aus, um Artefakte zu erzeugen, die alle benötigten Abhängigkeiten mitbringen und extern konfigurierbar sind. Die Schwelle, mehr als eines dieser Artefakte zu erzeugen und zu verteilen, ist deutlich geringer und Grundlage erfolgreicher Verteilung von Microservices.

Schneller Start

Mit sogenannten »Startern« ist es möglich, die Konfiguration des Builds erheblich zu vereinfachen. Durch Deklaration einer einzigen Abhängigkeit wird zum Beispiel die Unterstützung einer Template-Sprache bereitgestellt, inklusive benötigter Abhängigkeiten und Konfiguration.

Testdriven

Spring Boot bietet herausragende Möglichkeiten, die technischen Schichten einer Anwendung, zum Beispiel Persistenz- und Webschicht, getrennt voneinander zu testen und die Ergebnisse dieser Tests anschließend zu Dokumentationszwecken zu nutzen. Ein nicht unerheblicher Gewinn, um sauberen Code zu schreiben.

Spring Boot liebt Microservices.

Darüber hinaus hat sich um Spring Boot, insbesondere durch die Firma Netflix, ein weiteres Ökosystem an Komponenten aufgetan, das insbesondere auf die Entwicklung verteilter, widerstandsfähiger (»resilient«) Microservices zielt. Machen Sie aber bitte nicht den Fehler und betrachten Spring Boot als »Microservice«-Framework. Sie können mit Spring Boot Teile von Anwendungssystemen auf Basis einer Microservice-Architektur bauen, aber Spring Boot ist deutlich mehr als ein »Microservice«-Framework.

Spring Boot und Cloud-native Java

Im Umfeld von Spring Boot hören Sie oft die Begriffe *Cloud-native* und *12-Factor-App*. Die Firma Pivotal erklärt den Begriff Cloud-native-Anwendungen als Anwendungen, die speziell mit dem Cloud-Modell im Hintergrund entworfen wurden und von kleinen Teams mit Fokus auf einzelnen Features entwickelt, deployt und betrieben werden. Skalierbare Infrastruktur, flexible Zielplattformen, Entwickler, die sich auf

Features anstelle von Boilerplate-Code konzentrieren können, und ein Betrieb, der sich am Geschäftsbedarf orientiert, sind heute noch Wettbewerbsvorteile, in naher Zukunft wohl eher Standard. Die folgenden Themen beschreiben Kernaspekte einer Cloud-native-Anwendung:

Entwicklung und Betrieb arbeiten Hand in Hand zusammen, so dass Builds, Tests und Releases oft und zuverlässig durchgeführt werden (»DevOps«),
Continuous Delivery, um einzelne Aspekte freigeben und veröffentlichen zu können, wenn sie fertig sind,
Microservices als architektonischer Ansatz, um Anwendungen als Sammlung kleiner Dienste zu strukturieren, die unabhängig voneinander verteilt, aktualisiert und skaliert werden können, und
Container als Laufzeitumgebung, die ähnlich schnell wie ein Microservice gestartet, skaliert und gestoppt werden können

Die 12-Factor-App als Methode beschreibt Anwendungen, mit denen die Vorteile einer Cloud-native-Umgebung ausgenutzt werden können, und wurde als Manifest unter 12factor.net von Entwicklern der Heroku-Plattform (https://www.heroku.com) in mehreren Sprachen veröffentlicht. Ihr Ziel war es, ideale Praktiken für die Entwicklung von Anwendungen zu schaffen, die wiederkehrende Probleme im Hinblick auf Konfiguration, Portierbarkeit, Verteilung und Skalierung von verteilten Anwendungen verhindern.

Viele Konzepte sind in die Gestaltung von Spring Boot eingeflossen und helfen dabei, Spring Boot zu einer idealen Plattform von Cloud-native-Anwendungen zu machen. Nach der Lektüre des Kapitels können Sie einige der Designentscheidungen Spring Boots besser nachvollziehen. Die Kenntnis der Idee der 12-Factor-Anwendung hilft Ihnen, einige Defaults von Spring Boot besser zu verstehen und sie – falls notwendig – bewusst anders zu konfigurieren.

Einige der Faktoren, die Sie ziemlich klar in Spring Boot wiederfinden, sind:

Explizite und vollständige Deklaration von Abhängigkeiten. Spring Boot macht dies zu großen Teilen über das Konzept sogenannter »Starter«, denen Sie im Folgenden sehr häufig begegnen werden.
Die externe Konfiguration einer Anwendung erfolgt nicht über Code, sondern der »reinen Lehre« nach nur über Umgebungsvariablen. Spring Boot geht einen Schritt weiter: Spring-Boot-Anwendungen können sehr leicht über Umgebungsvariablen konfiguriert werden, aber ähnlich effektiv auch durch Profile. Profile bündeln verschiedene, zusammengehörende Einstellungen für ein Ziel-Deployment (zum Beispiel `staging`, `test`, `prod` und andere). Die kon-

kreten Parameter der Profile sind dabei Teil des Artefakts, die Profile selber werden üblicherweise über die Umgebung aktiviert. Wichtig ist jedoch, dass es niemals notwendig sein sollte, eine Anwendung für ein einzelnes Ziel auf spezielle Art und Weise zu bauen. Die durch Spring Boot geschaffenen Möglichkeiten externer Konfiguration, die in Kapitel 4 besprochen werden, sind immens wichtig, um Anwendungen zu erstellen, die genau diesen Anspruch erfüllen. Gleiches gilt für intelligente »Magie«, die es in vielen Fällen erlaubt, komplett auf explizite Konfiguration zu verzichten und nur anhand der Umgebung zu entscheiden, welche Konfiguration sinnvoll ist oder nicht (Kapitel 5).

