

## Spring Starter – endlich Code! 🎉

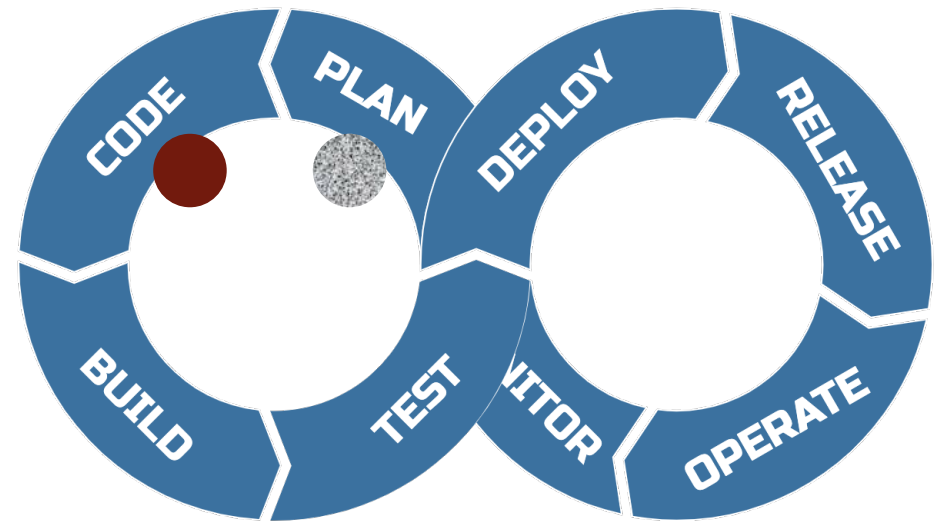
**WEITER WISSEN →**

# Ziel


Nach der Lektion haben die Studierenden eine Spring App aufgesetzt und können diese starten sowie deren initialen Inhalt analysieren.

# Spring Starter

- Java / Gradle aus Semester 1
- Docker
- Security Wissen
- Continuous Integration und GitHub Actions
- Wir brauchen ein Projekt
  - für Unterrichtsbeispiele
  - für Transferaufgaben
- **Mit Lektion 9 + Transferaufgabe haben wir eine App die man nach PROD deployen kann! Als Docker, vertretbar sicher!**



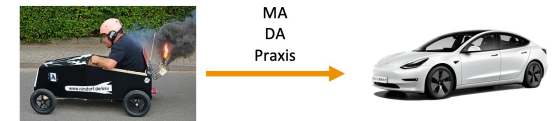
# Agenda

- Warum Java?
- Warum Dependencies? Ich will das selber coden 
- Spring
- Annotations
- Unser Spring Starter
- Zielkontrolle

# Java?

- Ihr könnt/wollt nicht 4 neue Sprachen lernen
- Wir schliessen ans Semester 1 an
- Immer noch weit verbreitet
- z.T. mit Kotlin erweitert
- Super extrem erwachsen!
  - jedes noch so kleine Problemchen wurde bereits gelöst
  - u.a.
    - Multi-Tenancy isoliert
    - Datenbank ORMs und Migrations
    - Security
    - Logging, Tracing, Observability
- Schlussendlich lernen wir Techniken, nicht genau eine Sprache!

Analogie: Ihr könntet auch jedes Auto fahren, nicht nur Golf 3 TDI 1.4l Diesel mit 4 Gurten



# Warum führen wir jetzt Dependencies ein?

- Zeit für das Tattoo...
- **DO NOT REPEAT (yourself)**
- DRY gilt nicht nur für Sicherheits-Gedanken sondern auch für alles andere!
  - Wir vermeiden es, irgendwas was es schon gibt neu zu erfinden
    - Achtung, «Dependency Hell» (nur Erfahrung kennt diese Grenze)
      - Um `STRING` zu `string` zu «converten» brauchen wir keine neue Library
  - Beispiele
    - User Management
    - Authentication
    - Datenbankzugriffe
    - Email Templates
    - etc.
- Klar, das kann selber gebaut werden, ist aber generell nicht «empfohlen»\*

