Hilfreiche Informationen ☺

- Listen initialisieren (LinkedList)
- JsonBTransient

DB - Connection

Wenn des Bild ned geht hast a Pech ghabt

Panache - Entity

- PanacheEntityBase: Bei eigener ID (extends)
- PanacheEntity: Bei generated ID (extends)
- implements Serializible
- @Id --> @GeneratedValue --> Long
- @Column, @JoinColumn, @JoinTable (Attribute wie name, length, ... vergeben)
- @OneToMany, @ManyToOne, @ManyToMany --> mappedBy
- @NamedQuery(name = Klasse.Name, query = "...")

Repository

- ApplicationScoped
- implements PanacheRepository<Type>
- @Transactional
- Panache.getEntityManager() --> merge, persist, ...
- Panache-Methoden, wie find, findById, findAll, --> .list() nicht vergessen

```
TypedQuery<Genre> query = Panache.getEntityManager().createQuery( "...",
Genre.class);

Panache.getEntityManager().createNamedQuery(Genre.FINDALL,
Genre.class).getResultList();

query.getResultList().stream().map(entity -> new EntityDTO(entity.bezeichnung,
entity.id)).collect(Collectors.toList());
```

Boundary

- ApplicationScoped
- Path(...) --> Klasse und Methoden
- @GET, @POST, ...

- @Inject
- Response.ok().build()

Achja, DTO kommt übrigens auch 😄 🙂

LG

EmbeddedID

Entity

```java
@Entity()
public class Entity {
    @EmbeddedId
    EntityEmbeddedId id;
    ...
```

EntityEmbeddedId

```java
@Embeddable
public class EntityEmbeddedId implements Serializable {

    //Kann aus ids oder ganzen Entities bestehen, wie zB:
    @ManyToOne
    private Hall hall;

    private int genreId;


    // btw vergessts in hashcode und des equals ned haha
}
```

IdClass

Entity

```java
@Entity
@IdClass(ProjectId.class)
public class Project {
    @Id
    private int departmentId;
    @Id
    private int projectNo;
    ...
}
```

ProjectId

```java
public class ProjectId implements Serializable {
    private int departmentId;
    private int projectNo;

    //Konstruktoren, equals, hashcode
}
```

Socket

```java
@ServerEndpoint("/...")
@ApplicationScoped
public class Socket {
    Set<Session> sessions = new ConcurrentHashSet<>();
}
```

Events:

- onOpen
- onMessage
- onError
- onClose

```java
@OnOpen
public void onOpen(Session session) {
    sessions.add(session);
    send(...);
}

@OnClose
public void onClose(Session session) {
    sessions.remove(session);
}

@OnError
public void onError(Session session, Throwable throwable) {
    sessions.remove(session);
}


@OnMessage
```

```
public void onMessage(String message, @PathParam("username") String username) {
    send()
}

public void send(Session session, Survey survey) {
    session.getAsyncRemote().sendObject(surv, sendResult -> {
    });

}
```

Hilfreiche SQL-Methoden

- null value handling = COALESCE(value, default)
- Cast datatypes = cast(ivd.trnDatetime as LocalDate) (LocalDateTime to LocalDate)
- eliminate duplicates = select DISTINCT c from...
- count, sum, avg, max, min,
- Document AS d WHERE :kauf MEMBER OF b.kaeufe
- SELECT new htl.model.Kauf()

Datumsgschichtl

```
LocalDateTime.parse(
    dateString,
    DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss")
)
```

```
LocalDateTime jetzt = LocalDateTime.now();
LocalDate datum = jetzt.toLocalDate();
```

```
String s = neich.format(DateTimeFormatter.ofPattern("yyyy.MM.dd"));
```