Unterstützende Dienste wie Datenbanken und dergleichen sollen als »angehängte« Ressourcen betrachtet werden: Der Code einer 12-Factor-Anwendung macht keinen Unterschied zwischen lokalen Diensten oder den Diensten Dritter. Diese Aspekte werden Sie in Kapitel 18 wiederfinden.

Zum Thema »Logging« wird folgende Aussage gemacht: Eine 12-Factor-Anwendung betrachtet Logs als Strom von Ereignissen. Das Ziel dieser Ereignisse ist irrelevant für eine 12-Factor-Anwendung. Sie schreibt ihre Logging-Ereignisse in der Regel nach `stdout`. In Kapitel 6 »Logging« wird ganz klar dieser Einfluss auf Spring Boot deutlich: Es gibt sehr einfache und schöne Möglichkeiten, das Format und die Detailtiefe (Level) der einzelnen Logs zu konfigurieren, das Ziel ist aber in der Regel die Standardausgabe.

Eine 12-Factor-App ist noch lange keine »perfekte« Anwendung, und die 12 Faktoren machen erst recht keine Aussage über ein Anwendungssystem, sind aber ein wesentlicher Bestandteil von Cloud-native-Anwendungen. Sie helfen, sowohl die Anwendung selber als auch die Entwicklung ohne wesentliche Änderungen im Tooling und der Architektur zu skalieren, maximale Portierbarkeit zwischen verschiedenen Ausführungsumgebungen zu erreichen und die Kluft zwischen Entwicklung und Produktion zu minimieren.

Warum dieses Buch?

Die Referenzdokumentation von Spring Boot ist ebenso wie die des Spring-Frameworks selber ausgesprochen gut, und es gibt eine Vielzahl von Einführungen für unterschiedliche Themen. Warum also noch ein Buch?

Dieses Buch soll interessierte Entwickler aus dem Java-EE-Bereich ebenso wie Spring-Entwickler ansprechen und ihnen ein »Rezept« an

die Hand geben, immer wiederkehrende Aufgaben aus dem fachlichen Alltag elegant und ohne Ablenkung mit Spring Boot zu nutzen.

Spring Boot bringt eine Menge automatischer Konfigurationen für fast alle Aspekte des Spring-Frameworks. Das Buch erklärt diese Magie und hilft dabei, sie für die eigenen Zwecke zu nutzen.

Warum legt das Spring-Boot-Team so viel Wert auf externe Konfiguration? Welchen Vorteil bietet ein Fat Jar? Wie funktionieren die Spring-Boot-Starter? Auch diese Fragen werden beantwortet.

Für wen ist dieses Buch?

Sie sollten grundlegende Kenntnisse der objektorientierten Programmierung besitzen und die Programmiersprache Java einschließlich der Sprachfeatures von Version 8 sowie die funktionalen Erweiterungen verstehen, um diesem Buch folgen zu können.

Spring Boot kann genutzt werden, ohne alle Module des Spring-Frameworks im Detail zu kennen oder vollständig zu verstehen. Ich habe versucht, für dieses Buch einen ähnlichen Kompromiss zu finden: Auch ohne Detailkenntnisse des Spring-Frameworks kennen Sie nach der Lektüre die Philosophie hinter Spring Boot und können damit Anwendungen erstellen. An vielen Stellen bin ich zusätzlich auf grundlegende Konzepte und Ideen des Spring-Frameworks eingegangen, insbesondere im Hinblick auf den Spring-Container, das Web-MVC-Framework und die Datenmodule.

Erfahrene Spring-Entwickler, die bisher noch nicht in Kontakt mit einer Spring-Boot-Anwendung gekommen sind, die unter modernen Gesichtspunkten entworfen wurde, werden überrascht sein, wie leichtfüßig die Entwicklung einer neuen Anwendung mittlerweile vonstatten geht.

Falls Sie bereits eine Vorstellung von Spring Boot und der grundlegenden Konfiguration haben, können Sie mit Kapitel 5 starten. Dort erfahren Sie, wie die automatische Konfiguration funktioniert. In Teil III erfahren Sie, wie Spring Boot bekannte Spring-Funktionen zur Verfügung stellt und wie diese Ihren Zwecken angepasst werden können.

Sie sollten die im Abschnitt »Werkzeuge« auf Seite 10 genannten Tools zur Hand haben, den Umgang mit Maven kennen und die IDE Ihrer Wahl beherrschen.