SPENGERGASSE

Vergleich des Java EE Standard-Frameworks im Vergleich zu Spring zur Entwicklung von Business Applications

Paul MAUROVICH

Version 0.5 - 2021-04-01

Inhaltsverzeichnis

Chapter 1. Abstract

In diesem Paper werden die zwei (Web) Frameworks, Java EE und Spring, auf unterschiedliche Kriterien, wie unter anderem Regelmäßigkeit von Updates, langfristiger Support, Kosten sowie Eleganz beim Programmieren, Funktionen und Querschnittsfunktion wie Logging, Wartbarkeit sowie die Developer Codebase untersucht. Die aus den Kriterien resultierenden Vor- und Nachteilen werden gegenübergestellt und bewertet.

Um die Arbeit angemessen zu verstehen, werden fortgeschrittene Fähigkeiten und Erfahrung in Informatik bzw. Programmiersprachen vorausgesetzt. Das Paper dient als Basis einer Entscheidungsgrundlage des zu wählenden Frameworks für die Entwicklung von Business Applikationen.

Chapter 2. Einleitung

Java Entwickler und Java Entwicklerinnen stehen, sei es im Projekt für einen Kunden oder privat, für die Entwicklung von Business Applikationen, vor der Entscheidung, welches (Web) Framework sie für ihr Projekt wählen sollten. Sei es Hibernate, Spring, Apache Wicket, Grails oder Java EE bzw. Jakarta EE, für solch ein Projekt stehen viele Frameworks zur Verfügung. So macht es die Entscheidung schwer, welches Framework am besten für das Projekt passt. Hierfür wird in diesem Paper auf die zwei Frameworks Spring und Java EE, speziell auf die Entwicklung von Business Applikationen und Web Lösungen, eingegangen.

Zu den jeweiligen Kategorien wird eine Bewertung in Form von Punkten vergeben, welchen den positiven Nutzungsgrad widerspiegeln. Dafür werden Punkte von einer SKale von 0 bis maximal 5 vergeben, wobei 0 für "Nicht ausreichend bzw. nicht vorhanden" und 5 für "Sehr gut" steht. Die Punktezahl wird am Ende jeder Kategorie definiert und am Ende ein finales resümierendes Fazit abgegeben.

Wichtig zu beachten ist, dass es kein "das beste Framework" für die Entwicklung **spezieller** Lösungen gibt, denn es kommt ganz darauf an, was die Business-Lösung können und welche Rahmenbedingungen wie Kosten oder Entwicklungsstandards es erfüllen soll.

Chapter 3. Java EE Framework

Java Enterprise Edition, kurz Java EE, früher auch J2EE genannt, ist ein Framework, ausgelegt für mittel bis große Projekte, welches es ermöglicht, umfassende und meist komplizierte (Unternehmens)Anwendungen zu entwickeln. Vor allem für Webanwendungslösungen im Unternehmensbereich, meist in Kombination mit REST Schnittstellen und verteiltes Computing kommt Java EE häufig zum Einsatz. Das Framework baut auf Java SE (Standard Edition) auf und wird beispielsweise mit einer Microservice- oder Monolithen Architektur entwickelt. Das Framework wurde seit Java EE 8 auf Jakarta EE 8 umbenannt und Augenmerk auf Cloud Migration gelegt. Java EE baut auf der Verwendung von der objektorientierten Programmiersprache Java.

Benutzt wird es hauptsächlich zur Entwicklung von browserbasierten Web- und Enterprise (Business) Applikationen und bietet zahlreiche Spezifikationen wie

- **Servlets – WebServlets**: Handhabung von Web Requests und Responses
- **Java API**: RESTful Webservices
- **JPA und JMS**: Datenbank Verbindungen und Datenbank Transaktionen
- **Bean Validation**: Deklariert eindeutige Constraints über eine Methode, ein Feld oder einer Klasse einer JavaBeans-Komponente
- **Uvm.**

Eine grundlegende Java EE Architektur sieht wie folgt aus:

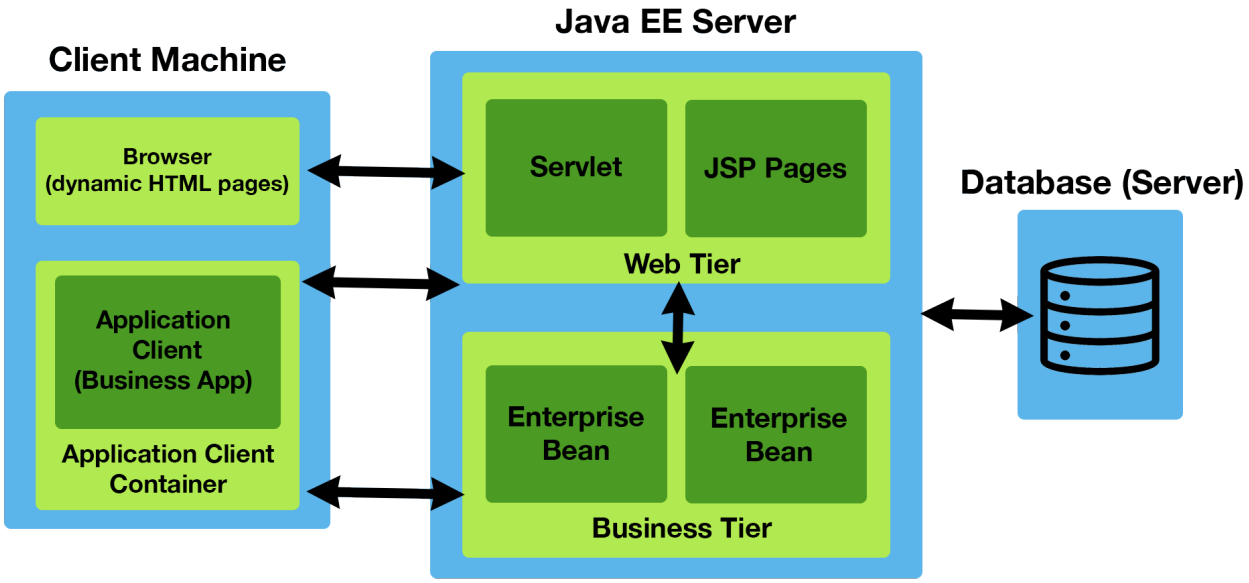


Figure 1. Java EE Architektur Überblick

Es besteht aus drei Hauptkomponenten. Die "Client Machine" ist das Endgerät: Browser und/oder die Business-Applikation. Diese hat Verbindungen zum zweiten Teil, dem Java EE Server, welcher über ein Web Container, mit dem Web Servlet sowie den JSP Pages, diese sind "Jakarta Server Pages" (JSP), welches es ermöglichen, Java Code in HTML- oder XML-Seiten einzubetten, und dem Business Container, wo sich die Enterprise Beans befinden. Der Server ist mit einer Datenbank bzw. einem Datenbank-Server verbunden, worüber Datenbanktransaktionen (CRUD) durchgeführt werden können. Transaktionen können mit zum Beispiel einer REST-API oder via SOAP-API durchgeführt werden.

Chapter 4. Spring Framework

Das Spring Framework ist eine Art Erweiterung, von der Java EE Platform ausgehend, für die Entwicklung von Java Applikationen im Hinblick auf Web Services und REST Schnittstellen. Vor allem die zwei Teile von Spring, Spring Boot und Spring MVC, werden dafür häufig eingesetzt. Hauptaugenmerk liegt auf die Entwicklung von Applikationen Business Logik, ohne viel Konfiguration von der Umgebung sowie Spezifikationen.

Spring (Boot) kommt mit einem automatisch konfigurierten Anwendungskontext sowie einem Web Server, wie Tomcat, um den Einstieg zu erleichtern und aufwendiges Aufsetzen des Frameworks zu unterbinden. Wohingegen Java EE nur mit der Programmiersprache Java läuft, so benötigt Spring keine spezifische Sprache, doch meist wird Java, Groovy oder Kotlin verwendet.

Der Schwerpunkt von Spring (Boot) liegt auf der Entwicklung von und mit Microservices. Es ist größtenteils dafür ausgelegt und verfügt, wie auch Java EE, über einige Module. Diese sind zum Beispiel:

- **Datenbank Zugang, Daten Integrität**: JDBC, JMS, ORM, OXM
- **Web**: Servlet, WebSocket, Portlet
- **Core Container**: Beans, Core, Context und SpEL
- **Testing**: JUnit, Mockito etc.

In diesem Beispiel wird sich auf die Spring Komponente Spring Boot fokussiert. Die Architektur dessen sieht wie folgt aus:

