



# Enterprise Java mit Spring

Eine Praktisch Einführung  
EB Zürich

# Über Mich

## Beruflich

- Software Engineering
- Cloud Enablement Themen
  - Cloud Transformation, DevOps
- Professionalität und Professionalisierung in der IT

## Nicht Beruflich

- Snowboarden, Sport



[github.com/Vad1mo](https://github.com/Vad1mo)



[twitter.com/vad1mo](https://twitter.com/vad1mo)

# Kursablauf

- **Jeden Donnerstag - 18:00 - 21:30 - BiZE - Raum 211**
- 13. Juni 2019 18:00 - 21:30
- 20. Juni 2019 18:00 - 21:30
- 27. Juni 2019 18:00 - 21:30
- 4. Juli 2019 18:00 - 21:30
  - Ggf. etwas früher zwecks gemeinsamen Abschlussabend
    - In unmittelbaren Umgebung EB Zürich
    - Pizza, Burger, Sushi, Indisch

# Erste Kursabend

- Kennenlernen
- Organisatorisches
- Einführung
- DevEnv Setup
- Hello World!
- Aufgabenstellung
- Dependency Injection und Inversion of Control

## Zweiter Kursabend

- Umsetzung der Anforderungen für unsere Beispielanwendung
- Komponenten und die Bedeutung
- Spring MVC
- Testing

# Dritter Kursabend

- Neue Anforderungen für unsere Beispielanwendung
  - Persistenz
  - REST API
  - Cloud Deployment

# Vierter Kursabend

- Spring Ökosystem
  - Erweiterungen
  - Ergänzungen
- Alternativen zu Spring
- Diskussion
- Abschluss

# Referenz und Leitfaden

## Spring Boot 2: Moderne Softwareentwicklung mit Spring 5 von Michael Simons





# Kursgrundlage

Mit der Buchreferenz erstellen wir eine Anwendung, welche die besprochenen Themen aufgreift und praktisch umsetzt.

# Ablauf

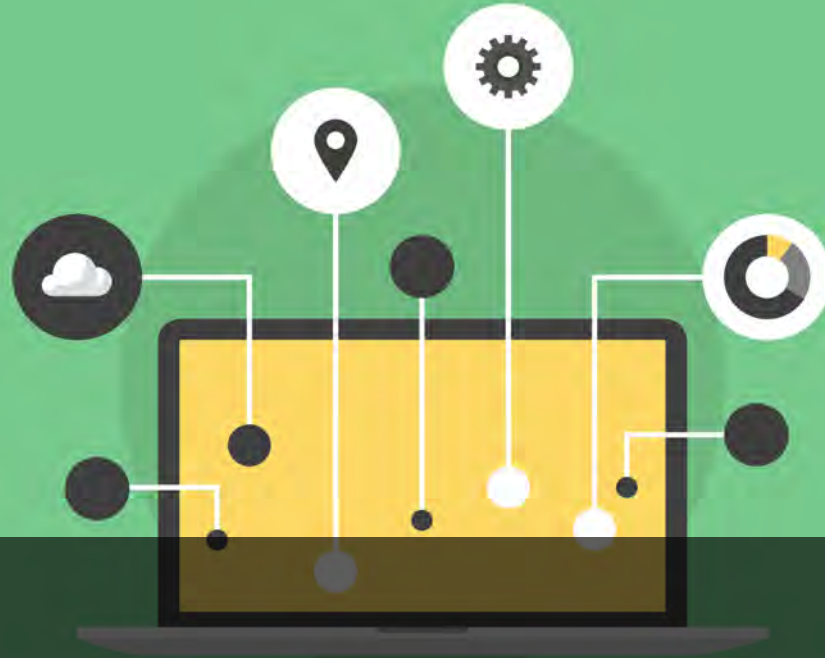
- Wechsel zwischen Theorie und Praxis
  - Kein sinnloses „Copy & Paste“
  - Erkenntnis statt Predigt
  - Raum für unterschiedlichen Geschwindigkeiten
- Gegenseitige Unterstützung
- Zwischenfragen erwünscht

- Was sind eure Erwartungen
- Was hofft Ihr heute zu lernen
- Was kennt Ihr schon über Spring

