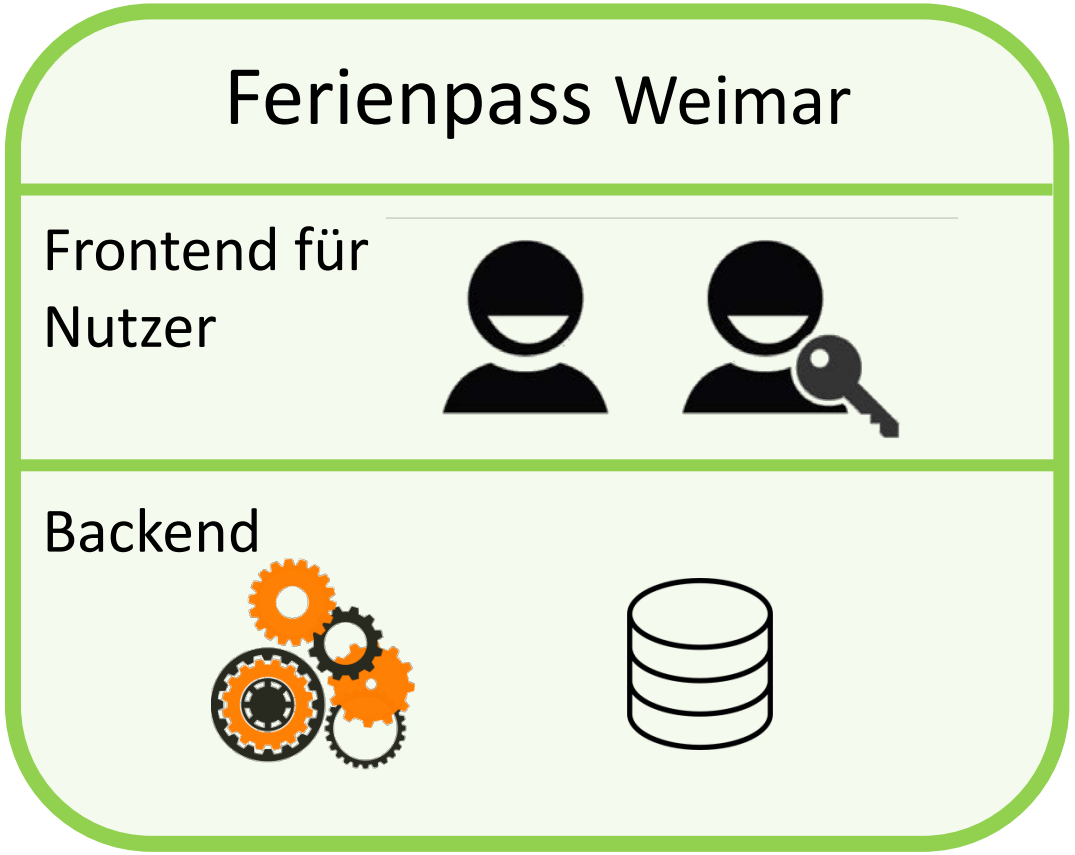
Eine Einführung zu Spring Boot

Von Anny Hißbach

Unsere Anwendung für Ferienpass-Weimar

Monolith

- Anwendung nicht in Module / Aspekte aufgeteilt
- Wartung & Erweiterung schwer
- bei Änderungen unvorhersehbares Verhalten