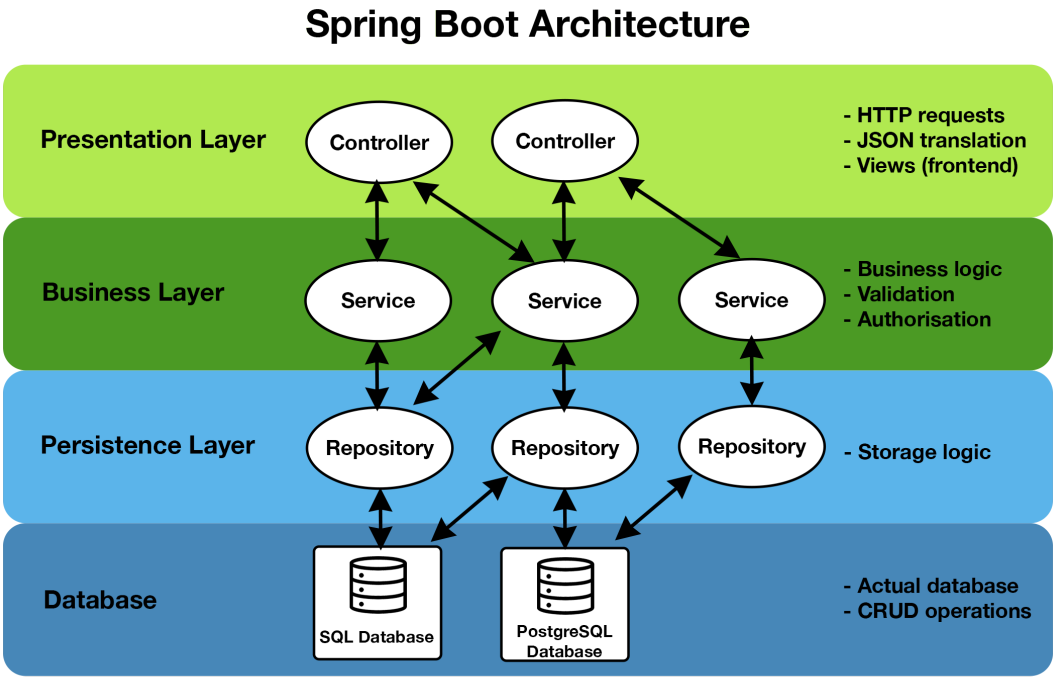


Figure 2. Spring Boot Framework 4- Schichten Architektur

Die Spring Boot Framework Architektur besteht aus vier Teilen. Die "Presentation Layer" Ebene, welche sich zum Beispiel um eingehende und ausgehende HTTP Requests/Responses kümmert. Sie übersetzt die Daten von und zu JSON Objekten, um den datenmäßigen Umgang mit der Applikationen zu gewährleisten. Grundsätzlich ist sie mit einer Frontend HTML oder XML Seite verbunden.

Die "Business Logic" Ebene beinhaltet die Services, welche mit Business Logik ausgestattet sind. Die eingehenden Daten werden in die Ebene transferiert und dort validiert bzw. autorisiert. Um Daten abzugleichen und welche Speicher Lösungen gewählt wurden, um das geht es bei der "Persistence Layer" Ebene. Sie verfügt über mehrere Repositorien, welche Datenbankzugriffe festlegen bzw. meist ein geeignetes SQL-Skript im Hintergrund ausführen.

Schlussendlich werden die Daten zum Beispiel in SQL, H2 oder PostgreSQL Datenbanken gespeichert und dort zugänglich gemacht. Die "Persistence Layer" Ebene führt mit der "Database Layer" Ebene "CRUD" (Create, Read, Update und Delete) Operationen durch, welche das Hinzufügen, Lesen, Aktualisieren und Löschen von Daten bereitstellt.

Chapter 5. Erste Schritte

Die API Beispiele werden mittels IntelliJ IDE realisiert.

Java EE:

1. In IntelliJ ein neues Projekt erstellen und "Java Enterprise" wählen. Danach bei "Projekt Template" "Rest service" wählen und "GlassFish 5.0.0" als "Application server" auswählen.
2. Danach Dependencies überprüfen und Namen festlegen.
3. Nach der Erstellung des Projektes wurden von IntelliJ Demo Klassen angelegt.
4. Dependencies werden in der `pom.xml` Datei verwaltet:

```
<dependencies>
    <dependency>
        <groupId>javax.ws.rs</groupId>
        <artifactId>javax.ws.rs-api</artifactId>
        <version>2.1.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>${junit.version}</version>
        <scope>test</scope>
    </dependency>
    <!-- ... -->
</dependencies>
```

1. Es wird ein grundlegender API Controller unter `JavaEETest.java` angelegt:

```
@Path("/hallo-java-ee") 
public class JavaEETest {
    @GET 
    @Produces("text/plain") 
    public String hello() {
        return "Willkommen zu Java EE!"; 
    }
}
```

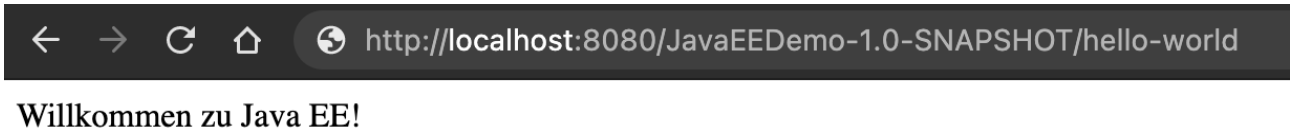
 Name der Basis Route

 Festlegung der Operation (GET, POST, ...)

 Definierung des Rückgabeformats. In dem Fall plain Text.

 Rückgabe eines String, um die Funktion zu testen.

Nachdem das Projekt gestartet wure, ist unter "http://localhost:8080/JavaEEDemo-1.0-SNAPSHOT/hello-world" die sehr grundlegende REST-API abrufbar:



Willkommen zu Java EE!

Figure 3. Java EE REST API GET-Response

Spring:

In diesem Beispiel wird sich auf eine Komponente von dem Spring Framework konzentriert: Spring Boot, in Kombination mit Java und dem Build Tool "Maven".

Unter https://start.spring.io/ ist es möglich, ein fertiges Spring Boot Projekt Template anlegen zu lassen.

1. Die Webseite besuchen und wichtige Informationen wie das Build Tool, die Programmiersprache, die Versionen und geeignete Namen vergeben.
2. Anschließend mit Klick auf den Knopf "ADD DEPENDENCIES..." die gewünschten Dependencies wie "Spring Web", "Spring Data JPA", "H2 Database" und "Thymeleaf" mit erneutem Klick hinzufügen.

3. Letztlich auf den Knopf "GENERATE" klicken und ein ZIP-Ordner mit dem vorkonfiguriertem Projekt wird automatisch heruntergeladen und ist, sofern es in eine IDE wie IntelliJ geladen wird, einsatzbereit.