**YOUR TURN**



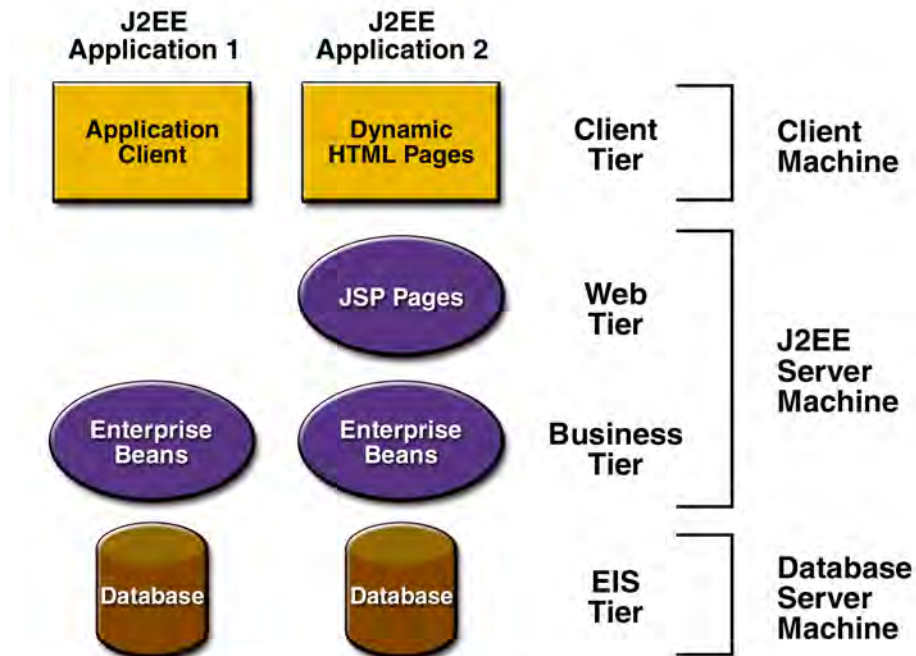
EINFÜHRUNG



# Die Geschichte von Spring und Spring Boot

## Es war einmal ...

- Java 2 Enterprise Edition (J2EE) in 1999
  - Plattform zur Entwicklung unternehmensweiten verteilten Anwendungen
  - Java Server Pages (JSP), Enterprise Java Bean (EJB), JDBC, JMS, JNDI, Java Transaction API, Java Mail
  - 4 Schichten Architektur
- Sehr Komplex
  - Fragile Konfiguration
  - XML-Hölle
- J2EE Application Server



# Die Geschichte von Spring und Spring Boot

## Idee

- J2EE Anwendungen ohne EJBs
  - Vereinfachte Entwicklung von Enterprise Java Anwendungen
  - Ergänzend zu J2EE und Java EE
  - Koexistenz mit J2EE und keine Alternative

## Erste Veröffentlichung

- 0.9 im Jahr 2003

## Kernfunktionen

- Inversion of Control mit Dependency Injection
- AOP
- JDBC, JPA
- MVC-basierte Webanwendungen
- RESTful Services

# Was ist Spring?

- Definition ist Kontextbezogen
  - Spring Framework (Core Container)
  - Familie von Komponenten rund um Spring (von Pivotal)
- Das Spring Framework ist in Module unterteilt
  - Core Container
    - Konfigurations Modell und Dependency Injection
  - Messaging
  - Transaktionale Daten und Persistenz
  - Servletbasierte Spring MVC Web Framework
  - Reactive web Framework Spring WebFlux
  - Uvm.



# Die Geschichte von Spring und Spring Boot

## Spring - 11 Jahre Später

- Flexibilität
- Breite Palette von unterstützten Anwendungsszenarien
- Komplexität
  - XML Konfiguration
  - Sehr hohe Einstiegshürde
  - Unübersichtliche Abhängigkeit
  - Wenig Cloudfreundlich

# Spring Boot

Spring Boot ist eine neue Sichtweise auf Spring

- Flachere Lernkurve
  - Konvention über Konfiguration gegenüber explizierter Konfiguration
  - Starter Pakete
- Vordefinierte Pakete
- Grösserer Fokus auf Zwölf-Faktoren-App
  - [12factor.net/de/](https://12factor.net/de/)
- Cloud Fokus
- Loslösung von JEE
  - Embedded Container

# Spring Aufbau

- Das Spring-Programmiermodell umfasst nicht die Java EE-Plattform-Spezifikation vollständig, sondern integriert ausgewählte Einzelspezifikationen aus dem EE-Dach
  - Servlet API (JSR 340)
  - WebSocket API (JSR 356)
  - Gleichzeitigkeits-Dienstprogramme (JSR 236)
  - JSON Binding API (JSR 367)
  - Bohnenvalidierung (JSR 303)
  - JPA (JSR 338)
  - JMS (JSR 914)
  - sowie ggf. JTA/JCA-Setups zur Transaktionskoordination.

# Referenzen

- Cheat Sheet mit wichtigsten Annotation
  - [bit.do/spring-boot-cheat](https://bit.do/spring-boot-cheat)
- Spring Referenzdokumentation
  - [docs.spring.io/spring/docs/current/spring-framework-reference/](https://docs.spring.io/spring/docs/current/spring-framework-reference/)
- Spring Boot
  - [docs.spring.io/spring-boot/docs/current/reference/htmlsingle/](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/)



## DEVELOPMENT ENVIRONMENT SETUP

# Basic Tooling

- Package Manager für die verschiedenen Tools
- Windows 10
  - Chocolatey, Jabba,
  - Mit WSL - SDKMAN!, Homebrew
- MacOS & Linux
  - SDKMAN!, Homebrew, Jabba
- jabba - cross-platform - Java Version Manager  
[github.com/shyiko/jabba](https://github.com/shyiko/jabba)