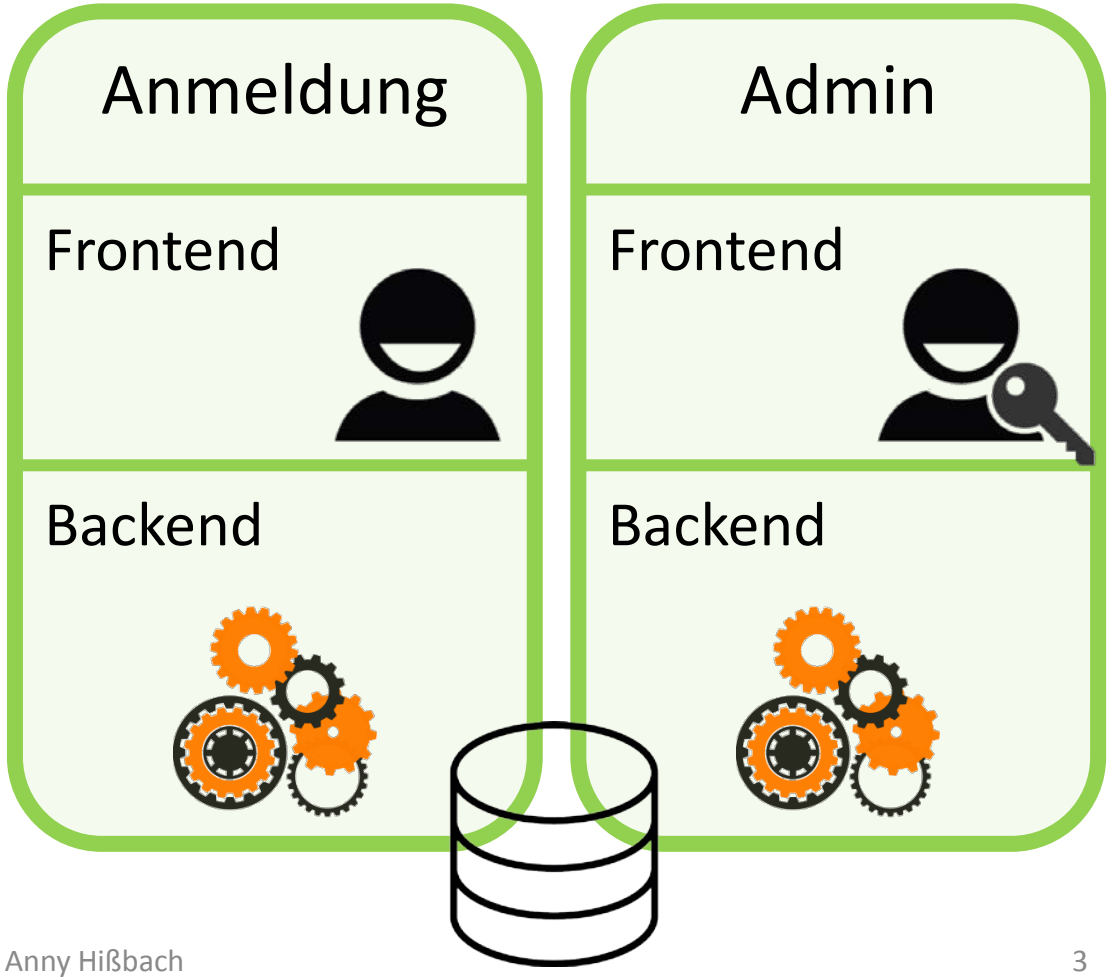
Unsere Anwendung für Ferienpass-Weimar

Monolith

- Anwendung nicht in Module / Aspekte aufgeteilt
- Wartung & Erweiterung schwer
- bei Änderungen unvorhersehbares Verhalten

Microservices

- Anwendung nach Funktion / Komponenten gegliedert
- einzelne MS isoliert wartbar
- Wiederverwendung in anderen Anwendungen



MicroServices

Eingabe

Frontend

Verarbeitung

Datenbank

Application Server

Backend

MicroServices – diese Frameworks verwenden wir

Vue.js

Eingabe

Frontend

Verarbeitung

Datenbank

Tomcat

Backend

Verwendung der Frameworks

Vue.js

… unter anderem:

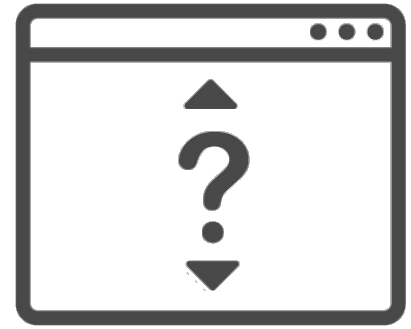
- notwendige Abhängigkeiten identifizieren
- .jars herunterladen

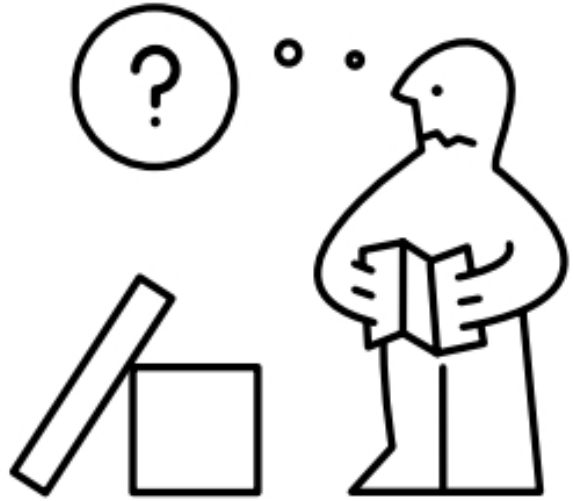
Tomcat

- Aufsetzen des Tomcat Web-Servers
- Methoden für Datenbankzugriff
- Methoden Frontend ←→ Backend

Anforderungen nicht vergessen!

- fehlerfreie Funktionalität (Tests?)
- Wartbarkeit
- Erweiterbar
- Anpassbar
- leicht zu verwenden





Unsere Rettung: Spring Boot!