Die Dependencies werden im späteren `pom.xml`, in dem die Dependencies verwaltet werden, wie folgt angezeigt:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
    </dependency>
    <!-- ... -->
</dependencies>
```

Anschließend das Projekt öffnen und beispielsweise einen Controller `BootTestController.java` anlegen:

```
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("api/v1/test")
public class BootTestController {

    @GetMapping("/hello")
    public String hello() {
        return "Willkommen zu Spring Boot!";
    }
}
```

- Name der Basis Route
- Name der spezifischen Route, welcher der Basis Route ergänzt wird, der Methode
- Rückgabe eines String, um die Funktion zu testen

Darüber hinaus hat der "Spring Initializr" auch eine Main Klasse erzeugt, welche nach nötigen Gebrauch mit zum Beispiel `@Bean` Annotations ergänzt werden kann.

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

Die Applikation kann entweder via spezifischen Startknopf der gewählten IDE oder mit dem Befehl `./mvnw spring-boot:run` gestartet werden.

Für den Gebrauch von Maven Kommandos muss dieses erst auf der CLI installiert werden.

Nun ist unter "http://localhost:8080/api/v1/test/hello" die sehr grundlegende REST-API abrufbar:

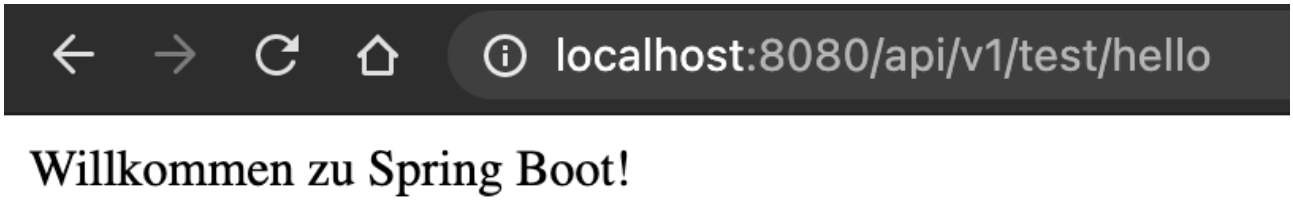


Figure 4. Spring Boot REST API GET-Response

Chapter 6. Regelmäßige Updates

Java EE, seit Java EE 8 umbenannt auf Jakarta EE, erhält regelmäßig neue Versionsupdates. Die Frequentierung der Stable Updates ist durchschnittlich alle zwei bis vier Jahre und enthält meist neue Features und Verbesserungen. Das letzte Hauptupdate (Jakarta EE 9) fand im Jahr 2020 statt. Das Spring Framework wird kontinuierlich aktualisiert und dessen letztes großes Update war im Jahr 2017 mit der Version Spring 5.0 und als letzter Stable Release gilt Spring 5.3.4, welches im Februar 2021 veröffentlicht wurde. Zwar erhalten beide Frameworks regelmäßig Updates, wohingegen das Spring Framework öfters Updates bekommt als Java EE. Dies ist wohl auch der stetig expandierenden Nutzerbasis von Spring geschuldet. Auch Dependencies werden seitens beider Frameworks im Laufe von Updates mit aktualisiert.

Durch Dependency Injection und Cloud Migration beider Frameworks, bei Spring ist dies beispielsweise Spring Boot, ist eine gute Wartbarkeit gegeben. Spring Boot verfügt über Plain Old Java Objects (POJO), welche sich durch kleine und "leichtgewichtige Klassen" auszeichnet, ermöglicht eine präzise Wartbarkeit, da jegliche Logik in kleinen Klassen leicht erreichbar und nicht zu umfassend verschachtelt ist. Auch Java EE verfügt über Dependencies, welche einfach aktualisiert werden können. Bei der Wartbarkeit beider Frameworks kommt es vor allem darauf an, ob eine Monolithen- oder Microservice Architektur gewählt wurde. Letzteres bietet einen weit ausgehend mehr wartbaren Code, da der Code je nach Spezifikationen in verschiedene Module aufgesplittet ist. Hunderte Klassen in einem Package, unübersichtliche Klassennamen und hunderte Codezeilen in Klasse sind hauptverantwortlich für schlechte Wartbarkeit.

