Spring Framework

Eine Einführung für Einsteiger

Lam, Luzak, Pleschinger

19. Jänner 2024

Inhalt

Spring Framework

Ein **Java Framework** ist eine vorgefertigte Struktur oder Plattform, die die Entwicklung von Java-Anwendungen erleichtert[1]

- Beispiele: Hibernate, Struts, ...

Das **Spring Framework**

- ist ein Open-Source-Java-Framework entwickelt von Rod Johnson in 2003[2]
- ermöglicht Entwicklungsteams durch Bereitstellung eines Modells sich auf die Businesslogik zu konzentrieren

[1] Morales-Zamora, Paredes-Xochihua, and Sanchez-Juarez. "Importance of the Spring Framework in web programming". In: *Journal of Computational Systems and ICTs* (2022).

[2] Johnson et al. *The spring framework-reference documentation*. interface, 2004.

Relevanz

- Spring Framework 1.0 (März 2004)
  - *Inversion of Control*, aspektorientierte Programmierung und JDBC-Abstraktion, ...
- Spring Framework 2.0 (Oktober 2006)
  - Spring MVC-Webmodul, verbesserte XML-Konfiguration und Unterstützung für Java 5, ...
- Spring Framework 3.0 (Dezember 2009)
  - *Spring Expression Language* (SpEL), Java-basierte Konfiguration, REST-Unterstützung und Unterstützung für Java 5, 6 und 7, ...
- Spring Framework 4.0 (Dezember 2013)
  - Unterstützung für Java 8, ...
- Spring Framework 5.0 (September 2017)
  - Unterstützung für Java 9, Java EE 8, der Sprache Kotlin, ...
- Spring Framework 6.0 (November 2022)[3]
  - Java 17+ Baseline, ...

[3] *Spring Framework Versions*. Accessed: 2024-01-16. URL: https://github.com/spring-projects/spring-framework/wiki/Spring-Framework-Versions.

Warum Spring?

- Modularität
- Flexibilität[4]
- Interface-orientierte Programmierung
- *Dependency Injection*
- Aspektorientierte Programmierung[5]
- ...

[4] *Why Spring*. Accessed: 2024-01-16. URL: https://spring.io/why-spring.

[5] Morales-Zamora, Paredes-Xochihua, and Sanchez-Juarez. "Importance of the Spring Framework in web programming". In: *Journal of Computational Systems and ICTs* (2022).

