

Inhaltsverzeichnis

Vergleich des Java EE Standard-Frameworks im Vergleich zu Spring zur Entwicklung von Business Applications	1
1. Abstract	2
2. Einleitung	3
3. Java EE Framework	4
4. Spring Framework	6
5. Get Started.	8
6. Regelmäßige Updates.	11
7. Langfristiger Support.	13
8. Kosten, Eleganz beim Programmieren	15
9. Funktionen	16
10. Developer Codebase und Community Größe	17
11. Querschnittsfunktion.	20
12. Vorteile und Nachteile von Java EE	21
13. Vorteile und Nachteile von Spring Framework.	22
14. Unterschied Tabelle zwischen beiden	23
15. Wann was verwenden? (Entscheidungskriterien)	24
16. Verwendung von Spring Boot im Diplomprojekt	25
Glossary	26
References	27

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

Chapter 1. Abstract

In diesem Paper werden die zwei (Web) Frameworks, Java EE und Spring, auf unterschiedliche Kriterien, wie 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 der zu wählenden Architektur für 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 wollen. 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 zu wählen ist und welches 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 Punktezahl vergeben, welchen den positiven Nutzungsgrad widerspiegelt. Dafür werden 0 bis maximal 5 Punkte vergeben, wobei 0 für "Nicht gut/vorhanden" und 5 für "Sehr gut" steht. Die Punktezahl wird am Ende jeder Kategorie angezeigt und am Ende ein finales resümierendes Fazit gegeben.



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

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 mit 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 die 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 Anfragen und Antworten
- **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 könnte wie folgt aussehen:

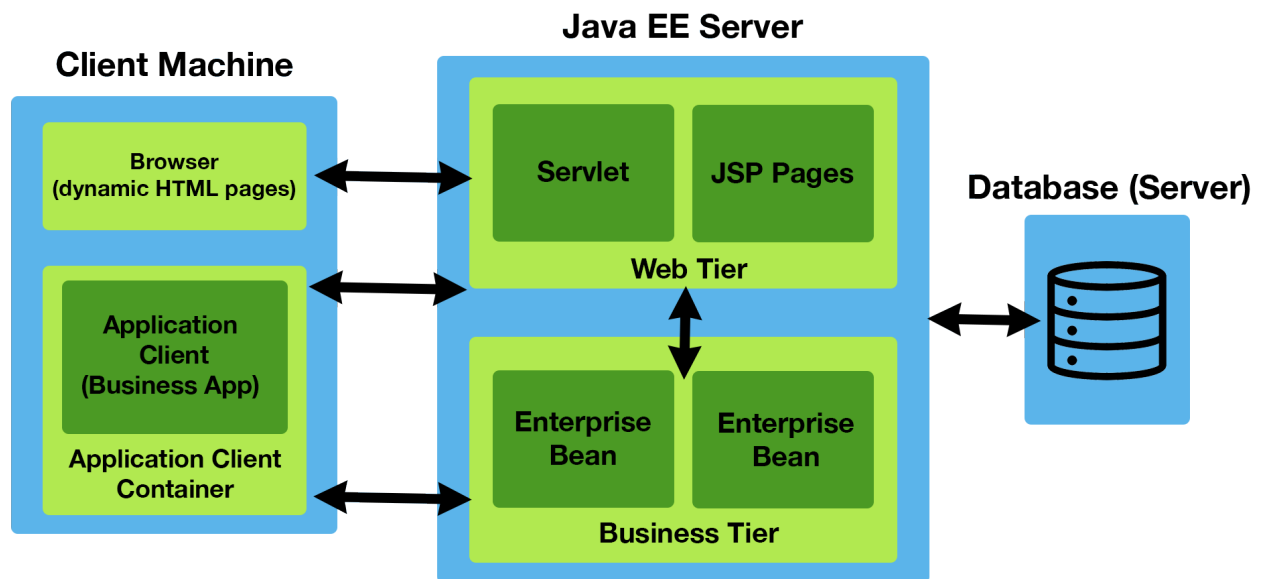


Figure 1. Java EE Architektur

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 die Enterprise Beans sind. Der Server ist mit einer Datenbank bzw. einem Datenbank-Server verbunden, worüber Datenbanktransaktionen durchgeführt werden können. Transaktionen können mit zum Beispiel einer REST-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. 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, um den Einstieg zu erleichtern und aufwendiges Aufsetzen des Frameworks zu unterbinden. Wohingegen Java EE nur mit der Programmiersprache Java läuft, so braucht Spring keine spezifische Sprache, doch meist wird Java oder Kotlin verwendet.

