#22 TECHARTIKEL

Einführung in Spring Framework und Spring Boot

30.10.2025

Mateusz Klimentowicz

Inhalt

Einführung in Spring Framework und Spring Boot

Spring Framework und Spring Boot gehören zu den zentralen Werkzeugen der Java-Welt und unterstützen die Entwicklung moderner, skalierbarer und wartungsfreundlicher Anwendungen. Das Spring Framework bildet dabei die Grundlage mit einer robusten Architektur, während Spring Boot den Entwicklungsprozess erheblich vereinfacht und beschleunigt. In diesem Artikel werden die Kernfeatures beider Frameworks detailliert beschrieben, ihre Unterschiede erläutert und ihre Bedeutung für die Softwareentwicklung hervorgehoben.

1. Hintergrund zu Spring Framework und Spring Boot

Um die Bedeutung von Spring Framework und Spring Boot zu verstehen, lohnt sich ein Blick auf ihre Entstehung und Zielsetzung. Das Framework legt die Basis für eine modulare Architektur, während Spring Boot die Entwicklung durch Automatisierung und Vorkonfiguration erheblich vereinfacht.

1.1. Spring Framework

Seit seiner Einführung im Jahr 2004 hat sich das Spring Framework als eines der führenden Open-Source-Frameworks für die Java-Plattform etabliert. Es wurde entwickelt, um die Komplexität klassischer Java-Enterprise-Entwicklung zu reduzieren, indem es Softwareentwicklern ermöglicht, sich stärker auf die Geschäftslogik zu konzentrieren, anstatt sich intensiv mit technischen Details wie Objektverwaltung oder Ressourcenzugriff auseinandersetzen zu müssen. Mit Modulen für Webanwendungen, Datenbankintegration und viele weitere Bereiche ist es äußerst vielseitig einsetzbar.

1.2. Spring Boot

Spring Boot wurde 2014 als Erweiterung von Spring Framework veröffentlicht. Ziel war es, die Entwicklung von Spring-Anwendungen deutlich benutzerfreundlicher und effizienter zu gestalten. Viele Konfigurationsschritte, die im klassischen Spring Framework manuell erforderlich wären, werden automatisiert. Zudem stellt Spring Boot einsatzbereite Lösungen für typische Anwendungsfälle wie Microservices oder eigenständige Webanwendungen bereit. Dabei bleibt die volle Kompatibilität zum Spring Framework erhalten, dessen Kernfunktionen als Basis genutzt werden.

Nachdem die Grundlagen von Spring Framework und Spring Boot dargestellt wurden, folgt nun eine detaillierte Betrachtung der zentralen Funktionen von Spring Framework. Diese bilden die Basis für zahlreiche Einsatzszenarien und sind entscheidend für die Flexibilität und Leistungsfähigkeit des Frameworks.

2. Kernfeatures des Spring Frameworks

Die Kernfeatures des Spring Frameworks sind das Herzstück, das es zu einem so unverzichtbaren Werkzeug für Java-Entwickler macht. Sie bieten eine solide Grundlage für die Entwicklung flexibler, wartbarer und skalierbarer Anwendungen. Im Folgenden werden diese Features im Detail betrachtet.

2.1. Inversion of Control (IoC)

Inversion of Control ist ein grundlegendes Prinzip im Spring Framework, das die Art und Weise, wie Objekte erstellt und verwaltet werden, umkehrt. Statt dass eine Anwendung selbst ihre Komponenten instanziiert, übernimmt ein zentraler Container – der sogenannte Spring-Container – diese Aufgabe. Dies führt zu einer besseren Trennung der Anwendungslogik und macht den Code flexibler und einfacher zu testen. Besonders nützlich ist die sogenannte Dependency Injection (DI), eine spezielle Form von IoC: Hierbei werden Abhängigkeiten (z. B. andere Objekte oder Dienste, die eine Komponente benötigt) automatisch zur Verfügung gestellt, anstatt sie manuell zu erzeugen. Das Ergebnis ist ein sauberer, modularer Code, der weniger anfällig für Fehler ist.

2.2. Spring Lifecycle

Der Spring Lifecycle beschreibt den gesamten Lebenszyklus eines Beans (so werden Objekte genannt, die vom Spring-Container verwaltet werden). Dieser Zyklus umfasst mehrere Phasen: die Erstellung des Beans, seine Initialisierung, die Nutzung während der Laufzeit und schließlich seine Zerstörung, wenn dieser nicht mehr benötigt wird. Entwickler können in diesen Prozess eingreifen, indem sie spezielle Methoden definieren, die bei der Initialisierung oder beim Aufräumen ausgeführt werden. Das ist besonders praktisch, um Ressourcen wie Datenbankverbindungen effizient zu verwalten oder sicherzustellen, dass eine Anwendung ordnungsgemäß startet und herunterfährt. Der Lifecycle bietet somit Kontrolle und Transparenz über die Verwaltung von Komponenten.