Spring Boot kümmert sich um …

- Web-Server
- Anbindung der Datenbank
- REST-API
- Tests
- Metriken

… und liefert die Anwendung als ausführbare **.jar**-Datei

Was **ist** Spring Boot?

 +  =  ?

Spring

Boot

Was **ist** Spring Boot?


+

=

?

Spring Framework

- umfangreiches & mächtiges Framework
- moderne Programmierpraktiken
- Integration externer Frameworks

Boot

Was **ist** Spring Boot?

Spring Framework

- umfangreiches & mächtiges Framework
- moderne Programmierpraktiken
- Integration externer Frameworks

Bootstrap

- einfach und schnell komplexe Software erstellen
- komplexe Konfiguration
- Einstieg in Spring schwer
- Grundstruktur aufwendig

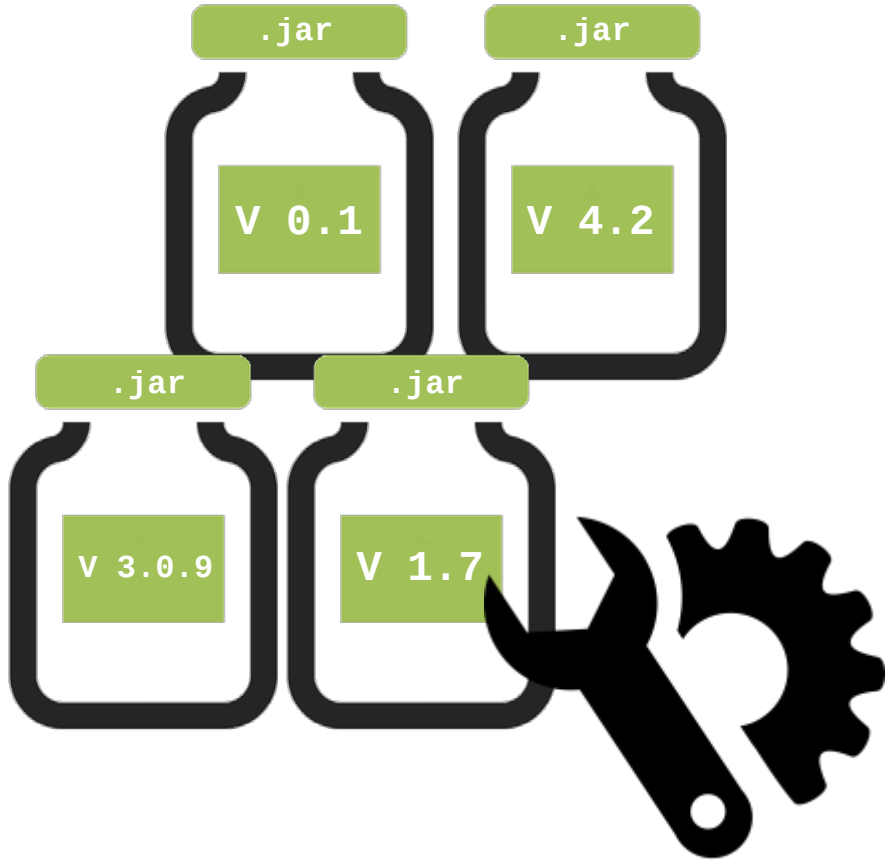
Was **macht** Spring Boot?

Verwaltung der Abhängigkeiten

- häufig benötigte Frameworks
- Abstimmung der Versionen

Konfiguration der Komponenten

„Konvention vor Konfiguration“



Konvention vor Konfiguration…?