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

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

Eine Spring Boot Web Applikation Architektur sieht wie folgt aus:

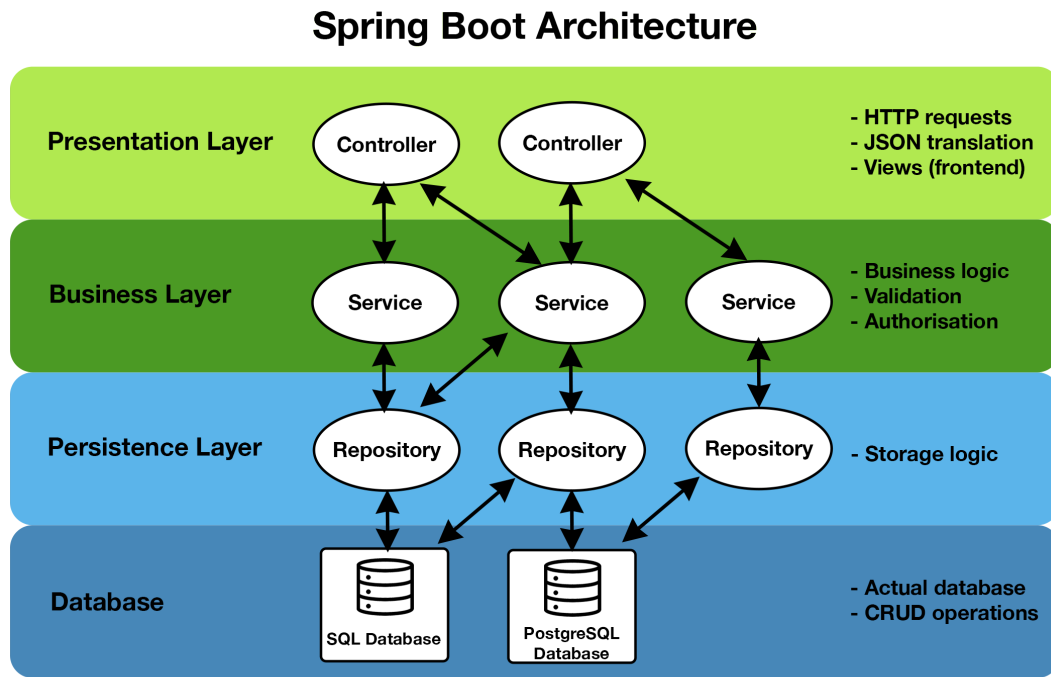


Figure 2. Spring Boot Framework 4- Schichten Architektur

Die Spring Boot Framework Architektur besteht aus vier Teilen. Die “Presentation Layer” Ebene, welche sich zum Beispiel um eingehende und ausgehende HTTP Requests/Response kümmert. Sie übersetzt die Daten von und zu JSON Objekten, um den Umgang mit der Applikationen zu gewährleisten. Grundsätzlich ist sie mit einer Front-End 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 werden 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. Get Started

Java EE:

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. Seite besuchen und wichtige Informationen wie Build Tool, Sprache, Versionen und Namen vergeben.
2. Anschließend rechts mit Klick auf den Knopf "ADD DEPENDENCIES..." klicken und die gewünschten Dependencies wie "Spring Web", "Spring Data JPA", "H2 Database" oder "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 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>

```

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 der Methode
- Simple Rückgabe eines String, um Funktion zu testen

Darüber hinaus hat der "Spring Initializr" auch eine Main Klasse erzeugt, welche nach Belieben 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.

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

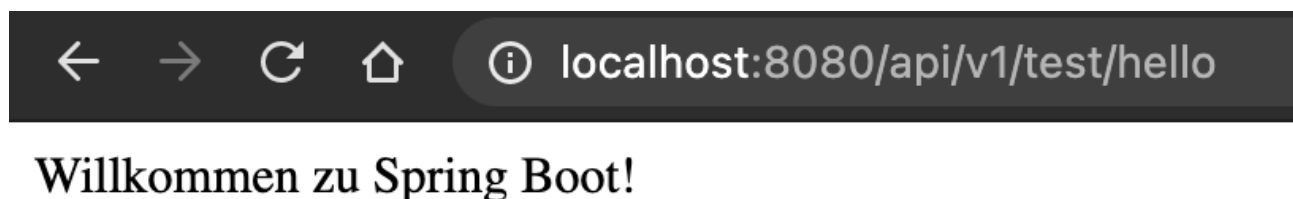


Figure 3. REST API Response localhost

Chapter 6. Regelmäßige Updates

Java EE, seit Java EE 8 umbenannt auf Jakarta EE, erhält regelmäßig neue Version Updates. 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) war im Jahr 2020. Das Spring Framework wird kontinuierlich aktualisiert. Das letzte große Update war 2017 mit 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 öfter 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 seitens 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 aufgeteilt 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.

Fazit:

	Java EE	Spring
Punkte	5	5

Table 1. Punkte Resümee "Regelmäßige Updates"

Chapter 7. Langfristiger Support