Durch immer neue Updates und deren neuen Funktionen und Verbesserungen wird die Wartbarkeit immer besser, sei es mit der Cloud Migration von Java EE 8 oder Spring Boot, beide wurden auf den heutigen Stand der Technik gehoben und erreichen somit die volle Punktezahl von 5 Punkten.

Fazit:

	Java EE	**Spring**
Punkte	5	5

Table 1. Punkte Resümee "Regelmäßige Updates"

Chapter 7. Langfristiger Support

Wie bereits im vorhergehenden Kapitel erwähnt, erhalten beide Frameworks stetig Updates. Jedoch wirkt seit 2017 Oracle, der damalige Leiter von der Java Enterprise Platform, nicht mehr an der Entwicklung von Java EE mit, da sie die Leitung dafür aus mangelnder Interesse einer Weiterentwicklung abgegeben haben. Dies zeigt auf, dass Java EE immer weniger Relevanz in der heutigen Software Gemeinschaft hat und einen langfristigen Support fragwürdig macht. Zwar wird mit einigen Updates in der Zukunft rechnen, vor allem notwendige Security Updates, doch bahnbrechende Updates werden auf sich warten lassen.

Die Website "JRebel" beispielsweise, hat einige Entwickler befragt, ob und wenn sie von Java EE zu Spring migriert hätten bzw. es tun möchten. Der Report ergab, dass lediglich 14 Prozent von Spring zu Java EE, wohingegen 36 Prozent eher von Java EE zu Spring migriert haben bzw. es tun möchten. Deswegen wird Java EE resümierendes für diese Kategorie 3 von Punkten erhalten.

Viele Portale sprechen von dem "Tod von Java EE", nachdem Oracle die Leitung dafür abgegeben hat. "Negotiations Failed: How Oracle killed Java EE", so schreibt es beispielsweise der Autor Markus Krag in seinem Blog. In dem Bericht geht hervor, dass es einen Markenstreit zwischen Oracle und der Eclipse Foundation gab, welcher in keiner Einigung resultierte und Java EE einiges an Relevanz kostete.

Seitens des Spring Frameworks, vor allem die zwei Komponenten Spring Boot und Spring MVC, gibt es keine Anzeichen eines nahestehenden Endes des Supports. Unter der Leitung der Apache Foundation gewinnt das Framework immer mehr und mehr an Interesse und Nutzung unter der Entwicklergemeinschaft. Nicht nur sind große skalierbare Projekt mit dem Framework möglich, auch regelmäßige stabile Versionen kommen auf den Markt. Diesen Fakten geschuldet, erhält das Spring Framework in dieser Kategorie die volle Punktezahl.

Fazit:

	Java EE	Spring
Punkte	3	5

Table 2. Punkte Resümee "Langfristiger Support"

Chapter 8. Kosten, Eleganz beim Programmieren

