Josef Roth, 2017

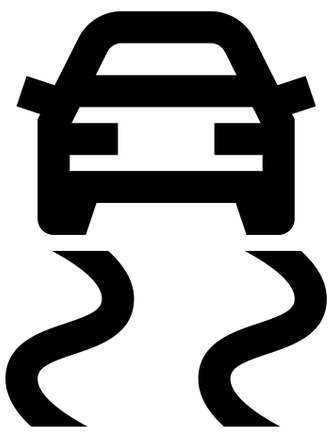
Spring Boot

# METRIKEN

# WAS, WIE, METRIK?

- Wonach bewertet man Software?
- Wie kann man Software über Zeit analysieren?

## Frage nach der Softwarequalität

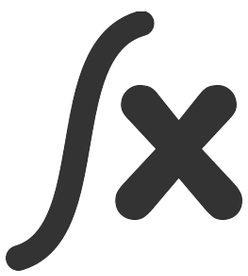


# DEFINITION

**„Eine Softwarequalitätsmetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet, welcher als Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit interpretierbar ist.“**

**– IEEE Standard 1061, 1998**
Institute of Electrical and Electronics Engineers

**Software** → **Metrik** → **Maßzahl**



**42**

Bildquellen: http://www.urltarget.com/images/flat-theme-action-math-function-icon.png | https://www.shareicon.net/data/512x512/2016/10/29/849298_paper_512x512.png

- Interpretation ist Teil der **Softwaremetrie**
- ermöglicht Vergleich von Software