Wie bereits im vorhergehenden Kapitel eingegangen, 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. Dies zeigt auf, dass Java EE immer weniger Relevanz in der heutigen Software Gemeinschaft hat und einen langfristigen Support fragwürdig macht. Zwar kann man noch mit einigen Updates in der Zukunft rechnen, doch bahnbrechende Updates werden auf sich warten lassen.

Die Website “JRebel” 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 gewechselt haben bzw. es tun möchten, wohingegen 36 Prozent eher von Java EE zu Spring migriert haben bzw. es tun möchten.

Viele Portale sprechen von dem Tod von Java EE, nachdem Oracle die Leitung abgegeben hat. “Negotiations Failed: How Oracle killed Java EE”, so beispielsweise der Autor Markus Krag in esinem 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.

Fazit:

	Java EE	Spring
Punkte	3	5

Table 2. Punkte Resümee "Langfristiger Support"

Chapter 8. Kosten, Eleganz beim Programmieren

Chapter 9. Funktionen

Chapter 10. Developer Codebase und Community Größe

Seitens beider Frameworks gibt es eine größere Developer-Base. Java EE bzw. Jakarta EE wird laut der Webseite “<https://stackshare.io/>”, welche unterschiedliche Frameworks, Programmiersprachen etc. bewertet und aufzeigt, wer und welche Technologien verwendet, von 29 Unternehmen genutzt. Darunter “TripAdvisor”, “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”.

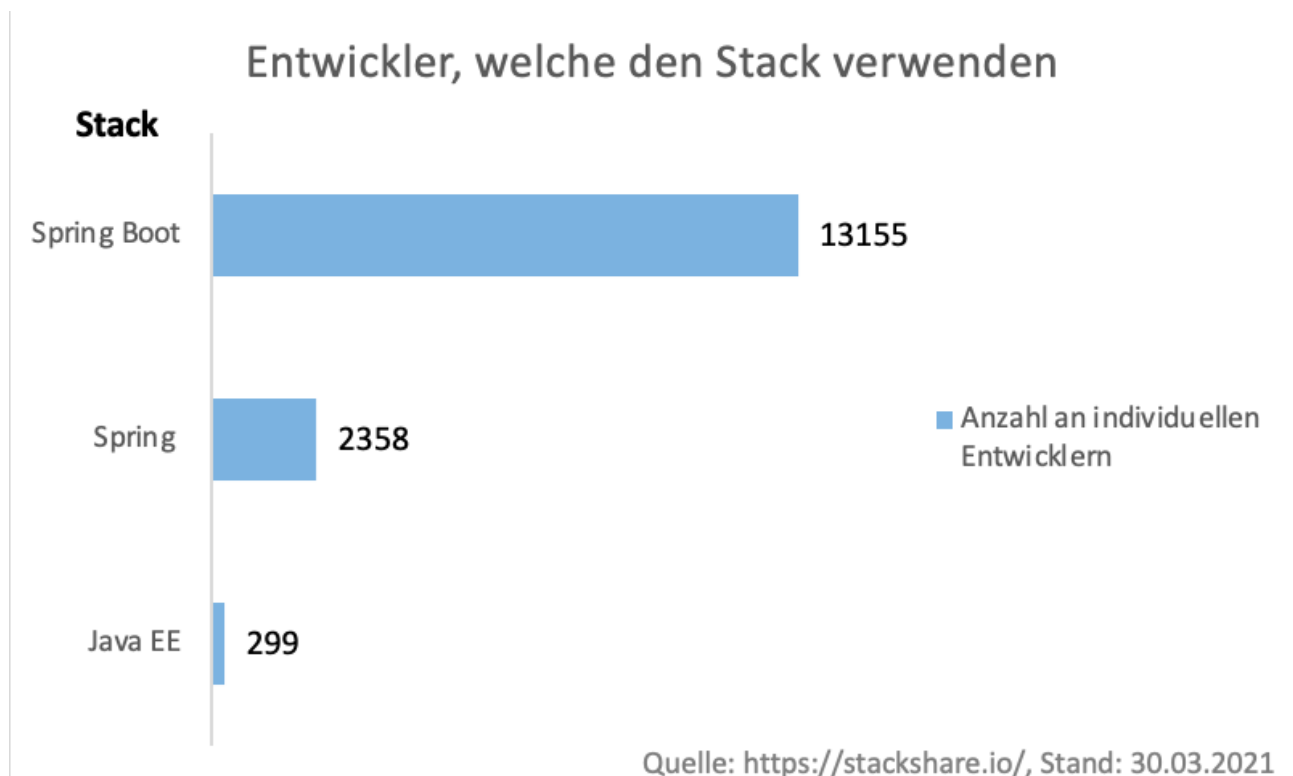


Figure 4. Entwickler, welche den Stack verwenden

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

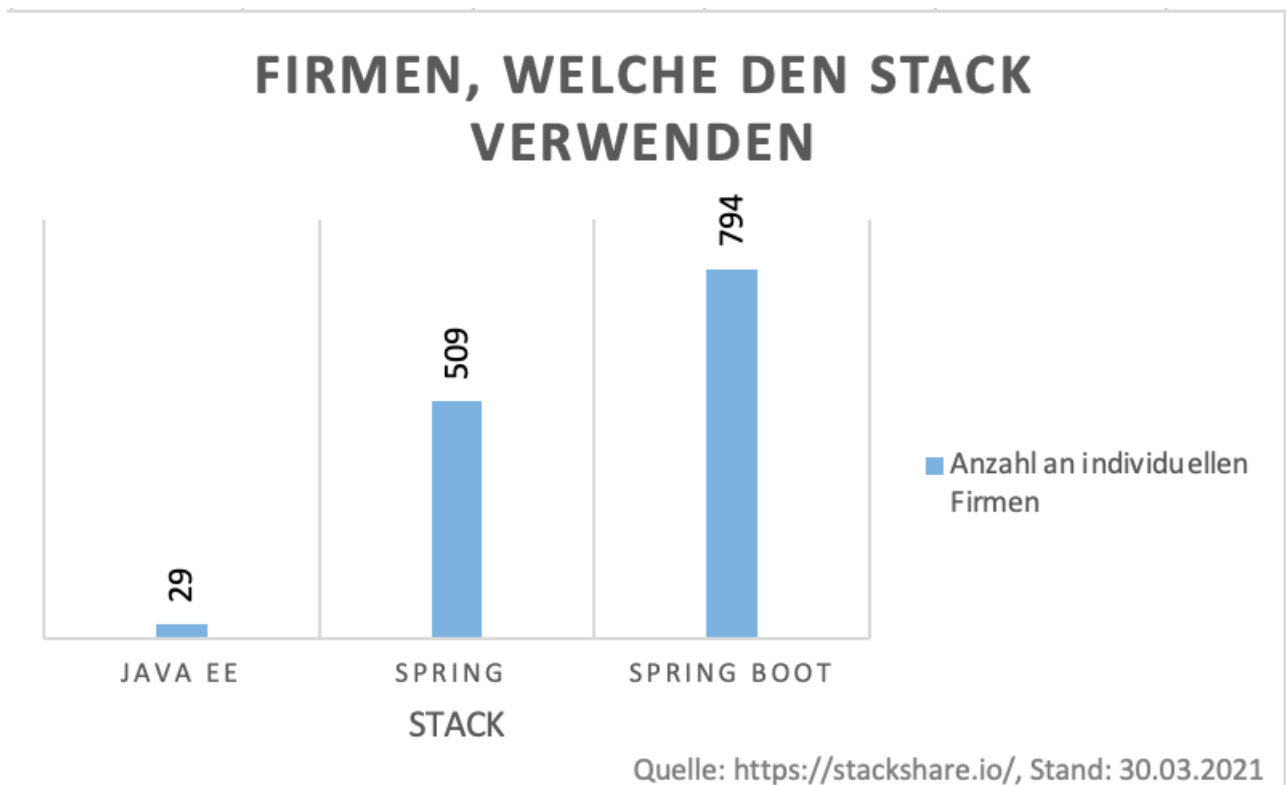


Figure 5. Firmen, welche den Stack verwenden