Seit Jakarta ist dieses Framework komplett Open Source, weswegen Oracle verfügt über die Markenrechte von "Java EE", weswegen die neue Leitung, die Eclipse Foundation, es auf "Jakarta EE" umbenannt hat. Dadurch ist Jakarta EE größtenteils kostenlos zu nutzen, jedoch gibt es neben den frei zugänglichen Java EE Servern wie "Tomcat" oder "Glassfish", auch kostenpflichtige Server. Java EE bietet folgende Paradigmen:

- Cloud und PaaS: Cloud Migration (Web), durch Java EE 8, und PaaS (Platform as a service)
- Aspect oriented programming (AOP)
- Design Paradigmen POJO: Unterstützung von POJO (Plain Old Java Object)
- Java EE unterstützt die Reactive Programmierung

Das Spring Framework unterliegt der Apache-Lizenz, welche eine Free-Software-Lizenz ist. Das Framework ist somit unentgeltlich und auch Open-Source. Auch Spring hat sowohl kostenlose als auch kostenpflichtige Module und Server, welche aber grundsätzlich nicht nötig sind. Das Framework verfügt unter anderem über folgende Prinzipien:

- **Lightweight**: Spring ist einfach aufgebaut und benötigt nicht viel Speicherplatz, beispielsweise ist die Basis Version nur 1 Megabyte groß.
- **Inversion of control (IOC)**: Entwickler müssen Komponenten wie Libraries nicht selbst erstellen/anlegen, sondern lediglich durch Dependency Injection diese in einer Konfigurationsdatei bestimmen. Spring IOC hat dann die Aufgabe, alle Dependencies lauffähig zu vereinen.
- **Aspect oriented programming (AOP)**: Spring unterstützt auch die aspektorientierte Programmierung. Wartbarkeit und Modularität wird durch die Trennung von logischen Aspekten und der hauptsächlichen Business Logik. AOP

trennt diese zwei Komponenten, was bei der einfachen objektorientierten Programmierung schwer möglich ist.

- Container: Spring unterteilt Code in Container und handhabt Lebenszyklen und Anwendungskonfigurationen.
- Spring unterstützt auch eine Reactive Programmierung mit der Dependency "Reactor", vor allem in Kombination mit einer Microservice Architektur.

Chapter 9. Funktionsumfang

Chapter 10. Developer Codebase und Community Größe

Seitens beider Frameworks gibt es eine mittel bis große Developer-Base. Java EE bzw. Jakarta EE wird laut der Webseite "https://stackshare.io/", welche unterschiedliche Frameworks, Programmiersprachen etc. bewertet und aufzeigt, welche Technologien heutzutage verwendet werden, von 29 Unternehmen genutzt. Darunter "TripAdvisior", "Biting Bit" und "IWB". Außerdem hat Java EE auch Integrationen in "Eclipse", "NetBeans IDE" sowie "Apache Wicket".

Auch Spring ist auf derselben Webseite vertreten. Angaben zufolge benutzen 501 Unternehmen Spring in ihrem Stack, darunter "Accenture", "Zalando" und auch "deleokorea".

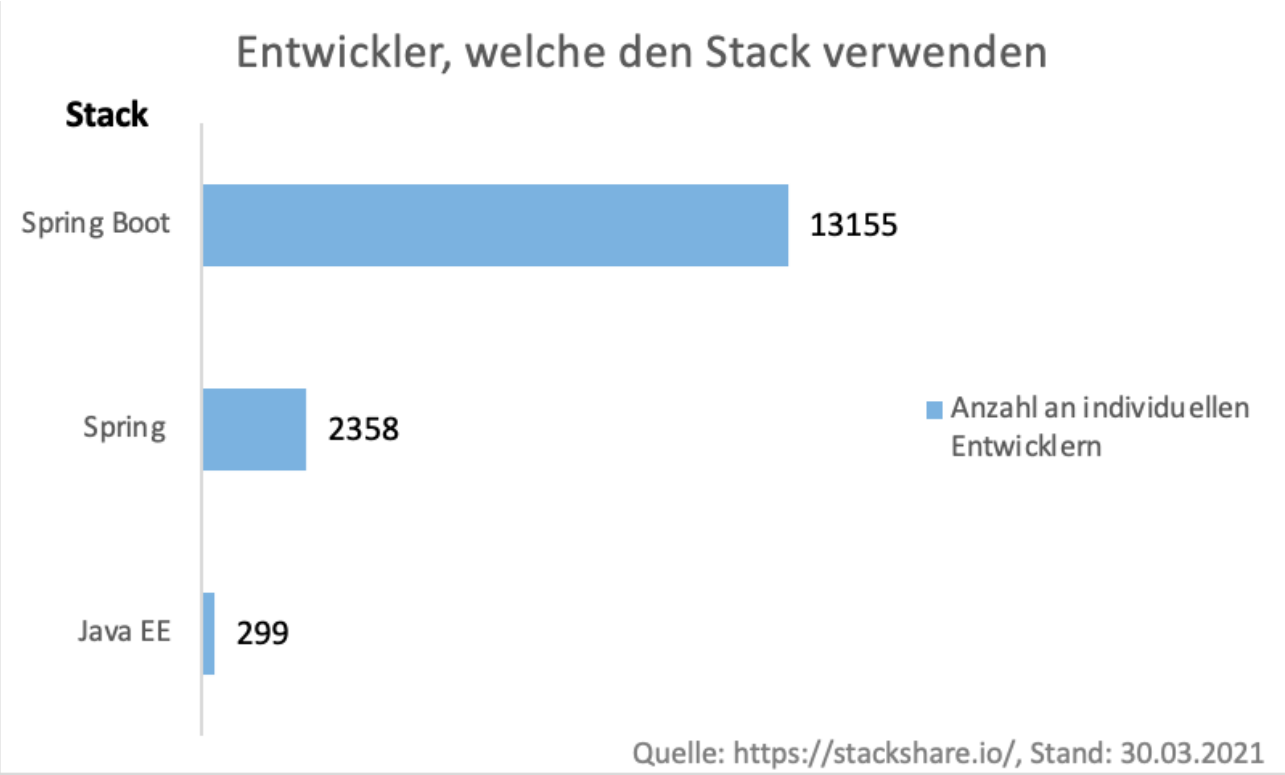


Figure 5. Entwickler, welche den Stack verwenden

Rund 13155 Entwickler haben bekannt gegeben, dass sie die Spring Komponente Spring Boot in ihrem Stack benutzten, bei Spring sind es rund 2358 Entwickler und Java EE mit nur wenigen 299 Entwicklern.