Überblick der Spring Architektur

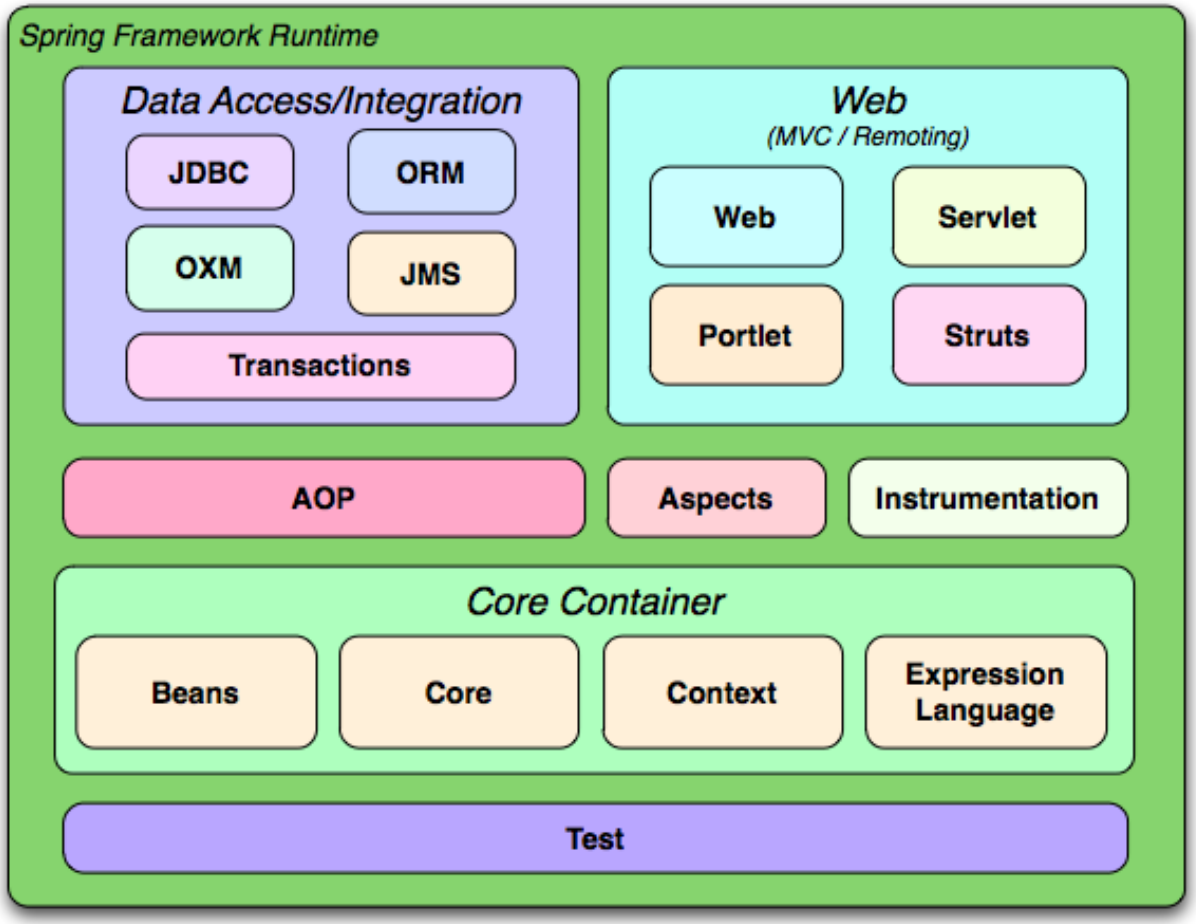


Figure: Spring Architektur[6]

[6] Johnson et al. *The spring framework-reference documentation*. interface, 2004.

Core Container

- Core-Modul[7]
  - stellt die grundlegenden Teile des Frameworks bereit
- Beans-Modul
  - befasst sich speziell mit der Verwaltung von *Beans*
- Context-Modul
  - Interface ApplicationContext dient zur Verwaltung von *Beans* und anderen Komponenten in Spring
- ...

[7] Mane, Chitnis, and Ojha. "The Spring Framework: An Open Source Java Platform for Developing Robust Java Applications". In: *International Journal of Innovative Technology and Exploring Engineering* (2013).

Data Access/Integration und Web Layer

Data Access/Integration Layer:[8]

- erleichtert Zugriff und Bearbeitung von Daten
  - Java Database Connectivity (JDBC)-Modul
  - ...

Web Layer:

- bietet diverse Funktionen für die Webentwicklung mit Spring

[8] Mane, Chitnis, and Ojha. "The Spring Framework: An Open Source Java Platform for Developing Robust Java Applications". In: *International Journal of Innovative Technology and Exploring Engineering* (2013).

Sonstiges

- AOP-Modul[9]
  - bietet eine Implementierung für aspektorientierte Programmierung
- Testmodul
  - unterstützt das Testen von Spring-Komponenten mit JUnit, etc.
- ...

[9] Mane, Chitnis, and Ojha. "The Spring Framework: An Open Source Java Platform for Developing Robust Java Applications". In: *International Journal of Innovative Technology and Exploring Engineering* (2013).

