Spring Boot

Java Enterprise Apps

Tomasz Lubowiecki

Spring Boot

- Ist ein Java-Web-Framework mit sehr vielen Features

Webframework

„[…] ist eine Software, die für die Entwicklung von dynamischen Webseiten, Webanwendungen oder Webservices ausgelegt ist. Sich wiederholende Tätigkeiten werden vereinfacht und die Wiederverwendung von Code und die Selbstdokumentation der Software- Entwicklung gefördert. [...] Durch vordefinierte und vorgefertigte Klassen werden häufig gebrauchte Funktionen wie Mailversand, sichere Authentifizierung und Authentisierung, Sicherheitsfunktionen, Lokalisierung, Performance (z. B. HTTP Caching) oder grundlegende Funktionen für Webformulare vom Framework mitgebracht.“

Wikimedia

Spring Boot

- Java-Code für Entwicklung von Webanwendungen
- Wiederholende Tätigkeiten werden vereinfacht
- Wiederverwertbarkeit von Code wird erhöht
- Namens-Konventionen helfen bei Selbstdokumentation
- Vordefinierte Klassen für
  - Authentifizierung
  - Performance
  - Lokalisierung
  - ...

Spring Framework

- Von Haus aus viele Features
- Schwergewichtige Anwendung
- Hoher Konfigurations-Aufwand

Spring Boot

- Konzentriert sich auf Features für Anwendung
- Leichtgewichtige Anwendung
- Wenig bis keine Konfiguration (Convention over configuration)

Spring Boot

Nötige Vorkenntnisse

- Java
- Maven / Gradle
- HTTP
- Datenbanken

Spring Boot

Einbindung / Erzeugung

- Erzeugung eines vorkonfigurierten Projekts: https://start.spring.io/
- Maven Parent:
  - Group ID: org.springframework.boot
  - Artifact ID: spring-boot-starter-parent
- Maven Dependency:
  - Group ID: org.springframework.boot
  - Artifact ID: spring-boot-starter-web

Spring Boot
Konfiguration

- @SpringBootApplication - Markiert die Main-Klasse
- Aufruf SpringApplication.run(Main.class, args) in der Main-Methode

Spring Boot
Annotations

- @EnableAutoConfiguration
  - Automatischer Konfigurationsmechanismus
- @ComponentScan
  - Scannen nach Komponenten
- @Configuration
  - Markiert eine Konfigurations-Klasse
- @SpringBootApplication
  - Fasst die obigen 3 Annotation mit ihren Standardeinstellungen zusammen

Spring Boot

Beans

- Eine Bean ist eine von Spring verwaltete Klasse
- Beans implementieren Interfaces (Inversion of Control)
- Automatische Bereitstellung mit @Autowired Annotation

Spring Boot

Beans

- Beans sind mit Annotationen gekennzeichnet z.B.
  - @Component
  - @Service
  - @Repository
  - @Controller

Spring Boot
Komponenten

- @Component markiert eine Komponente
- Spring Boot scannt die Anwendung nach Komponenten
  - @ComponentScan

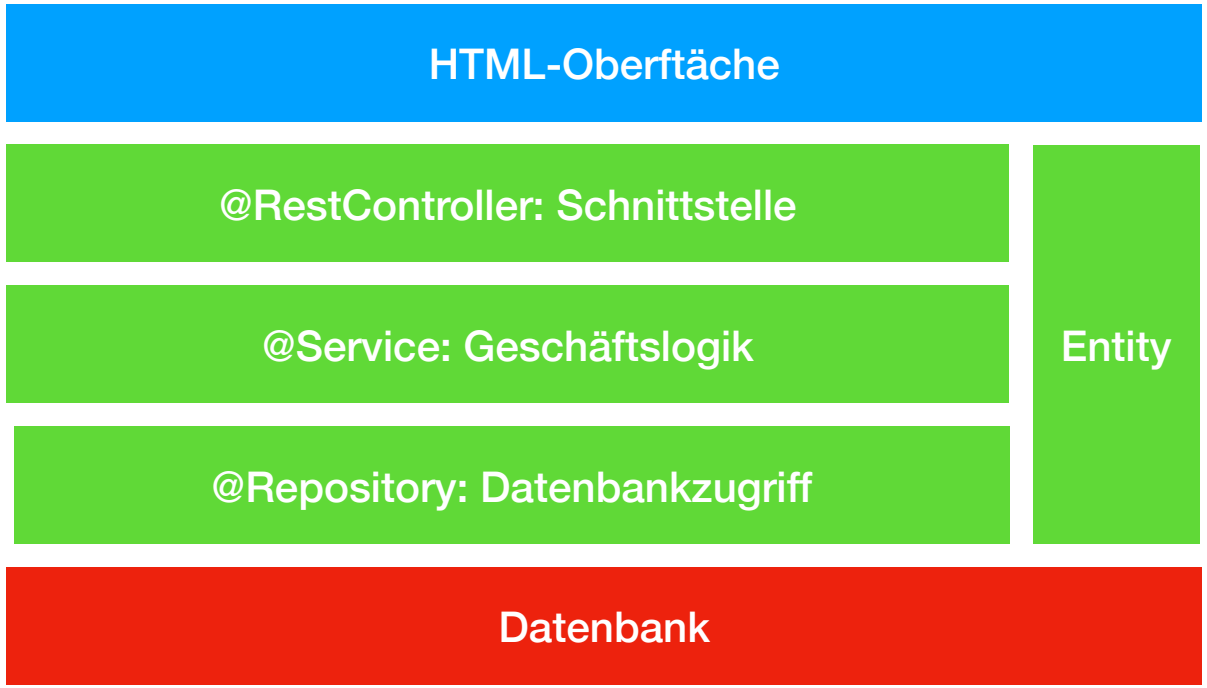
Spring Boot

Komponenten

- @Component
  - Allgemeine Komponente
- @Repository
  - Datenbank-Zugriffs-Klasse
- @Service
  - Geschäftslogik-Klasse
- @Controller
  - Kommunikation mit Außenwelt

Spring Boot

Schichtenmodell (Beispiel)



Spring Boot
Persistenzierung

- @Repository
- Klasse der Datenbank-Zugriffs-Schicht
- Maven Dependency:
  - Group ID: org.springframework.boot
  - Artifact ID: spring-boot-starter-data-jpa
- Implementierung durch Ableitung von Repository, CrudRepository etc.
- CRUD = Create Read Update Delete

Spring Boot
Geschäftslogik

- @Service
- Geschäftslogik
- Kern der Anwendung
- Wird vom Entwickler implementiert

Spring Boot
Schnittstelle

- @RestController ist ein spezieller @Controller
- Kommunikation mit der Außenwelt
- Mögliche Annotationen an Klassen und Methoden:
  - @RequestMapping, @GetMapping, @PostMapping
- Empfangen von HTTP-Anfragen auf definierte URLs
- Empfangen von Parametern (@RequestParam) und Objekten (@RequestBody)
- Rückantwort (z.B. als JSON-Objekte)