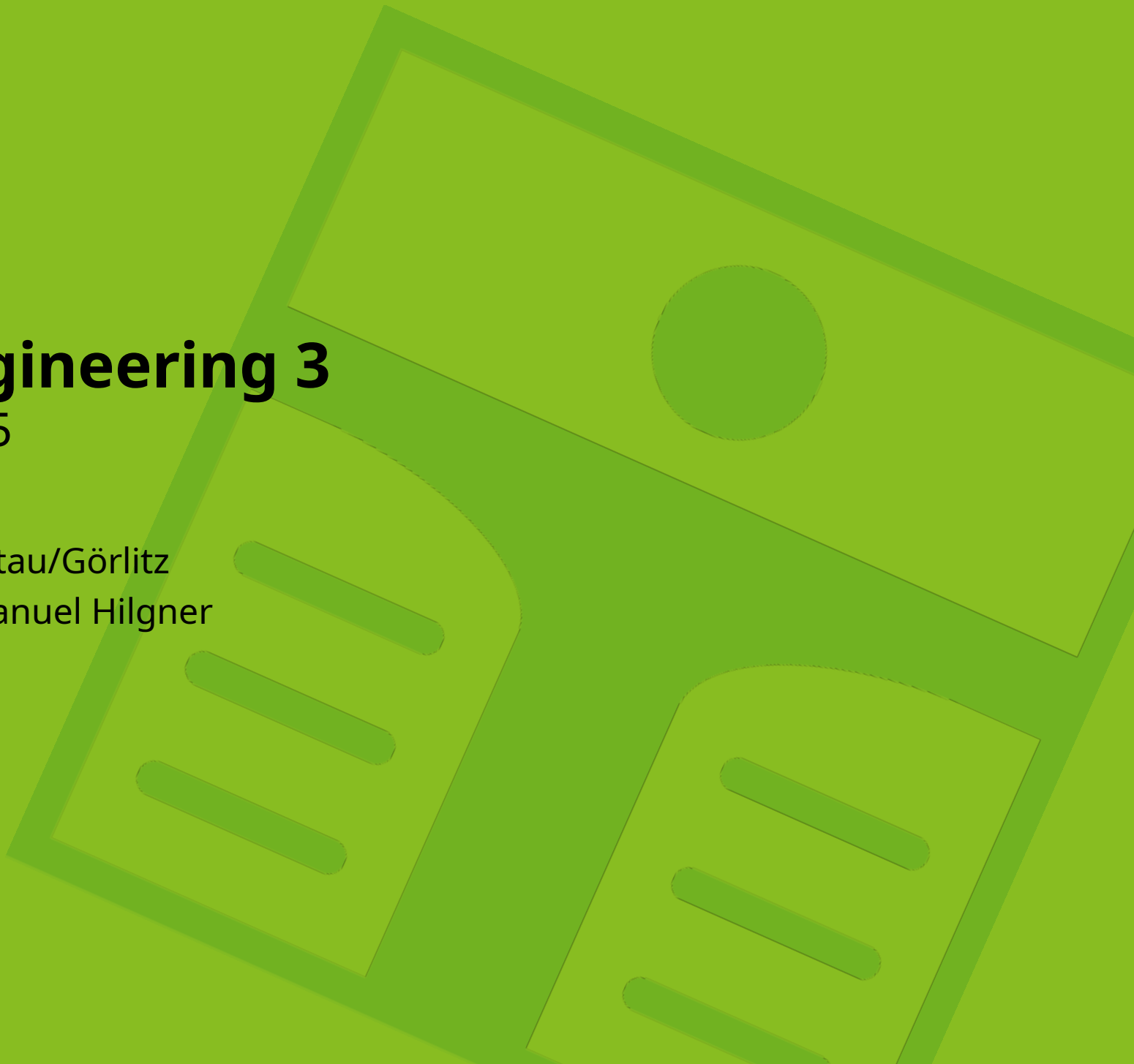
Web Engineering 3

Vorlesung 5

Hochschule Zittau/Görlitz
Christopher-Manuel Hilgner



Agenda

Vorlesung

- Spring Data JPA
- Konfiguration von Spring Anwendungen
- Einführung in Docker

Seminar

- Relationen zwischen Entities
- Konfiguration der Spring Anwendung
- Einfaches Docker
- PostgreSQL

Spring Data JPA

Konfiguration einer Spring Anwendung

Konfiguration einer Spring Anwendung

YAML File

YAML

```yaml
server:
  port: 8080

spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/db
    username: db_user
    password: db_password
  jpa:
    hibernate:
      ddl-auto: create-drop
    show-sql: true
    properties:
      hibernate:
        format_sql: true
```

YAML File

```yaml
server:
  port: 8080
```

- Konfiguration für den Spring server
- `port` bestimmt den Port der Spring Anwendung

YAML File

application.yml YAML

```yaml
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/db
    username: db_user
    password: db_password
```

- Konfiguration des Datenbankzugriffs
- `url` enthält URL zur gewünschten Datenbank.
- `username`: Username der ausgewählten Datenbank
- `password`: Passwort der ausgewählten Datenbank

YAML File

application.yml YAML

```yaml
jpa:
  hibernate:
    ddl-auto: create-drop
  show-sql: true
  properties:
    hibernate:
      format_sql: true
```

- `ddl-auto`: Bestimmt, welche Operationen automatisch auf der Datenbank ausgeführt werden sollen
  - `create-drop` erschafft immer eine neue Datenbank, wenn die Spring Anwendung gestartet wird. Die Alte wird dabei gelöscht.
- **Wichtig**: Wenn keine embedded Datenbank genutzt wird, werden Datenbanken nicht automatisch erstellt.