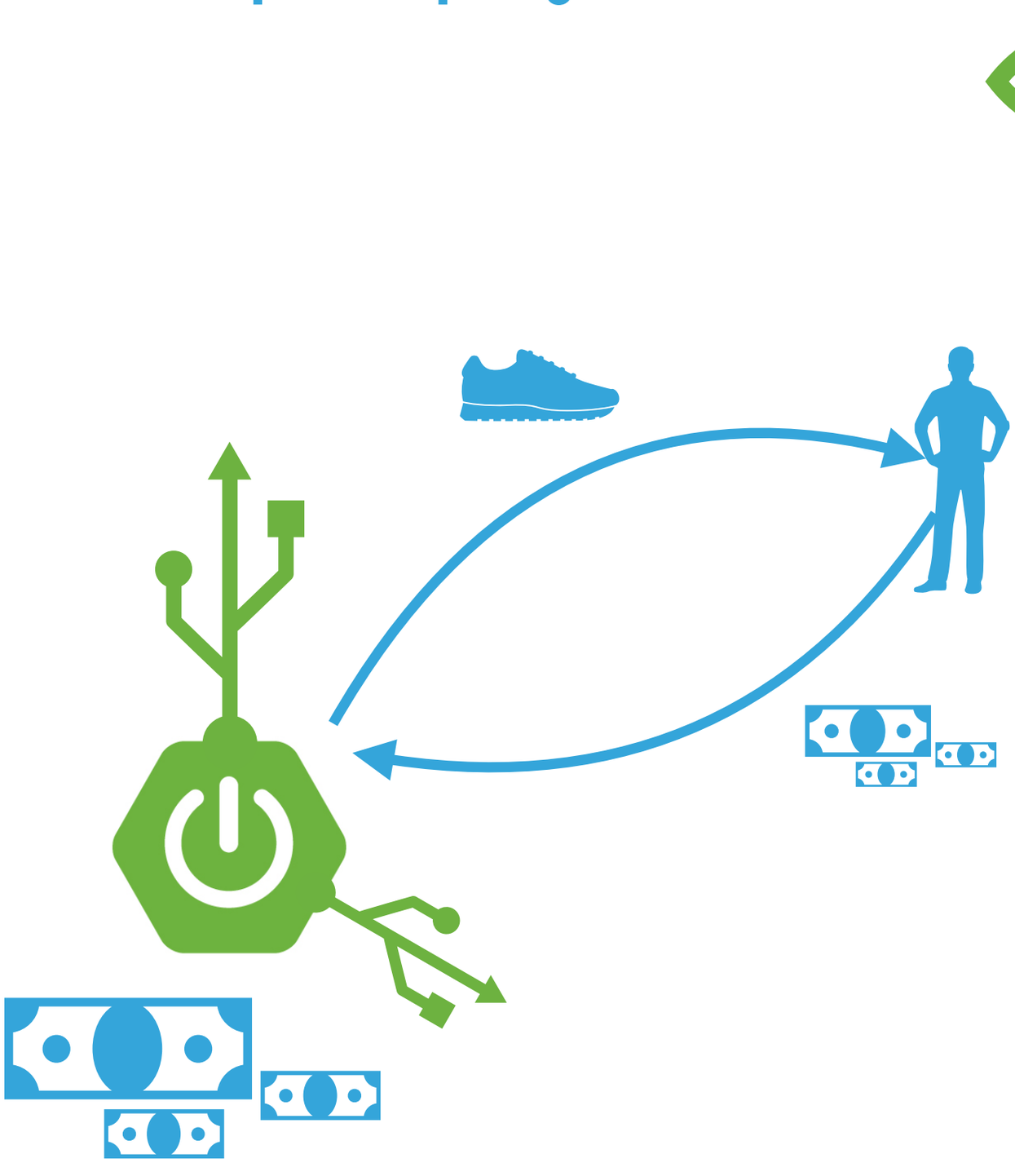
...in der Theorie


https://www.iconexperience.com/_img/g_collection_png/standard/512x512/scales.png

# Online-Service mit Spring Boot


https://cpanelhosting.ru/blog/wp-content/uploads/2017/01/ddos-attack.png

# Online-Shop mit Spring Boot



Metrik zeichnet Anzahl
an Transaktionen
(Funktionsaufrufen) auf

Metrik meldet


ermöglicht Rückschlüsse auf
Funktionalität/Effizienz des Systems

## Beschränkungen

Aus der Sicht des …

- **Managements:**
  Kostenminimierung,
  Produktionssteigerung, …

- **Entwicklers:**
  Wartbarkeit, Effizienz,
  Sicherheit, …

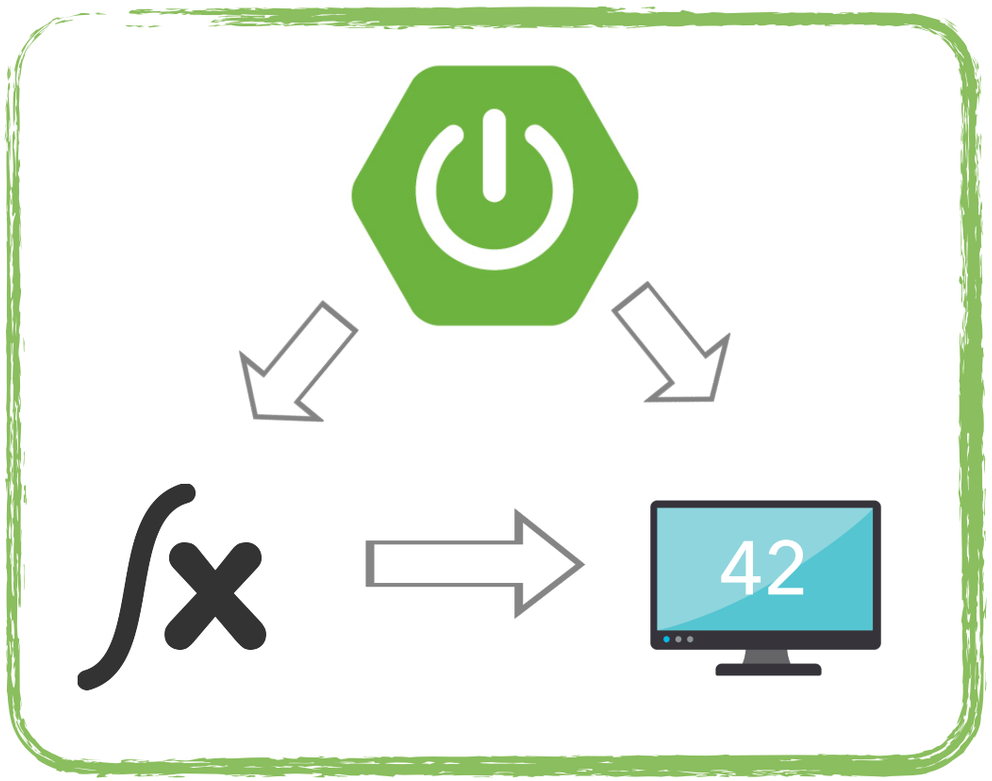
- **Kunden:**
  Erweiterbarkeit,
  Zuverlässigkeit, …

## Klassifikationen

Gegenstand der Messung in der …

- **Prozess-Metrik:**
  Ressourcenaufwand, Fehler, …
- **Produkt-Metrik:**
  Umfang (LoC), Lesbarkeit, …
- **Projektlaufzeit-Metrik:**
  Entwicklungszeit, …
- **Aufwandsmetrik:**
  Produktivität, Termintreue, …
- …

# METRIKEN IN SPRING BOOT




## Spring Boot Actuator

## HTTP Endpoints

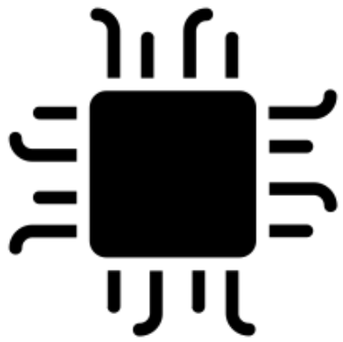
http://stormpath.com/wp-content/uploads/2016/05/spring-boot-logo.jpg | https://images.vexels.com/media/users/3/128243/isolated/preview/a1a5b9494d50bb3754bba547f2d88487-computer-monitor-flat-icon-by-vexels.png

