Cheatsheet NVS 4AHIF 2021

1. Entity

- @Entity
- @Table(name = "table_name")
- @Id
  - sollte Long sein
- @GeneratedValue(strategy = GenerationType.IDENTITY)
- @JoinColumn(name = "ROOM_NO")
  - Für Foreign-Key-Spalte in der Tabelle der referenzierenden Entität
- @Column(name = "DATE_TO")

1.1. String.format

```
@Override
public String toString() {
return String.format(...);
}
```

String → %s
Decimal integer → %d
Date → %tF

1.2. Unique Constraints

```
@Table(name = "H_CUSTOMER",uniqueConstraints = @UniqueConstraint(columnNames={
"FIRST_NAME", "LAST_NAME"}))
```

2. Control

2.1. ORM (Panache)

Ohne dem kein Panache. **@ApplicationScoped** nicht vergessen!

```
@ApplicationScoped
public class Repository implements Repository<Entity>{
    ...
}
```

```
PanacheRepositoryBase<Room, Integer> // Integer = ID
```

2.2. Insert Data

```
@Transactional
@PostConstruct
public void insertData(){
    ...
}
```

@Transactional immer verwenden bei änderungen in der DB.

2.3. NamedQuery

```
@NamedQueries({
        @NamedQuery(name = "Person.findByFirstName", query = "select p from Person p
where firstName = :firstName ")
})
```

```
return (Customer)getEntityManager()
.createNamedQuery("Customer.findByFirstNameAndLastName") // name der NamedQuery
.setParameter("firstName", firstName) // parameter :firstName
.getSingleResult();
```

2.4. Logger

```
@Inject
Logger log;
log.info("...");
```

2.4.1. Logger Producer

```
public class LoggerProducer {

    @Produces
    public Logger produceLogger(InjectionPoint injectionPoint) {
        return Logger.getLogger(injectionPoint.getBean().getBeanClass());
    }
}
```

2.5. Unmodifiable List

```
Collections.unmodifiableList(...)
```

2.6. Read CSV

```
List<String> readFile(String fileName) {
        URL url = Thread.currentThread().getContextClassLoader().getResource(fileName
);
        assert url != null;
        try (Stream<String> stream = Files.lines(Paths.get(url.getPath()),
StandardCharsets.UTF_8)) {
            return stream
                    .skip(1)
                    .distinct()
                    .map(line -> {
                        if (line.length() <= 2) {
                            return line + " - 1";
                        } else {
                            return line + " - 2";
                        }
                    })
                    .collect(Collectors.toList());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
```

3. Boundary/Service

- @RequestScoped
- @Path("/endpoint")

3.1. Inject Repository

```
@Inject
Repository repository;
```

3.2. UriInfo

@Context UriInfo info

```
UriBuilder uriBuilder = info
.getAbsolutePathBuilder()
.path(Long.toString(person.getId()));
return Response.created(uriBuilder.build()).build();
```

```
 return Response.status(400).header("reason", "out of range ").build();
```

3.3. Params

- @PathParam
  - @Path("{id}")
- @QueryParam("name")
  - @Path("/name")

4. JAX-RS @FormParam example (HTML-Formular)

4.1. HTML Form

simple HTML form with "post" method

```
<html>
<body>
<h1>JAX-RS @FormQuery Testing</h1>

    <form action="rest/user/add" method="post">
        <p>
            Name : <input type="text" name="name" />
        </p>
        <p>
            Age : <input type="text" name="age" />
        </p>
        <input type="submit" value="Add User" />
    </form>

</body>
</html>
```

4.2. @FormParam Example

Example to use @FormParam to get above HTML form parameter values.

```
import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

@Path("/user")
public class UserService {

    @POST
    @Path("/add")
    public Response addUser(
        @FormParam("name") String name,
        @FormParam("age") int age) {

        return Response.status(200)
            .entity("addUser is called, name : " + name + ", age : " + age)
            .build();

    }

}
```

5. I18N

I18N Beispiel

```
String language;
String country;

if (args.length != 2) {
    language = new String("en");
    country = new String("US");
} else {
    language = new String(args[0]);
    country = new String(args[1]);
}

Locale currentLocale;
ResourceBundle messages;

currentLocale = new Locale(language, country);

messages = ResourceBundle.getBundle("at.htl.MessagesBundle", currentLocale);

System.out.println("Writing messages for " + currentLocale.toLanguageTag());
System.out.println(messages.getString("greetings"));
System.out.println(messages.getString("inquiry"));
System.out.println(messages.getString("farewell"));
```

properies

```
greetings = Hallo.
farewell = Tschüß.
inquiry = Wie gehts?
```

6. Marshalling und Unmarshalling JSON

```
@JsonSerialize(using = LocalDateSerializer.class)
@JsonDeserialize(using = LocalDateDeserializer.class)
@Column(name = "DATE_SIGNED")
private LocalDate contractSigned;

    @JsonSerialize(using = LocalDateSerializer.class)
    @JsonDeserialize(using = LocalDateDeserializer.class)
    @Column(name = "DATE_END")
    private LocalDate contractEnd;
```

```
public class LocalDateDeserializer extends JsonDeserializer<LocalDate> {
DateTimeFormatter df = DateTimeFormatter.ofPattern("yyyy-MM-dd");

    @Override
    public LocalDate deserialize(JsonParser arg0, DeserializationContext arg1) throws
IOException {
        return LocalDate.parse(arg0.getText(), df);
    }
}
```

```
public class LocalDateSerializer extends JsonSerializer<LocalDate> {
    @Override
    public void serialize(LocalDate arg0, JsonGenerator arg1, SerializerProvider arg2)
throws IOException {
        arg1.writeString(arg0.toString());
    }
}
```

6.1. JSON P

```
JsonObjectBuilder classroomBuilder = Json.createObjectBuilder();

classroomBuilder.add("klasse", "4ahif");
classroomBuilder.add("raum", "107");

JsonObject classroom = classroomBuilder.build();
```