YAML File

application.yml YAML

```yaml
jpa:
  hibernate:
    ddl-auto: create-drop
    show-sql: true
  properties:
    hibernate:
      format_sql: true
```

- `show-sql`: Ausgeführte SQL Befehle werden in der Konsole ausgegeben. Macht Debugging einfacher.

YAML File

application.yml YAML

```yaml
jpa:
  hibernate:
    ddl-auto: create-drop
  show-sql: true
  properties:
    hibernate:
      format_sql: true
```

- `format_sql`: Formatiert den SQL Output in eine lesbarere Form

Propterties File

application.properties properties

```properties
server.port=8080
spring.datasource.url=jdbc:postgresql://localhost:5432/db
spring.datasource.username=db_user
spring.datasource.password=db_password
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Einführung in Docker

Dockerfile

Compose

YAML

```yaml
services:
  db:
    image: postgres
    container_name: postgres_db_container
    environment:
      - POSTGRES_USER=db_user
      - POSTGRES_PASSWORD=db_password
      - POSTGRES_DB=db
    networks:
      - application_network
    ports:
      - "8001:5432"
networks:
  application_network:
    driver: bridge
```

Compose

Services

```yaml
services:
  db:
    # Service Definition
  # Weitere Services
```

- In einer Map, unter dem Key `services`, werden alle Services definiert, die von der Compose gestartet werden sollen
- Name wird durch Key bestimmt
- Hier wäre der Name des Services: `db`

Compose

Services

```yaml
db:
  image: postgres
  container_name: postgres_db_container
  environment:
    - POSTGRES_USER=db_user
    - POSTGRES_PASSWORD=db_password
    - POSTGRES_DB=db
  networks:
    - application_network
  ports:
    - "8001:5432"
```

- Einzelne Services werden durch Map unter dem Service-Key definiert

Compose

Services

```yaml
services:
  db:
    image: postgres
```

- Key `image` bestimmt das Container-Image, dass hier genutzt werden soll
- Können vom System kommen oder zum Beispiel aus der Docker Registry
- Hier wird ein Image genutzt, welches für die PostgreSQL Datenbank zugeschnitten ist

https://hub.docker.com/_/postgres

Compose

Services

```yaml
services:
  db:
    container_name: postgres_db_container
```

- Key `container_name` bestimmt den Namen des Containers

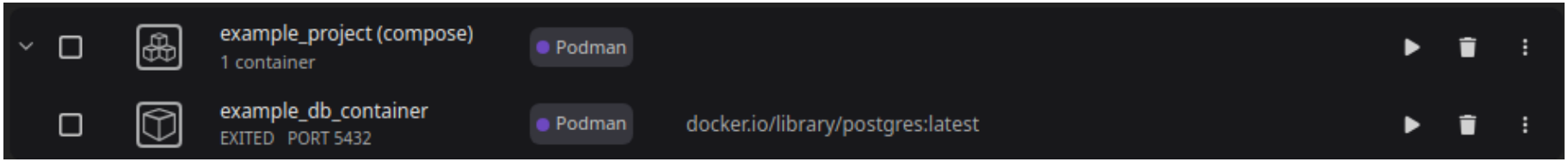


Figure 1: Aktuell nicht laufender Container in der Podman Desktop UI. Der Container hat den Namen: `example_db_container`

Compose

Services - Environment Variablen

compose.yml YAML

```yaml
environment:
  - POSTGRES_USER=db_user
  - POSTGRES_PASSWORD=db_password
  - POSTGRES_DB=db
```

application.yml YAML

```yaml
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/db
    username: db_user
    password: db_password
```

Compose

Services

```yaml
services:
  db:
    networks:
      - application_network
```

- Key `networks` bestimmt, zu welchen Netzwerken der Container zugeordnet werden soll
- Es können dabei mehrere Netzwerke genutzt werden
- Netzwerke werden in einer Map angegeben
- Hier wird nur das `application_network` genutzt

Compose

Services - Ports

```yaml
services:
  db:
    ports:
      - "8001:5432"
```

- Key `ports` bestimmt, welche Ports von dem Container exposed werden sollen
- Es können mehrere Ports in einer Map angegeben werden
- Jede Angabe besteht aus zwei Ports

Compose

Services - Ports

```yaml
1  - "8001:5432"
```

Der linke Port ist der Port, der von außerhalb des Containers erreicht werden kann
Der rechte Port ist der Port, der innerhalb des Containers genutzt werden soll. Er wird auf den linked Port gemapped.

In diesem Beispiel hier, wird der default Port einer PostgreSQL Datenbank, **5432**, der innerhalb des Container genutzt wird, auf den Port **8001** gemapped. Die Datenbank kann somit außerhalb des Containers auf dem Port **8001** erreicht werden.

Compose

Services - Depends On

```yaml
services:
  db:
    depends_on:
      - other_container
```

- Key `depends_on` bestimmt, welche Container bereits laufen müssen, damit der aktuelle Container gestartet werden kann
- Es können mehrere Container in einer Map angegeben werden
- Der Name des Containers ist hier der Service Name

Compose

Netzwerke

```yaml
networks:
  application_network:
    driver: bridge
```

- Definierung von Netzwerken für die Docker Container
- Container, die auf einem gemeinsamen Netzwerk sind, können untereinander kommunizieren
- `bridge` Netzwerk wird am Häufigsten genutzt
- Netzwerke werden als Map unter dem Key `networks` definiert
- Netzwerk-Name ist Key des jeweiligen Netzwerks
- `driver` Key bestimmt, welche Art von Netzwerk genutzt wird

MENSA IST VORBEI