Auch zeigt der Trend, dass viele Firmen auf neuer 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 deutlich, denn Spring (Boot) hat hierbei klar die Führung.

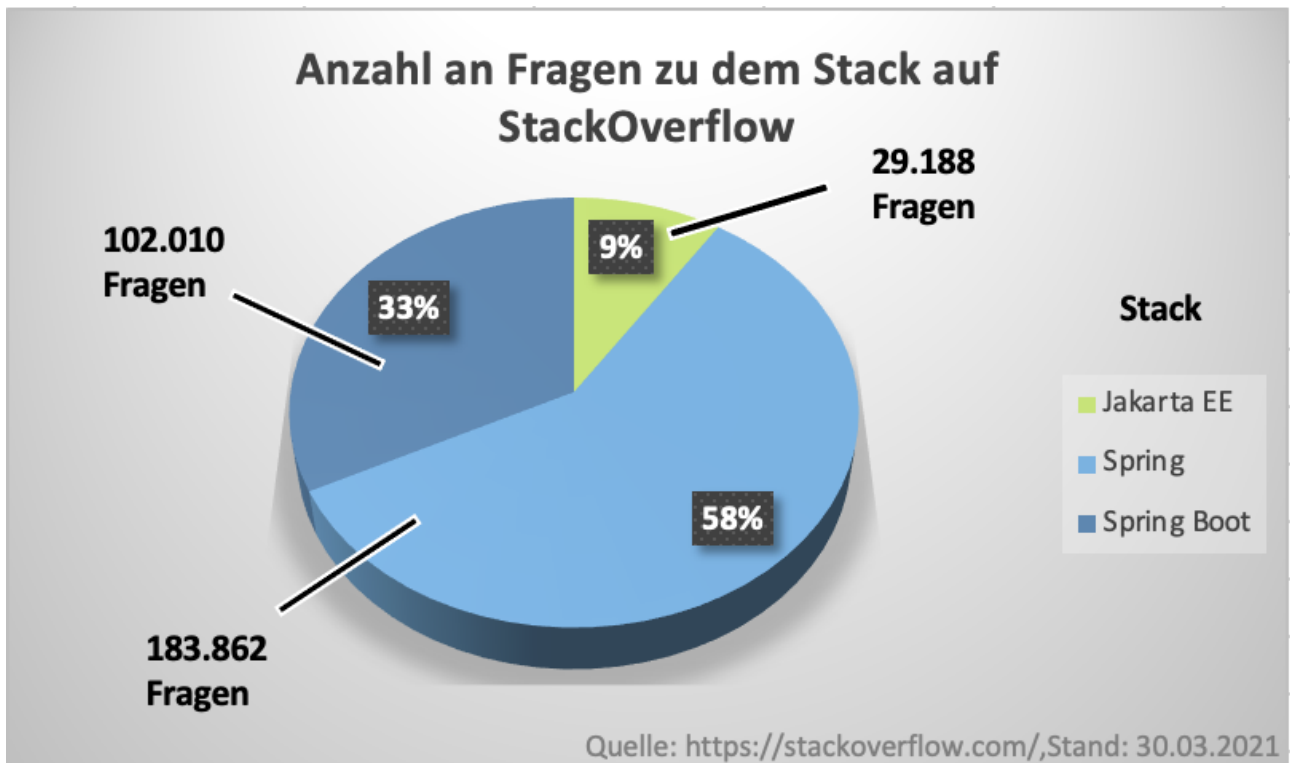


Figure 6. Anzahl an Fragen auf StackOverflow zu dem Stack

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 anscheinlich weniger relevant ist und Entwickler weniger Fragen bzw. hilfreiche Informationen auf StackOverflow diesbezüglich zur Verfügung stehen.

Chapter 11. Querschnittsfunktion

Chapter 12. Vorteile und Nachteile von Java EE

Chapter 13. Vorteile und Nachteile von Spring Framework

Chapter 14. Unterschied Tabelle zwischen beiden

Chapter 15. Wann was verwenden? (Entscheidungskriterien)

Chapter 16. 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.

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.

References

Beschreibung	Quelle	Zugriff
Flaticon - Bild Icons	https://www.flaticon.com/	29.03.2021
Java EE Architektur - Bild nachmodelliert	http://pawlan.com/monica/articles/j2eeearch/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#:~:text=The%20Java%20EE%20stands%20for,distributed%20computing%20and%20web%20services.	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

Beschreibung	Quelle	Zugriff
Spring Framework Dokumentation Überblick	https://docs.spring.io/spring-framework/docs/4.3.20.RELEASE/spring-framework-reference/html/overview.html	29.03.2021
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 Get Started	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

Table 3. Quellen