# Workshop:
# Spring REST + Architektur

# Übungen

## Ablauf

Dieser Workshop gliedert sich in mehrere Vortragsteile, die den Teilnehmern die Thematik Spring Framework und Spring Boot mit Spring Data usw. näherbringen. Im Anschluss daran sind jeweils einige Übungsaufgaben von den Teilnehmern – idealerweise in Gruppenarbeit – am Rechner zu lösen.

## Voraussetzungen

1) Aktuelles JDK 11, idealerweise auch JDK 17, installiert
2) Aktuelles Eclipse installiert (Alternativ: Spring Tool Suite oder IntelliJ IDEA)

## Teilnehmer

- Entwickler und Architekten mit Java-Erfahrung, die ihre Kenntnisse zu Spring, Spring Boot und Data vertiefen möchten

## Kursleitung und Kontakt


**Michael Inden**
**Derzeit freiberuflicher Buchautor und Trainer**
**E-Mail:** michael.inden@hotmail.com
**Blog:** https://jaxenter.de/author/minden


**Weitere Kurse (Java, Unit Testing, ...) biete ich gerne auf Anfrage als Inhouse-Schulung an.**

# TAG 3: Spring REST + Architektur

## Aufgabe 1: Spring REST Intro 15 min

Vollziehe das Beispiel unter **https://spring.io/guides/gs/rest-service/** nach.

## Aufgabe 2: Product Client 25 min

In der Präsentation haben wir einen Product-Server gesehen. Nun gilt es einen dazugehörigen REST Client zu entwickeln, um Produkte abzufragen und zu erzeugen:

- Erstelle Abfragen als SpringBootApp und als StandAloneApplikation
- Ergänze in der ursprünglichen Applikation noch Validierung. Was müsste man dazu ändern?

## Aufgabe 3: Highscore REST Server 45 min

Erstelle einen Highscore-Server zur Verwaltung und Abfrage von Highscore-Daten. Speichere die Highscores in einer In-Memory-DB. Stelle etwa folgende Endpoints bereit:

- **POST „Highscore“**
- **GET /highscores**
- **GET /highscores/top/<n>**
- GET /highscores/byPerson/<name>
- GET /highscores/byLocalDate/<localdate>
- DELETE /highscores/<n>
- **DELETE /highscores/of/<name>**
- DELETE /highscores/on/< localdate >

## Aufgabe 4: Erstelle einen User REST Server 45 min

Gegeben sei ein REST-Server zur Verwaltung und Abfrage von User-Daten, der ausschließlich mit GET arbeitet. Die User werden in einer In-Memory-DB gespeichert und durch folgende Endpoints bereitgestellt:

- `/create?email=[email]&name=[name]: create a new user with an auto-generated id and email and name as passed values.`
- `/delete?id=[id]: delete the user with the passed id.`
- `/get-by-email?email=[email]: retrieve the id for the user with the given email address.`
- `/update?id=[id]&email=[email]&name=[name]: update the email and the name for the user identified by the given id.`

Verbessere das Applikationsdesign durch

a) Nutzung von passsenden Status-Codes sowie HTTP-Verben
b) Integration eines Service
c) sowie durch Einsatz von Error Handling
d) Validierung

Prüfe die Validierung und die anderen Aktionen mit CURL, etwa

```
curl –X POST –H "Content–Type: application/json"
http://localhost:8080/users/ –d '{ "email" : "E–MAIL", "name" :
"Micha" }'
```

Und mit einer gültigen E-Mail:

```
curl –X POST –H "Content–Type: application/json"
http://localhost:8080/users/ –d '{ "email" : "mail@mike.de", "name"
: "Micha" }'
```