Überblick der Spring Architektur

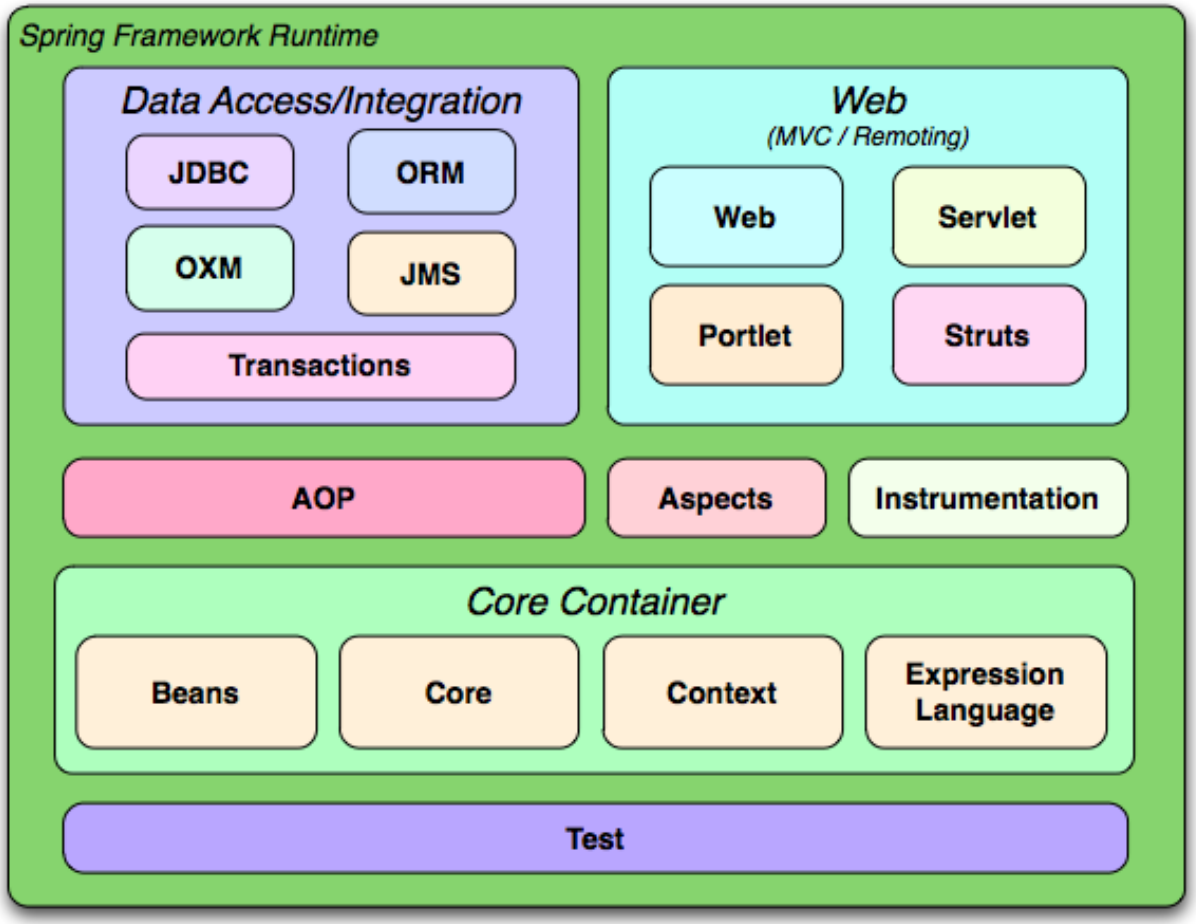


Figure: Spring Architektur[6]

[6] Johnson et al. *The spring framework-reference documentation*. interface, 2004.

Designkonzepte

- Verschiedenste umgesetzte Designkonzepte
- Präsentation zwei exemplarischer Konzepte:
  - Dependency Injection (DI)
  - Inversion of Control (IoC)

Service Objects

- Einzelne Objekte stellen Funktionalität / Dienste für andere Objekte / Funktionen bereit
- Beispiel:
  - Logger: Verarbeiten von Log-Einträgen
  - ConfigStorage: Speichern der Programmkonfiguration
  - Mailer: Senden von Emails
- Typische OOP-Architektur[10]

[10] Dhananjay Prasanna. *Dependency injection: design patterns using spring and guice*. Simon and Schuster, 2009, p. 2.