# Setup

- Java 11 Installation
  - Windows 10 (Chocolatey, Jabba, SDKman)
  - MacOS/Linux (SDKMAN!, Jabba, Homebrew)

## Setup 2

- IDE Installation
  - IntelliJ IDEA
  - Eclipse & Spring Tools 4 for Eclipse
  - Visual Studio Code & Spring Tools 4 for Eclipse
  - Netbeans
- Homebrew
  - `eclipse intellij-idea-ce, intellij-idea`
- Choco
  - `eclipse, intellijidea-community, intellijidea-ultimate`



# Checkliste

- ❑ Build Tool – **Maven, Gradle)**
  - ❑ Choco, SDKMAN!, Homebrew
- ❑ Docker Installieren
  - ❑ Choco, Homebrew
- ❑ **Spring Boot CLI**
  - ❑ Choco, SDKMAN!, Homebrew
- ❑ **Git**
  - ❑ Choco, Homebrew

# Checkliste

```
$ java -version  
$ idea  
$ mvn -v  
$ gradle -v  
$ docker --version  
$ spring --version
```

- ✓ Java 11
- ✓ IDE
- ✓ Build Tools
  - ✓ Gradle
  - ✓ Maven
- ✓ Docker
- ✓ Spring Boot CLI

A stylized illustration of a space launch facility. In the center, a large rocket is being mated to a mobile launcher platform by a crawler-transporter. To the right, another rocket is being moved by a similar transporter. In the background, two large smokestacks emit thick white smoke. The scene is set against a light blue sky. A dark grey horizontal bar is overlaid across the middle of the image, containing the text "HELLO WORLD" in white capital letters.

**HELLO WORLD**

# Ziel

- Erster Erfolg mit Spring Boot
- Kurzer Blick hinter die Kulissen
- Einfache Anwendung mit einem HTTP Endpunkt
  - Ausgabe vom String Hello, `${name}` im Browser
  - Wobei «name» ein Parameter ist

# Maven Projekt Setup

- Verzeichnis anlegen hello-world
- Maven Projekt erstellen
  - pom.xml anlegen
  - Inhalt in pom.xml kopieren von [bit.do/eUToi](https://bit.do/eUToi)
- Standardverzeichnisse erstellen für Projektstruktur
  - src/main/java
  - src/main/test

# Maven Project Object Model (POM)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.5.RELEASE</version>
  </parent>
  <groupId>com.example</groupId>
  <artifactId>hello-world</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>hello-world</name>
  <properties...>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build...>
</project>
```

**POM Vererbung**

**GAV von unserem Projekt**

**Abhängigkeiten**

**spring-boot-starter-web**

**spring-boot-starter-test**

[bit.do/eUToi](http://bit.do/eUToi)

# Testgetriebener Ansatz

```
@RunWith(SpringRunner.class)
@WebMvcTest
public class ApplicationTests {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void expectHelloWorldResponse() throws Exception {
        this.mockMvc.perform(
            get( uriTemplate: "/hello")
                .param( name: "name", ...values: "World"))
            .andExpect(status().isOk())
            .andExpect(content()
                .string( expectedContent: "Hello, World\n"));
    }
}
```

**Spezieller Runner JUnit**

**MVC Relevante Konfiguration  
Mock Umgebung**

**Injizieren einer Instanz**

<https://github.com/springbootbuch/helloworld>

# Unsere Businesslogik

```
@RestController
public class HelloWorldController {

    @GetMapping("/")
    public String helloWorld(
        @RequestParam final String name){
        return "Hello, " + name + "!";
    }
}
```

**Spezieller Runner JUnit**

**MVC Relevante Konfiguration**

**Test**

<https://github.com/springbootbuch/helloworld>



# Unsere Anwendung

```
@SpringBootApplication
public class Application {

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

**Spring Boot Starter  
Kompositions Annotation**

**Utility Klasse zum starten der  
Anwendung**

<https://github.com/springbootbuch/helloworld>

# Geheimnis hinter @SpringBootApplication

- @SpringBootApplication ist eine Zusammengesetzte Annotation
- Meta und Composed-Annotationen in Spring
- @SpringBootApplication ist eine Zusammensetzung aus:
  - @SpringBootConfiguration
    - Konfiguration einer Spring Boot Anwendung
  - @EnableAutoConfiguration
    - Ermittelt anhand von Abhängigkeiten welche Komponenten Verfügbar sind und konfiguriert diese
  - @ComponentScan
    - Bestandteil von Spring Core, Sucht im Classpath nach Spring Beans

# Was passiert in SpringApplication.run

- Auswahl und Erstellung eines Passenden ApplicationContext
- Registrierung einer CommandLinePropertySource
  - Veröffentlichung der Argumente über die CLI an Spring
- Aktualisierung des Kontextes mit allen gefundenen @ Komponenten
- Ausführung aller Beans vom Typ CommandLineRunner

# Aufgabe

- Findet einen Weg wie man den ASCII-Art-Banner (beim Starten der Anwendung) ändern könnte.

Kontakt

Vadim Bauer  
8gears AG

+41 786838023  
vb@8gears.com