## Die einfachste und effektivste Regel

- DR(Y)
  - Experten investieren  $\infty$  Aufwand in sichere Abhängigkeiten → **Nutzen**
  - Lösungen gibt es für jede Sprache!
    - Java – Gradle, Maven (Lektion 8 und 9)
    - Node – NPM, Yarn
    - Go – `go.mod`
    - PHP – Composer, Packagist
- Gründe
  - Siehe </docs/theory/principles#do-not-repeat>
  - Die Maintainer werden VULNAS schliessen, wir updaten
  - Alleine wird es bald ein 1-Personen-Projekt, dann 2, dann 3, dann 🧑

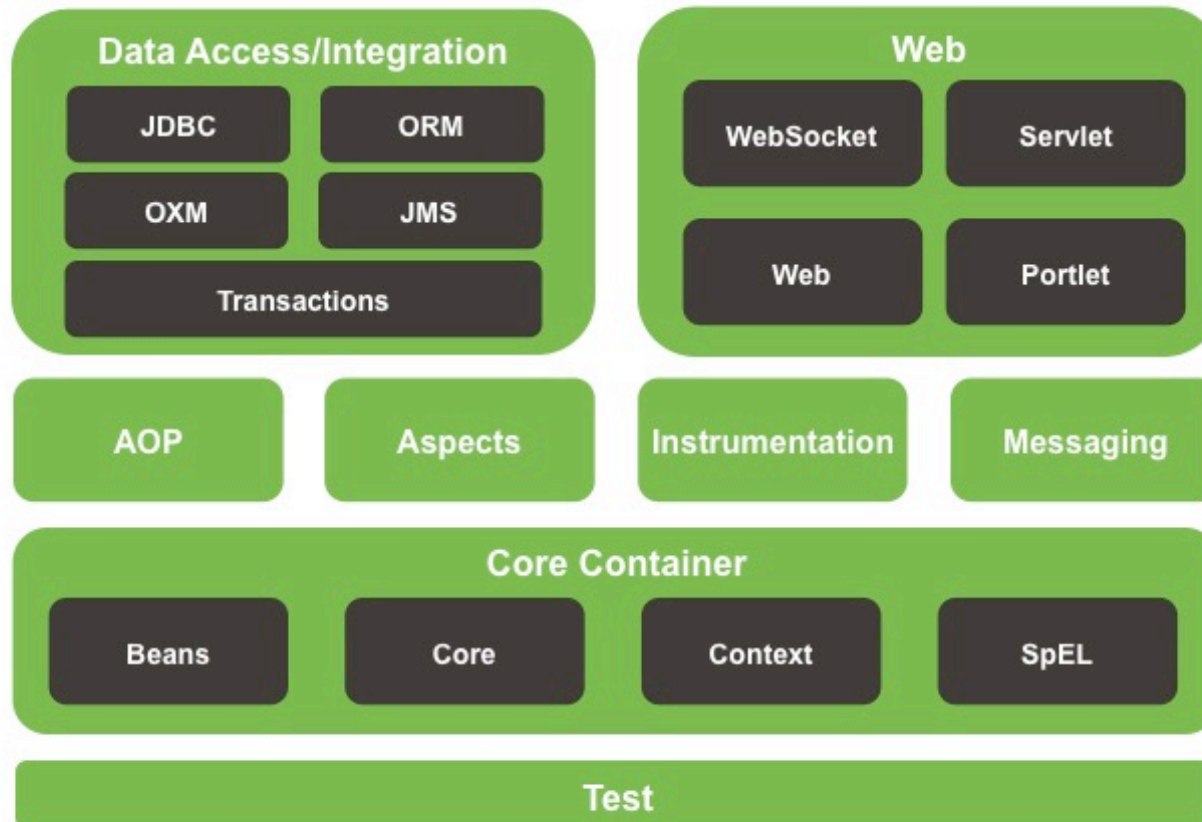


## – **DO NOT REPEAT**

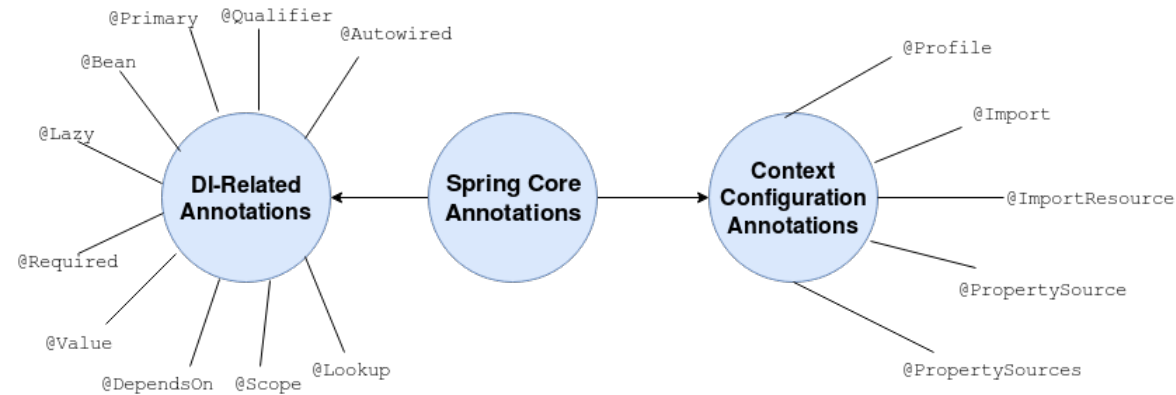




## Spring Framework Runtime



– Alternative: Spark, nutzt jedoch primär Kotlin