Object Dependencies

- Beispielszenario: kritische Fehlermeldungen sollen per Mail an den Admin gesendet werden
- Email-Adresse in Programmkonfiguration hinterlegt
- Logger benötigt Instanzen von ConfigStorage und Mailer!
- Mögliche Implementierungen:[11]
  - Logger erstellt Instanzen von benötigten Klassen selbst
  - "Factory"-Design Pattern
  - "Service Locator"-Design Pattern
  - ...

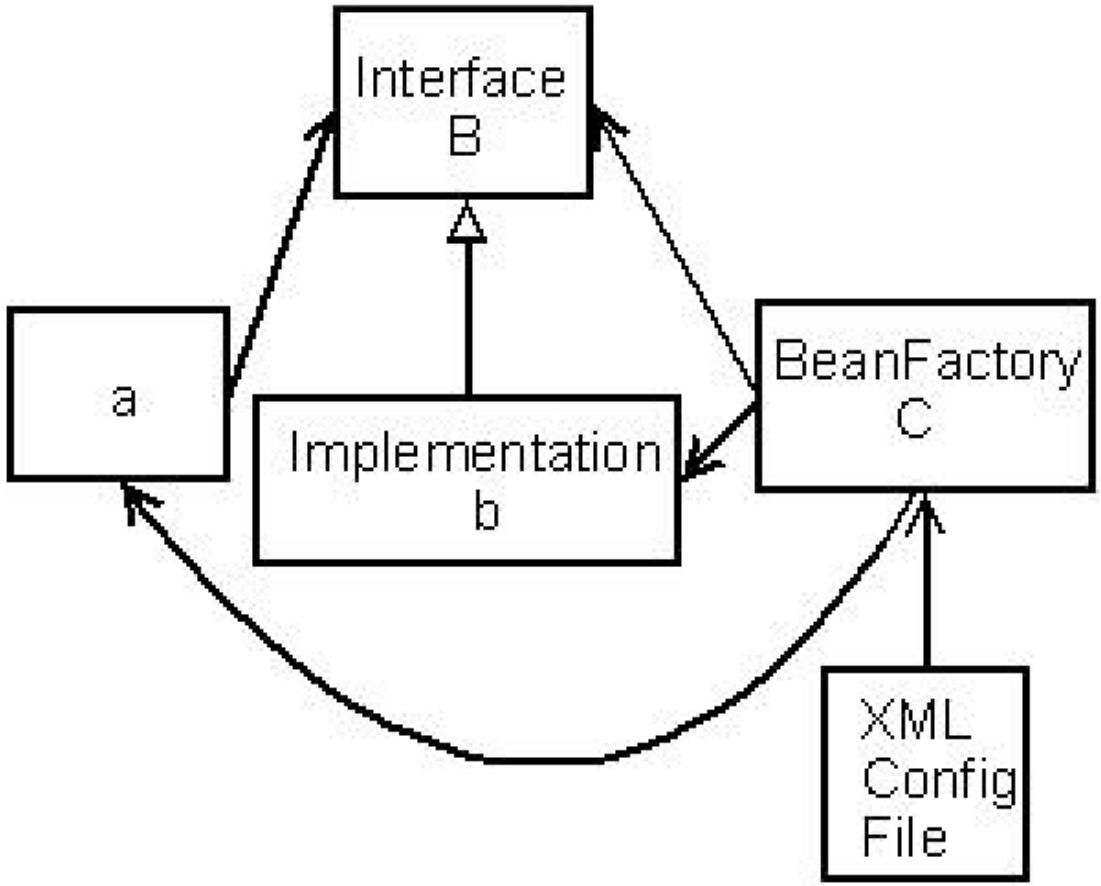
[11] Dhananjay Prasanna. *Dependency injection: design patterns using spring and guice*. Simon and Schuster, 2009, pp. 5–12.