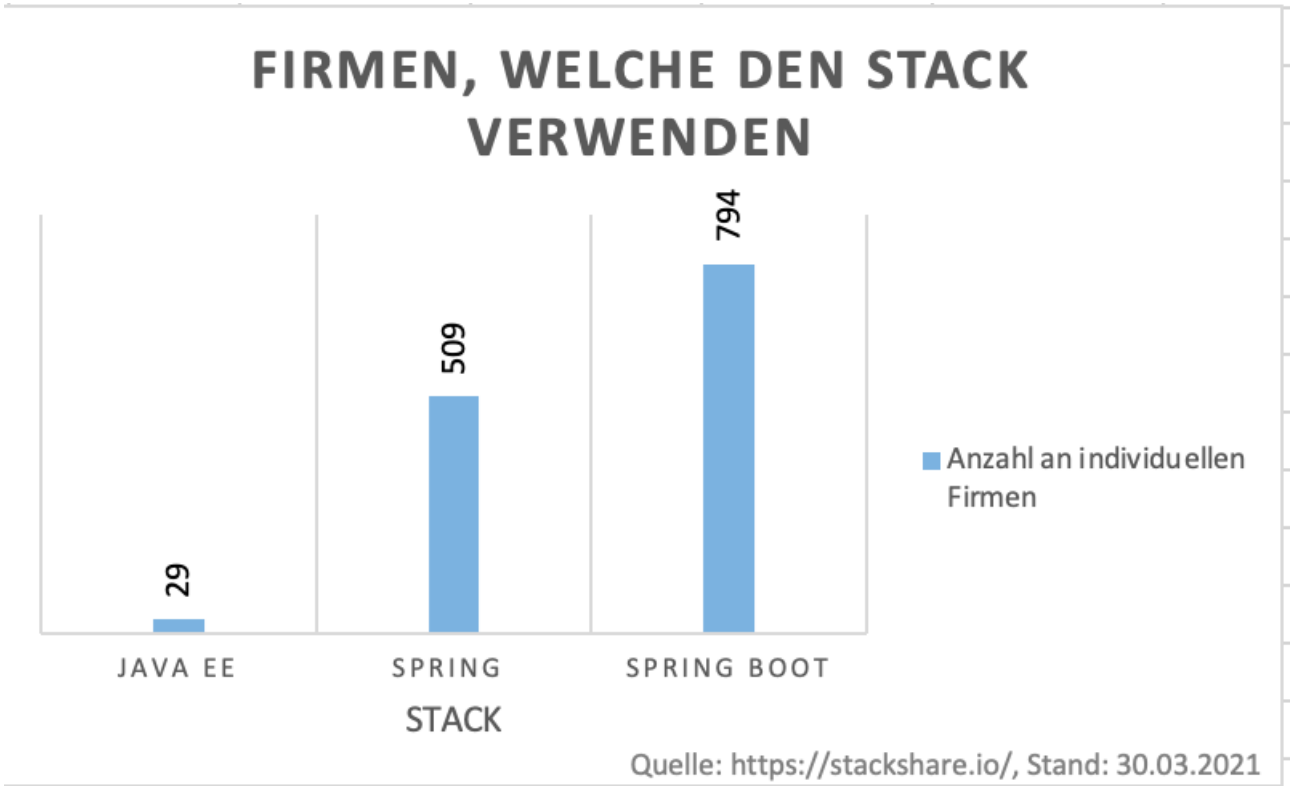


Figure 6. Firmen, welche den Stack verwenden

Auch zeigt der Trend, dass viele Firmen auf neue Stacks wie Spring und Spring Boot setzen und nur mehr wenige Unternehmen Java EE als Stack angeben, welchen sie verwenden. Dies zeigt den heutigen Einsatz der zwei Frameworks ziemlich deutig, denn Spring (Boot) hat hierbei klar die Führung.

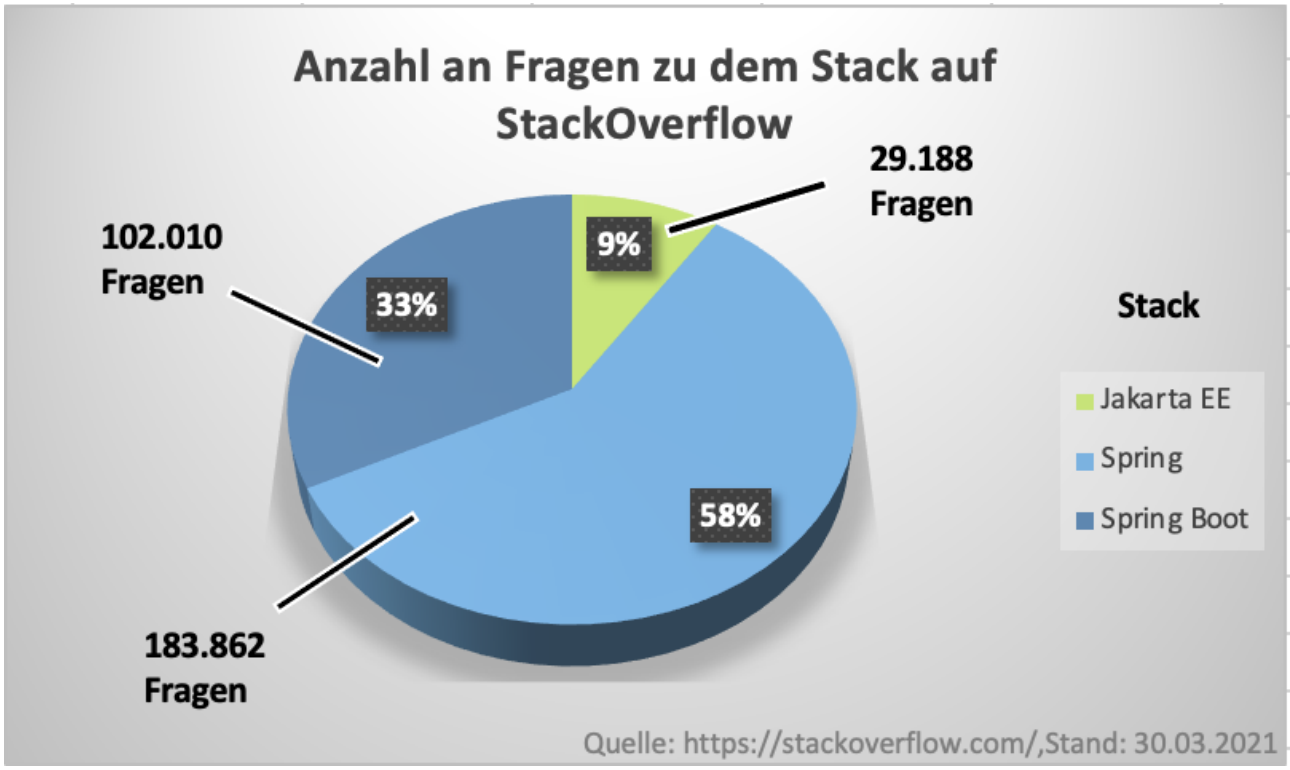


Figure 7. Anzahl an Fragen auf StackOverflow zu dem Stack

Am wohl bekanntesten Coding Portal "StackOverflow", wo täglich tausende Coding spezifische Fragen gestellt werden, dass es bei Spring insgesamt über 100.000 Fragen gibt, bei Spring Boot sogar mehr als 180.000. Java EE bzw. Jakarta EE hat demnach nur mehr als 29.000 Fragen. Dies zeigt, dass eine größere Community hinter Spring (Boot) steht und es sehr viele Fragen bzw. Informationsquellen dazu gibt, wohin gegen Java EE nur etwa ein Drittel der Fragen von Spring hat, und so anscheinend weniger relevant ist und Entwickler weniger Fragen bzw. hilfreiche Informationen auf StackOverflow diesbezüglich zur Verfügung stehen.

Durch die wenige Benutzung (von Firmen) und Information auf StackOverflow, erhält Java EE eine Punktezahl von 3 Punkten, wohingegen Spring (Boot) mit weitausgehend mehr Entwicklern, Firmen und Informationen die volle Punktezahl erhält.

	Java EE	**Spring**
Punkte	3	5

Table 3. Punkte Resümee "Developer Codebase und

Community Größe"

Chapter 11. Querschnittsfunktion

Chapter 12. Unterschiede

Chapter 13. Entscheidungsresümee

	Java EE	Spring
Regelmäßige Updates	5	5
Langfristiger Support	3	5
Kosten, Eleganz beim Programmieren	0	0
Funktionsumfang	0	0
Developer Codebase und Community Größe	3	5
Querschnittsfunktion	0	0
Ergebnis	**11**	**15**

Table 4. Resümee

Chapter 14. Verwendung von Spring Boot im Diplomprojekt

Im Diplomprojekt "ScanBuyGo" wurde als Framework auf die Verwendung von Spring, genauer Spring Boot, gesetzt.

Grund dafür war, dass bereits viel Erfahrung und praktische Programmierung Fähigkeiten in der Informatik Ausbildung erlernt wurden und so eine Programmierung mit dem Framework leicht fiel. In Kombination mit der Programmiersprache Kotlin und dem Build Tool Gradle wurde eine REST-API Lösung für das Projekt programmiert. Hauptaugenmerk lag auf der Verwendung einer Microservice Architektur statt einer Monolithen-Architektur, um einzelne Module unabhängiger und einzel startfähig zu machen und neues Know-How zu erlangen.

Ausschlaggebend war außerdem, die sehr gute Dokumentation des Frameworks, die herausstechenden Funktionen wie eingebetteter Tomcat Server, automatisierte Build Abläufe und produktionsfähige Metriken wie Health Endpoints sowie allgemein die Arbeit, welche Spring dem Backend Team durch vorgefertigte Templates, Projekte und Module abgenommen hat.

Glossary

REST-API