Komplexe Software → umfangreiche Konfiguration

…die sich oft wiederholt

Konvention für den häufigsten use-case

Konfiguration für Sonderfälle

In **Spring Boot** **:**

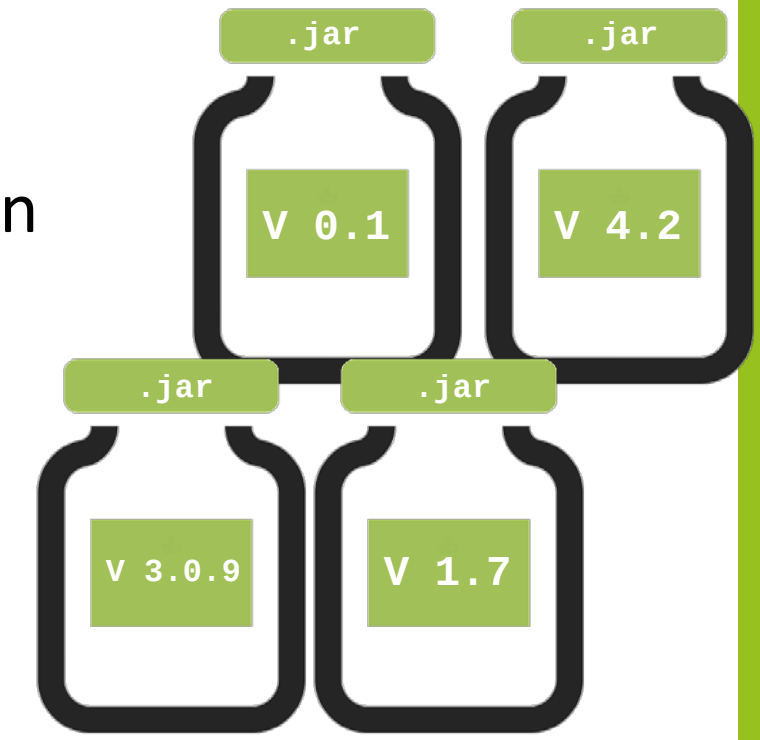
- Starters
- AutoConfiguration

Spring Boot Starters

`spring-boot-starter-*`

- Einbindung in Maven Projekt → POM.xml
- **konventionelle** Abhängigkeiten & Konfigurationen

- „Starter Parent“ → SpringBoot Abhängigkeiten
- „Starter“ → weitere Komponenten



```
//in POM.xml
<parent>
      <groupID>org.springframework.boot</groupID>
      <artifactID>spring-boot-starter-parent</artifactID>
      <version>1.5.9.RELEASE</version>
</parent>
```

```
//spring-boot-starter-parent
<properties>
        <activemq.version>5.13.4</activemq.version>
        <ehcache.version>2.10.2.2.21</ehcache.version>
        <ehcache3.version>3.1.1</ehcache3.version>
        <h2.version>1.4.192</h2.version>
        <hamcrest.version>1.3</hamcrest.version>
        <hazelcast.version>3.6.4</hazelcast.version>
        <hibernate.version>5.0.9.Final</hibernate.version>
        <hibernate-validator.version>5.2.4.Final</hibernate-validator.version>
        <hikaricp.version>2.4.7</hikaricp.version>
        <hikaricp-java6.version>2.3.13</hikaricp-java6.version>
        <hornetq.version>2.4.7.Final</hornetq.version>
        <hsqldb.version>2.3.3</hsqldb.version>
        <htmlunit.version>2.21</htmlunit.version>
        <httpasyncclient.version>4.1.2</httpasyncclient.version>
        <httpclient.version>4.5.2</httpclient.version>
        <jersey.version>2.23.1</jersey.version>
        <jest.version>2.0.3</jest.version>
        <jetty.version>9.3.11.v20160721</jetty.version>
        ....
</properties>
```

18 Zeilen aus >3000

```
//in POM.xml
<parent>
      <groupID>org.springframework.boot</groupID>
      <artifactID>spring-boot-starter-parent</artifactID>
      <version>1.5.9.RELEASE</version>
</parent>


<dependencies>
  <dependency>
      <groupID>org.springframework.boot</groupID>
      <artifactID>spring-boot-starter-web</artifactID>
  </dependency>
</dependencies>
```

Spring Boot Auto Configuration

konventionelle Konfiguration der Komponenten:

- Welche Frameworks sind im Classpath?
- Was wird bereits vom Entwickler konfiguriert?

manuelle Konfiguration:

- in **application.properties**
- mittels **Annotationen**: `@Configuration`
- **Keine** XML-Dateien

Spring Boot Auto Configuration

```
//in POM.xml
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
</dependency>
```

```
//in application.properties
spring.datasource.url=jdbc::mysql://localhost/test
spring.datasource.username=dbuser
spring.datasource.password=dbpass
```

```
//in POM.xml
<parent>  ...  </parent>
<dependencies>  ... </dependencies>

<packaging>jar</packaging>
```

```
//in Terminal

$ mvn package
$ java –jar myPath/app-0.0.1.jar
```

Hello World (web) mit Spring Boot

Starter Parent

Web-Starter

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.9.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

Hello World (web) mit Spring Boot

AutoConfiguration

REST-API

„Hello“

main

```java
package hello;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
@EnableAutoConfiguration
public class SampleController {

    @RequestMapping("/")
    @ResponseBody
    String home() {
        return "Hello World!";
    }

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

Fazit – Deshalb lieben wir Spring Boot

- mächtiges **Spring** Framework
- konventionelle **Abhängigkeiten**
- **schnell** funktionierende Anwendung in einer .jar Datei
- einfach **erweiterbar** mit vielen externen Frameworks
- auf eigene Anforderungen **konfigurierbar**