Dependency Injection

- "Hollywood Principle": "Don't call us; we'll call you!"
- Dependencies werden in Benutzerobjekte "injiziert"
  - durch z.B. Konstruktor, Setter-Methoden, etc.
- DI-Framework besitzt Liste an (Service-)Objekten, welche automatisch in neue Objekte injiziert werden
- Verwendung von Interfaces, um benötigte Funktionalität von spezialisierten Implementierungen zu separieren
  - Beispiel: ILogger-Interface mit MailLogger / ConsoleLogger / anderen Implementationen
  - Austausch der verwendeten Implementationen ohne / mit geringen Code-Änderungen möglich!
  - Vereinfacht z.B. Tests
- Entkoppeln von Klassen führt zu erhöhter Fehlerresistenz und Flexibilität der Codebase[12]

[12] Ekaterina Razina and David S Janzen. "Effects of dependency injection on maintainability". In: *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications: Cambridge, MA*. 2007, p. 7, p. 9.

Dependency Injection - Spring Framework



Quelle: Ekaterina Razina and David S Janzen. "Effects of dependency injection on maintainability". In: *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications: Cambridge, MA*. 2007, p. 7

Dependency Injection - Beispielcode

```
public class MailLogger implements ILogger {
    private final IConfigStorage config;
    private final IMailer mailer;

    // Dependencies are automatically injected here
    public MailLogger(IConfigStorage config, IMailer mailer) {
        this.config = config;
        this.mailer = mailer;
    }

    @Override
    public void log(LogLevel level, string message) {
        //...
    }
}
```

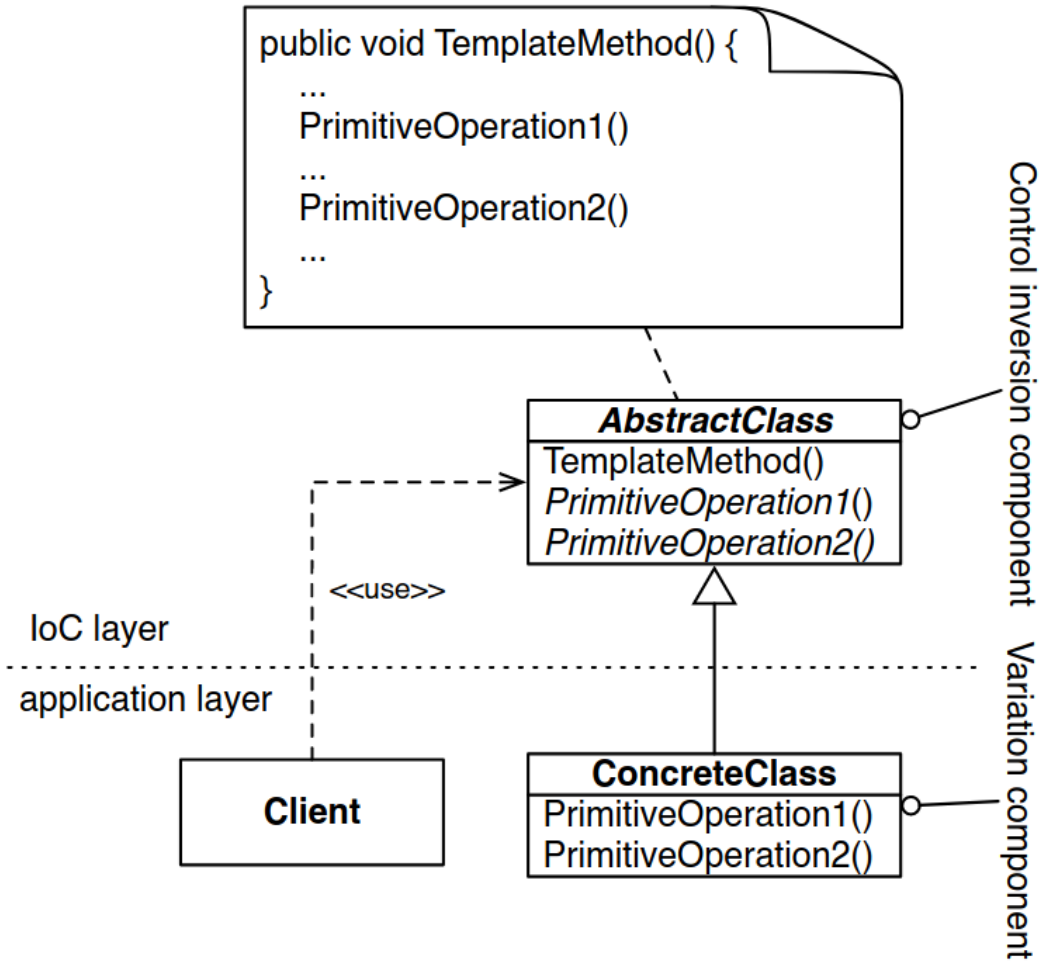
Basierend auf: Spring Framework Documentation. *Dependency Injection*. URL: https://docs.spring.io/spring-framework/reference/core/beans/dependencies/factory-collaborators.html (visited on 01/19/2024)