Programmierschnittstelle, welche über HTTP-Anfragen mittels CRUD Operationen agiert.

SOAP-API

Mit diesem Netzwerkprotokoll können Daten in Form von Envelopes zwischen System ausgetauscht werden.

Framework

Programmiergerüst, bei dem vorgefertigte Rahmen, wie Funktionen und Elemente, bereitgestellt wird und den Einstieg in die jeweilige Technologie erleichtert.

Build Tool

Automatisiert den Prozess der Bildung ausführbarer Dateien. Software wird erstellt und beispielsweise werden nötige Dependencies heruntergeladen und verwaltet.

JSP Pages

Steht für "Jakarta Server Pages" und sind Seiten gebaut durch "JHTML" und erlaubt die Integrierung von Java Code in HTML und XML Webseiten.

Microservice Architektur

Anwendungen werden in kleine Module aufgeteilt und werden besser separat steuerbar und unabhängiger. Zusammen bilden alle Module die Anwendung.

Monolithen Architektur

Alle Software Komponenten befinden sich in einem großen Anwendungssystem, sie sind zentral, einzelne Softwareteile untrennbar und kaum unabhängig steuerbar.

Quellen

Beschreibung	Quelle	Letzter Zugriff
Flaticon - Bild Icons	https://www.flaticon.com/	29.03.2021
Java EE Architektur - Bild nachmodelliert	http://pawlan.com/monica/articles/j2eearch/art/container1.jpg	29.03.2021
Spring Boot Architektur - Bild nachmodelliert	https://www.javatpoint.com/spring-boot-architecture	29.03.2021
Java EE Spezifikationen	https://www.javatpoint.com/java-ee	29.03.2021
Spring Framework Überblick	https://spring.io/projects/spring-framework	29.03.2021
Spring Funktionen	https://spring.io/why-spring	29.03.2021
Jakarta (Java) EE Wikipedia	https://en.wikipedia.org/wiki/Jakarta_EE	29.03.2021
Java EE Versionen, Funktionen	https://www.oreilly.com/library/view/java-ee-6/9781449338329/ch01.html	29.03.2021
Spring Framework Dokumentation Überblick	https://docs.spring.io/spring-framework/docs/4.3.20.RELEASE/spring-framework-reference/html/overview.html	29.03.2021

Beschreibung	Quelle	Letzter Zugriff
How Oracle killed Java EE	https://headcrashing.wordpress.com/2019/05/03/negotiations-failed-how-oracle-killed-java-ee/	29.03.2021
Java EE vs. Spring Statistiken	https://www.jrebel.com/blog/java-ee-vs-spring	29.03.2021
Spring Boot Erste Schritte	https://spring.io/guides/gs/spring-boot/	30.03.2021
Framework Community Statistiken	https://stackshare.io/	30.03.2021
Framework Fragen Statistiken	https://stackoverflow.com/	30.03.2021
Java EE REST Service Erste Schritte	https://www.jetbrains.com/help/idea/creating-and-running-your-first-restful-web-service.html#run_config	01.04.2021
	https://www.zdnet.com/article/java-finally-goes-all-in-on-open-source-with-the-release-of-jakarta-ee-8/	01.04.2021
Spring Paradigmen	https://java2blog.com/introduction-to-spring-framework/	01.04.2021
Spring Reactive	https://spring.io/reactive	01.04.2021
		01.04.2021

Table 5. Quellen