2.3. Properties

Bei Properties handelt es sich um ein strukturelles Konzept, mit dem Konfigurationsdaten außerhalb des eigentlichen Codes definiert werden können. Das bedeutet, dass Werte wie Datenbank-URLs, Benutzernamen, Passwörter oder Serverports in separaten Dateien (z. B. application.properties) oder als Umgebungsvariablen gespeichert werden. Diese Externalisierung macht Anwendungen flexibler, da Änderungen an diesen Werten keine Anpassung des Codes erfordern – dies ist ein großer Vorteil bei der Wartung oder beim Einsatz in unterschiedlichen Umgebungen wie Entwicklung, Test oder Produktion. Zudem fördert diese die Sicherheit, da sensible Daten nicht direkt im Quellcode stehen.

2.4. Spring Expression Language (SpEL)

Die Spring Expression Language, kurz SpEL, ist eine mächtige Funktion, die es Entwicklern erlaubt, dynamische Ausdrücke direkt in der Konfiguration oder im Anwendungskontext zu verwenden. Mit SpEL können Werte zur Laufzeit berechnet, auf Eigenschaften von Objekten zugegriffen oder sogar einfache logische Bedingungen definiert werden – und das alles ohne den Code selbst zu ändern. Beispielsweis könnte eine Konfiguration so gestaltet werden, dass sie abhängig von einer bestimmten Bedingung unterschiedliche Werte annimmt. Diese Flexibilität ist besonders hilfreich bei komplexeren Anwendungen, bei denen Anpassungen häufig erforderlich sind, ohne die gesamte Anwendung neu kompilieren zu müssen.

2.5. Profiles und Conditional

Mit Profiles und Conditional bietet das Spring Framework Werkzeuge, um Anwendungen an verschiedene Umgebungen oder Szenarien anzupassen. Profiles ermöglichen es, unterschiedliche Konfigurationssätze zu definieren – etwa eines für die Entwicklung mit einer lokalen Datenbank und eines für die Produktion mit einem entfernten Server. Diese Profile können dann je nach Bedarf aktiviert werden. Conditional geht noch einen Schritt weiter: Hier können Entwickler festlegen, dass bestimmte Komponenten nur dann erstellt werden, wenn bestimmte Bedingungen erfüllt sind, z. B. das Vorhandensein einer bestimmten Bibliothek. Zusammen machen diese Features Anwendungen anpassungsfähiger und einfacher zu verwalten.

Nach der Betrachtung der Kernfeatures des Spring Frameworks folgt nun eine Übersicht der zentralen Funktionen von Spring Boot. Aufbauend auf dem Fundament des Frameworks erweitert Spring Boot dessen Möglichkeiten und reduziert den Konfigurationsaufwand durch zahlreiche integrierte Mechanismen.

3. Kernfeatures von Spring Boot

Spring Boot bringt eine Reihe von innovativen Features mit, die es zu einem unverzichtbaren Werkzeug für moderne Java-Entwicklung machen. Diese Features zielen darauf ab, die Entwicklung zu beschleunigen, die Konfiguration zu vereinfachen und die Bereitstellung zu erleichtern. Im Folgenden werden diese Features im Detail betrachtet.

3.1. Autokonfiguration

Die Autokonfiguration ist eines der herausragendsten Merkmale von Spring Boot und hebt es deutlich vom reinen Spring Framework ab. Die Konfiguration analysiert die Abhängigkeiten einer Anwendung (z. B. welche Bibliotheken im Projekt enthalten sind) und richtet automatisch sinnvolle Standardeinstellungen ein. Wenn beispielsweise eine Datenbankbibliothek erkannt wird, konfiguriert Spring Boot automatisch eine Datenquelle, ohne dass der Entwickler dies manuell tun muss. Diese intelligenten Voreinstellungen können bei Bedarf angepasst werden, sparen aber enorm viel Zeit und reduzieren Fehlerquellen, insbesondere bei der Entwicklung neuer Projekte.