# METRIKEN IN SPRING BOOT

1. Systemmetriken
2. Datenquell-Metriken
3. Cache-Metriken
4. Tomcat Session-Metriken
5. Individuelle Metriken

# Systemmetriken

- Systemspeicher in KB (*mem / mem.free*)
- Anzahl der Prozessoren (*processors*)
- Systemlaufzeit in ms (*uptime / instance.uptime / …*)
- Heap und Thread Informationen (*heap / threads / …*)
- Klasseninformationen (*classes / …*)
- Garbage Collection Information (*gc.xxx.count / …*)



https://d30y9cdsu7xlg0.cloudfront.net/png/257088-200.png

## Datenquell-Metriken

- Maximale Anzahl an Verbindungen (*datasource.xxx.max*)
- Minimale Anzahl an Verbindungen (*datasource.xxx.min*)
- Anzahl aktiver Verbindungen (*datasource.xxx.active*)

## Cache-Metriken

- Aktuelle Cache-Größe (*cache.xxx.size*)
- Trefferrate (*cache.xxx.hit.ratio*)
- Verlustrate (*cache.xxx.miss.ratio*)

# Realisierung durch Spring Boot Actuator

> An actuator is a manufacturing term, referring to a mechanical device for moving or controlling something. Actuators can generate a large amount of motion from a small change.
>
> — Spring Boot Documentation

**gauge:** (Pegel) Erfassung eines einzelnen Wertes
**counter:** (Zähler) Erfassung einer Zu- oder Abnahme (in-/decrement)

# Endpoints

- ermöglichen das Abfangen der Werte einer Metrik
- bspw. */health*, */metrics*, */info*
- Erstellen eigener Endpoints möglich
- Übergabe an Tools wie *Prometheus* oder *Grafana* mit *MetricsRepository* für weitere Analysen und Darstellungen (Histogramme, etc.)

https://www.iconexperience.com/_img/o_collection_png/green_dark_grey/512x512/plain/barrier_open.png

# Abrufen der internen Metriken (für Maven)

1. Spring Boot Actuator in **pom.xml** (Backend) einbinden:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

2. Spring Boot Starter Security in **pom.xml** einbinden:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

# Abrufen der internen Metriken (für Maven)

3. Anpassen der Endpoints durch Definition von Port und IP-Adresse in **application.properties**:

```
management.port=8090
management.address=127.0.0.1 //localhost
```

4. Anpassen der Zugriffsrechte in **application.properties**:

```
security.basic.enabled=false
security.user.name=admin_josef
security.user.password=josefs_password
```

https://www.iconexperience.com/_img/g_collection_png/standard/256x256/lock.png

# Abrufen der internen Metriken (für Maven)

## 5. Ausgabe am *metrics*-Endpoint: **localhost:8090/metrics**


```
{
  "mem":522347,
  "mem.free":157749,
"processors":4,
"instance.uptime":167882,
"uptime":182609,
"classes":10327,
"classes.loaded":10327,
"classes.unloaded":0,
…
"httpsessions.active":0,
…
"datasource.primary.active":0,
"datasource.primary.usage":0.0

}
```


Systemmetriken

Tomcat Session-Metrik

Datenquell-Metriken

# Eigene Metrik mit CounterService

```java
@Controller
public class CounterController {

    @Autowired
    private CounterService counterService;

    @RequestMapping(value = "/decrement")
    @ResponseBody
    public String decrement() {
        counterService.decrement("important.counter");
        return "Decremented";
    }

    @RequestMapping(value = "/increment")
    @ResponseBody
    public String increment() {
        counterService.increment("important.counter");
        return "Incremented";
    }
}
```

# Eigene Metrik mit CounterService

```
{
  "mem":522347,
  "mem.free":157749,
  "processors":4,
  "instance.uptime":167882,
  "uptime":182609,
  "classes":10327,
  "classes.loaded":10327,
  "classes.unloaded":0,
  …
  "httpsessions.active":0,
  …
  "datasource.primary.active":0,
  "datasource.primary.usage":0.0
  …

  "counter.status.200.increment":5,
  "counter.status.200.decrement":3,
  "counter.important.counter":2,

  }
```

Systemmetriken

Tomcat Session-Metrik

Datenquell-Metriken

Eigene Metrik

# METRIKEN IN SPRING BOOT

- einfacher Zugriff auf integrierte Metriken
- einfaches Erstellen eigener Metriken
- Spring Boot Security ermöglicht kontrollierten Zugriff
- sehr performant auf Java >= 8