Inversion of Control

- Klassisches Framework-Paradigma: Hilfsfunktionen werden bereitgestellt und aufgerufen
  - Beispiel: SMTP-Bibliothek wird aufgerufen, um Email zu versenden
  - Applikation ist für Control Flow / Data Flow / etc. verantwortlich
- Inversion of Control: Umkehr jener Beziehung
  - Framework ruft Applikationscode wenn benötigt selbständig auf
- Delegation spezifischer Teilaufgaben an Applikation, Composite-Behavior von Framework definiert
- Anwendungsfälle: GUI-Frameworks, Event-Driven IO, Component-Oriented Programming, etc.[13]

[13] Stefan Sobernig and Uwe Zdun. "Inversion-of-control layer". In: *Proceedings of the 15th European Conference on Pattern Languages of Programs*. 2010, pp. 1–22.

Inversion of Control - Architektur



Quelle: Stefan Sobernig and Uwe Zdun. "Inversion-of-control layer". In: *Proceedings of the 15th European Conference on Pattern Languages of Programs*. 2010, pp. 1–22

Inversion of Control - Dependency Injection

- Dependency Injection ist ein spezifisches Beispiel einer Inversion of Control[14]
  - Häufige Verwendung des Begriffes im Kontext von DI (z.B. Spring's "IoC-Container"[15])
  - Ein GUI-Callback ist genauso IoC wie DI!
- "dependency inversion": Framework ist für die Verwaltung von Objektabhängigkeiten zuständig, nicht Applikation[16]
- "Hollywood Principle" generelles Mantra von IoC, nicht nur DI

[14] Dhananjay Prasanna. *Dependency injection: design patterns using spring and guice*. Simon and Schuster, 2009, p. 15.

[15] *The IoC Container*. URL: https://docs.spring.io/spring-framework/reference/core/beans.html (visited on 01/19/2024).

[16] Stefan Sobernig and Uwe Zdun. "Inversion-of-control layer". In: *Proceedings of the 15th European Conference on Pattern Languages of Programs*. 2010, pp. 1–22.

Beispielszenario

- Web-Application für Brettspiele
- Für den Anfang sollen nur Spielräume erstellt werden können
- Was wir dafür brauchen:
  - Maven Projekt
  - Spring Boot Server
  - Controller: RESTful-API
    - Interaktion mit Server über HTTP Requests (GET, POST, PUT, DELETE)
  - Service: Daten verarbeiten und weiterleiten
  - Repository: Daten speichern
  - Unit Test
    - Eine Klasse isoliert testen
  - Integration Test
    - Das Zusammenspiel der einzelnen Komponenten im Gesamtkontext testen

Main Klasse

- @SpringBootApplication: Autokonfiguriert den Kontext, scannt Projekt nach Beans und fügt diese dem Kontext hinzu[17]
- In main Methode muss nur noch SpringApplication.run(class, args) aufgerufen werde

```
@SpringBootApplication
public class BackendApplication {
    Mikolaj Luzak
    public static void main(String[] args) {
        SpringApplication.run(BackendApplication.class, args);
    }
}
```

[17] *Annotation Interface SpringBootApplication*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/SpringBootApplication.html (visited on 01/16/2024).