3.2. Starters

Spring Boot Starters sind vorgefertigte Pakete von Abhängigkeiten, die alle notwendigen Bibliotheken für bestimmte Anwendungsfälle bündeln. Es gibt Starter für Webanwendungen, Datenbankzugriff, Messaging und vieles mehr. Anstatt jede Bibliothek einzeln auszuwählen und deren Kompatibilität zu prüfen, fügt ein Entwickler einfach den passenden Starter hinzu – und schon sind alle benötigten Komponenten in der richtigen Version verfügbar. Das vereinfacht den Einstieg erheblich, da es die Konfigurationsarbeit minimiert und sicherstellt, dass alles reibungslos zusammenarbeitet.

3.3. Eingebettete Server

Mit eingebetteten Servern revolutioniert Spring Boot die Bereitstellung von Webanwendungen. Statt einen externen Server wie Tomcat oder Jetty separat zu installieren und zu konfigurieren, wird dieser direkt in die Anwendung integriert. Das Ergebnis ist eine eigenständige Anwendung, die einfach als JAR-Datei ausgeführt werden kann – ideal für Microservices oder Container-Umgebungen wie Docker. Diese Funktion erhöht die Portabilität und macht die Bereitstellung schneller und weniger fehleranfällig.

3.4. Fat Jars

Fat Jars (auch als „übergewichtige JARs“ bezeichnet) sind ausführbare Dateien, die die gesamte Anwendung – inklusive aller Abhängigkeiten, Konfigurationen und Ressourcen – in einem einzigen Paket enthalten. Anstatt eine Anwendung mit separaten Bibliotheken und Konfigurationsdateien zu verteilen, kann ein Entwickler einfach diese eine Datei bereitstellen und ausführen. Das vereinfacht die Verteilung und ist besonders nützlich in Cloud-Umgebungen, wo eine unkomplizierte Bereitstellung entscheidend ist.

3.5. Native Images

Native Images sind eine neuere Entwicklung in Spring Boot, die es ermöglichen, Anwendungen direkt in Maschinencode zu kompilieren, anstatt sie auf der Java Virtual Machine (JVM) auszuführen. Das Ergebnis sind Anwendungen, die extrem schnell starten und weniger Speicher benötigen – ein großer Vorteil in cloud-nativen oder serverlosen Szenarien. Diese Technologie, unterstützt durch Tools wie GraalVM, macht Spring Boot noch attraktiver für moderne, ressourceneffiziente Anwendungen.

3.6. Auto-Registration von Spring Beans

Spring Boot erleichtert die Arbeit mit Beans, indem es viele davon automatisch registriert. Anstatt jede Komponente manuell im Spring-Container zu definieren, erkennt Spring Boot Klassen mit bestimmten Markierungen (wie @Component oder @Service) und fügt sie automatisch hinzu. Dieser Prozess, auch als Component Scanning bekannt, reduziert den Aufwand für die Konfiguration

erheblich und macht die Entwicklung schneller, da Entwickler sich auf die eigentliche Logik konzentrieren können statt auf Verwaltungsaufgaben.

3.7. DevTools

Die Spring Boot DevTools sind eine Sammlung von Werkzeugen, die den Entwicklungsprozess beschleunigen. Dazu gehört etwa die Funktion des Hot Reloads: Änderungen am Code werden sofort in der laufenden Anwendung übernommen, ohne dass ein kompletter Neustart nötig ist. Ebenso werden bei Änderungen automatisch Neustarts der Anwendung durchgeführt, was den Feedback-Zyklus verkürzt. Diese Features sind besonders für Entwickler nützlich, die viel experimentieren oder schnell Prototypen erstellen möchten.

Im Anschluss an die Darstellung der Kernfeatures von Spring Boot wird nun die Beziehung zwischen Spring Framework und Spring Boot betrachtet und in das abschließende Fazit integriert.

4. Fazit

Das Spring Framework bildet das solide Fundament mit Funktionen wie IoC, Lifecycle-Management und Profiles, die eine flexible und robuste Entwicklung ermöglichen. Spring Boot nimmt diese Basis und ergänzt sie mit Automatisierung und Komfortfunktionen wie Autokonfiguration, Starters und eingebetteten Servern. Während das Spring Framework für Projekte geeignet ist, die maximale Kontrolle über die Konfiguration erfordern, ist Spring Boot die erste Wahl für schnelle Entwicklung und moderne Architekturen wie Microservices.

Zusammen bieten sie eine unschlagbare Kombination: Das Spring Framework sorgt für Stabilität und Vielseitigkeit, während Spring Boot die Hürden der Einrichtung und Wartung senkt. Beide sind unverzichtbar für die moderne Java-Entwicklung und ermöglichen es Entwicklern, Anwendungen zu schaffen, die den Anforderungen der heutigen Zeit gerecht werden.