### Spring MVC Annotations



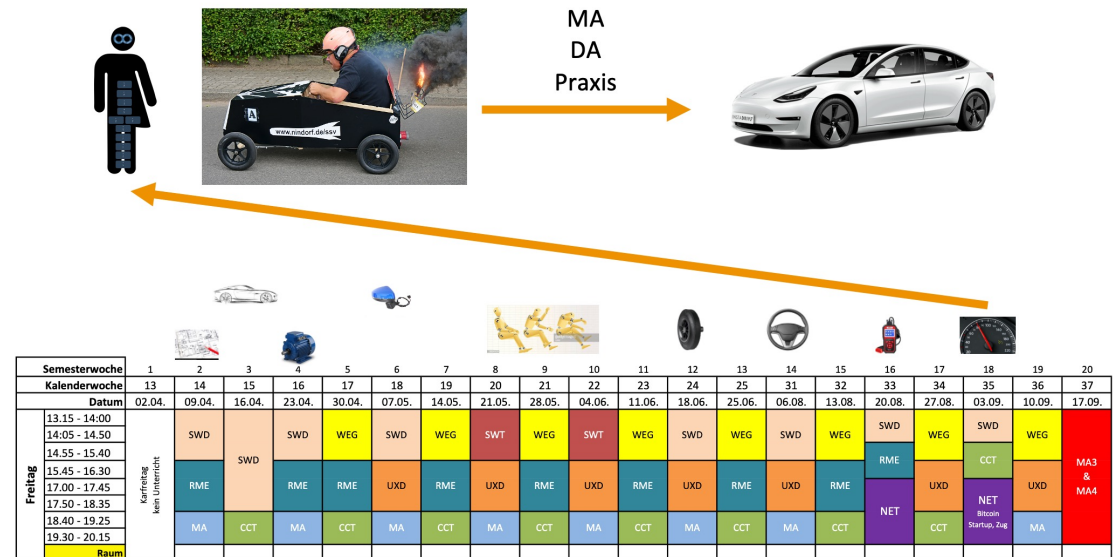
1. @Controller
2. @RequestMapping
3. @RequestParam
4. @PathVariable
5. @RequestBody &
6. @ResponseBody
7. @RestController

- Annotations dienen der Dependency Injection und der Konfiguration
- Wir instruieren Spring damit, etwas zu laden oder zu konfigurieren



# nds-swe/spring-starter

- Unser Starter  
<https://github.com/nds-swe/spring-starter>
- Nun interessant: Tag 1.0.0
  - Merke: Die Musterlösungen sind einfach weitere Tags in Git 😊
  - Tags sind Alias' für Commits, Commits sind unique
- Diesen Starter werden wir hegen und pflegen
- Er dient nur einem Zweck:



# Starter Klonen

In der Transferaufgabe werden Sie Ihren eigenen Starter anfangen!