Controller Annotations

- Klassenebene:
  - @RestController:[18]
    - @Controller: Fügt dem Spring Kontext eine Bean der Klasse hinzu
    - @ResponseBody: Returnwerte aller Methoden werden in ein Http Response Body verpackt
  - @RequestMapping: Basispfad, über den Controller Requests abfängt[19]
- Methodenebene;
  - @PostMapping: POST Request mit gegebener URL-Endung wird von annotierter Methode abgefangen[20]
- Parameterebene;
  - @PathVariable: Injiziert Variable aus dem Pfad der Anfrage[21]

[18] *Annotation Interface RestController*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html (visited on 01/16/2024).

[19] *Annotation Interface RequestMapping*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html (visited on 01/16/2024).

[20] *Annotation Interface PostMapping*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/PostMapping.html (visited on 01/16/2024).

[21] *Annotation Interface PostMapping*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/PostMapping.html (visited on 01/16/2024).

Controller

- Commandline command zum lokalen Aufruf von createRoom mit username "user1":
- curl -X POST http://localhost:8080/api/rooms/user1

```java
@RestController
@RequestMapping("/api/rooms")
public class RoomController {
    2 usages
    private final RoomService roomService;
    Stevan Nedic
    public RoomController(RoomService roomService) { this.roomService = roomService; }
    Mikolaj Luzak +1 *
    @PostMapping("/{username}")
    public ResponseEntity<Room> createRoom(@PathVariable String username) {
        Room room = roomService.createRoom(username);
        return new ResponseEntity<>(room, HttpStatus.CREATED);
    }
}
```

Service

- @Service: Fügt dem Spring Kontext eine Bean der Klasse hinzu[22]

```java
@Service
public class RoomService {
    2 usages
    private final RoomRepository roomRepository;
    3 usages
    private final IdGenerator idGenerator;
    Mikolaj Luzak *
    public RoomService(RoomRepository roomRepository) {
        this.roomRepository = roomRepository;
        idGenerator = new SimpleIdGenerator();
    }
    2 usages   Mikolaj Luzak *
    public Room createRoom(String username) {
        Room room = new Room(idGenerator.generateId());
        room.setRoomAdmin(new Player(idGenerator.generateId(), username));
        roomRepository.put(room.getId(), room);
        return room;
    }
}
```

[22] *Annotation Interface Service*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Service.html (visited on 01/16/2024).

Repository

- @Repository: Fügt dem Spring Kontext eine Bean der Klasse hinzu (Analog zu @Controller, @Service und @Component)[23]

```
@Repository
public class RoomRepository extends HashMap<UUID, Room> {}
```

[23] *Annotation Interface Repository*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Repository.html (visited on 01/16/2024).

Unit Test Annotations

- Klassenebene:
  - @WebMvcTest(Class): Deaktiviert Autokonfiguration, außer MockMvc und Controller Beans[24]
- Variablenebene:
  - @Autowired: Injiziert Bean der Klasse der annotierten Variable aus dem Kontext[25]
  - @MockBean: Fügt dem Kontext eine Dummy-Bean hinzu[26]

[24] *Annotation Interface WebMvcTest*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/autoconfigure/web/servlet/WebMvcTest.html (visited on 01/16/2024).

[25] *Annotation Interface Autowired*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html (visited on 01/16/2024).

[26] *Annotation Interface MockBean*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/mock/mockito/MockBean.html (visited on 01/16/2024).

Unit Test

```java
@WebMvcTest(RoomController.class)
public class RoomControllerTest {
    @Autowired
    private MockMvc mockMvc;
    @MockBean
    RoomService roomService;
    Mikolaj Luzak *
    @Test
    void createRoomTest() throws Exception {
        Mockito.when(roomService.createRoom(anyString())).thenAnswer(invocation -> {
            Room room = new Room(UUID.randomUUID());        //Hier wird der Username ausgelesen
            room.setRoomAdmin(new Player(UUID.randomUUID(), invocation.getArgument( i: 0)));
            return room;
        });
        mockMvc.perform(post( urlTemplate: "/api/rooms/user1"))
                .andExpect(status().isCreated())
                .andExpect(jsonPath( expression: "$.players.key.name").value( expectedValue: "user1"));
    }
}
```

Integration Test Annotations

- Klassenebene:
  - @SpringBootTest: Autokonfiguriert gesamten Applikationskontext für Tests[27]
  - @AutoConfigureMockMvc: Fügt dem Kontext autokonfigurierte MockMvc Bean[28]
- Variablenebene:
  - @Autowired

[27] *Annotation Interface SpringBootTest*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/context/SpringBootTest.html (visited on 01/16/2024).