Klonen Sie jetzt meinen Starter vom Tag 1.0.0

```
git clone https://github.com/nds-swe/spring-starter.git dozenten-starter && cd dozenten-starter  
git checkout sample-solution-1.0.0
```

Starten Sie den Starter mit

```
./gradlew bootRun
```

Windows bin ich nicht 100% sicher, Sie aber (Expectations).

**Oder, warte mal, spielt OSX oder Windows doch gar keine Rolle?**

```
git checkout sample-solution-1.3.1
```

```
docker build -t starter . && docker run starter -p 8080:8080 🍺
```

Langsam können wir die vorherigen 7 Lektionen wieder verwenden! Wir lernen langsam unser imaginäres Auto respektive die Seifenkiste zu bauen!

# Verstehen

- Ausführen klappt ja schon mal, aber was passiert da?
- tree
- StarterApplication.java
- build.gradle

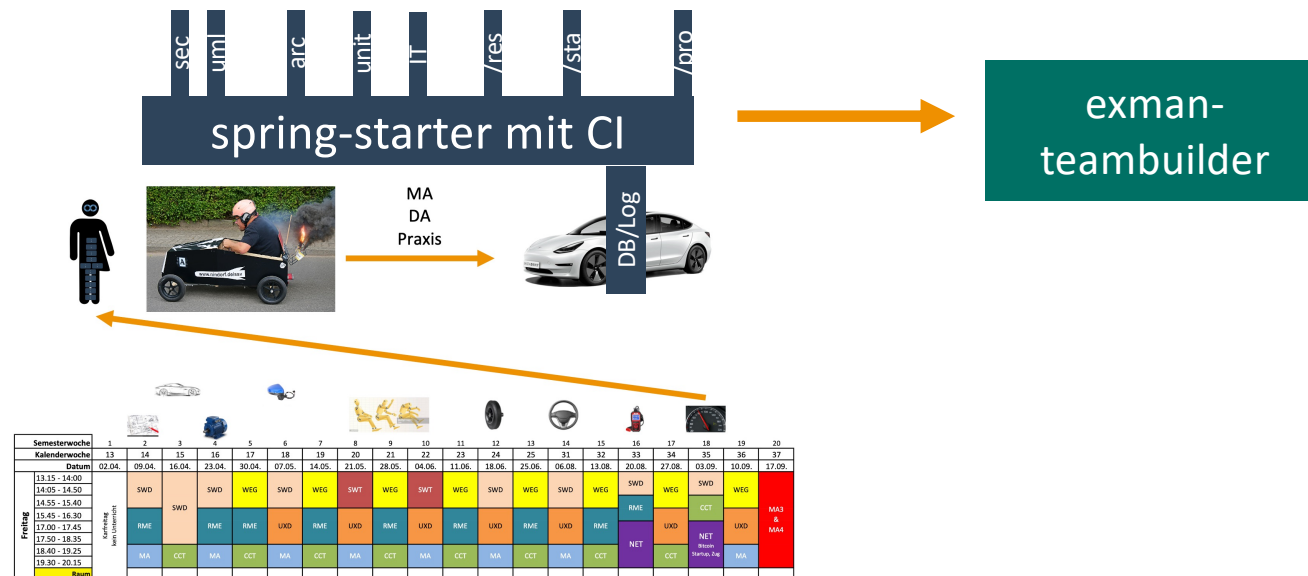
Mehr gibt's nicht...

# Zielkontrolle

Warum verwenden wir Spring und Dependencies?

Wozu dienen «Annotations» in Spring?

Hinweis: Spring «lernen» ist kein 5 Minuten Job. Dieser Aufwand, Dependencies zu verstehen und zu Nutzen ist aber nicht bezahlbar! Egal welche Sprache Sie in MA, DA oder Leben verwenden, die Prinzipien sind **IMMER** gleich.







**WEITER WISSEN.**

**Wir begleiten Sie!**