[28] *Annotation Interface AutoConfigureMockMvc*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/autoconfigure/web/servlet/AutoConfigureMockMvc.html.

Integration Test

```java
@SpringBootTest
@AutoConfigureMockMvc
public class RoomIntegrationTest {
    @Autowired
    MockMvc mockMvc;
    Mikolaj Luzak *
    @Test
    void createRoomTest() throws Exception {
        mockMvc.perform(post( urlTemplate: "/api/rooms/user1"))
                .andExpect(status().isCreated())
                .andExpect(jsonPath( expression: "$.id").isNotEmpty())
                .andExpect(jsonPath( expression: "$.players.key.id").isNotEmpty())
                .andExpect(jsonPath( expression: "$.players.key.name").value( expectedValue: "user1"));
    }
}
```

Bibliographie I

[1] *Annotation Interface AutoConfigureMockMvc*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/autoconfigure/web/servlet/AutoConfigureMockMvc.html.

[2] *Annotation Interface Autowired*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html (visited on 01/16/2024).

[3] *Annotation Interface MockBean*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/mock/mockito/MockBean.html (visited on 01/16/2024).

[4] *Annotation Interface PostMapping*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/PostMapping.html (visited on 01/16/2024).

[5] *Annotation Interface Repository*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Repository.html (visited on 01/16/2024).

[6] *Annotation Interface RequestMapping*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html (visited on 01/16/2024).

[7] *Annotation Interface RestController*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html (visited on 01/16/2024).

[8] *Annotation Interface Service*. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/stereotype/Service.html (visited on 01/16/2024).

[9] *Annotation Interface SpringBootApplication*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/SpringBootApplication.html (visited on 01/16/2024).

[10] *Annotation Interface SpringBootTest*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/context/SpringBootTest.html (visited on 01/16/2024).

Bibliographie II

[11] *Annotation Interface WebMvcTest*. URL: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/autoconfigure/web/servlet/WebMvcTest.html (visited on 01/16/2024).

[12] Spring Framework Documentation. *Dependency Injection*. URL: https://docs.spring.io/spring-framework/reference/core/beans/dependencies/factory-collaborators.html (visited on 01/19/2024).

[13] Johnson et al. *The spring framework-reference documentation*. interface, 2004.

[14] Mane, Chitnis, and Ojha. "The Spring Framework: An Open Source Java Platform for Developing Robust Java Applications". In: *International Journal of Innovative Technology and Exploring Engineering* (2013).

[15] Morales-Zamora, Paredes-Xochihua, and Sanchez-Juarez. "Importance of the Spring Framework in web programming". In: *Journal of Computational Systems and ICTs* (2022).

[16] Dhananjay Prasanna. *Dependency injection: design patterns using spring and guice*. Simon and Schuster, 2009.

[17] Ekaterina Razina and David S Janzen. "Effects of dependency injection on maintainability". In: *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications: Cambridge, MA*. 2007, p. 7.

[18] Stefan Sobernig and Uwe Zdun. "Inversion-of-control layer". In: *Proceedings of the 15th European Conference on Pattern Languages of Programs*. 2010, pp. 1–22.

[19] *Spring Framework Versions*. Accessed: 2024-01-16. URL: https://github.com/spring-projects/spring-framework/wiki/Spring-Framework-Versions.

[20] *The IoC Container*. URL: https://docs.spring.io/spring-framework/reference/core/beans.html (visited on 01/19/2024).

[21] *Why Spring*. Accessed: 2024-01-16. URL: https://spring.io/why-spring.

Fragerunde

Danke für eure Aufmerksamkeit!

Gibt es noch Fragen?