

Übung "Pet Obedience School" aka DogSchool (Hundeschule)

Table of Contents

Ausgangssituation	1
Aufgabe 1: Initialisieren der Datenbank	1
Aufgabe 2: Erstellen der Repositories	2
Aufgabe 3: RESTful Endpoint für CourseType	3
Liste aller Kurstypen	3
Erstellen eines neuen Kurstypen zB Schutzhunde-Ausbildung (mit Abkürzung "schutz")	3
Ändern eines Kurstypen	3
Löschen eines Kurstypen	5
Aufgabe 4: Erstellen von drei Buchungen	6

Version: 1.0

Ausgangssituation

Ein Besitzer einer Hundeschule tritt an sie als Softwareentwickler heran, und bittet Sie eine Anwendung in Jakarta EE für folgende Situation zu erstellen:

- Die Hundeschule bietet grundsätzlich Kurse für Hunde an
- Es gibt verschiedene Kurstypen (zB Welpenkurs und Begleithunde1)
- Hunde gehören zu Personen
- Bei den Kursbuchungen werden zwar die Hunde eingetragen, über diese sind jedoch die Eigentümer (owner) zu eruieren.

Aufgabe 1: Initialisieren der Datenbank

1. Ergänzen Sie die Entitäten so, dass eine Persistierung in der Datenbank erfolgt.
2. Ergänzen Sie ebenfalls die InitBean so, dass beim Start des Systems eine Initialisierung der Datenbank erfolgt
3. Beachten Sie beim Einlesen der Kurs-csv-Datei, dass keine doppelten Kurse eingelesen werden (siehe Kommentar bei Methode InitBean.readCsv(...))

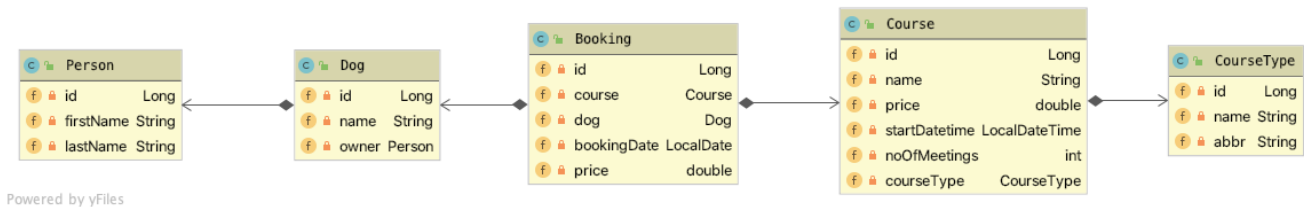


Figure 1. Class Diagram

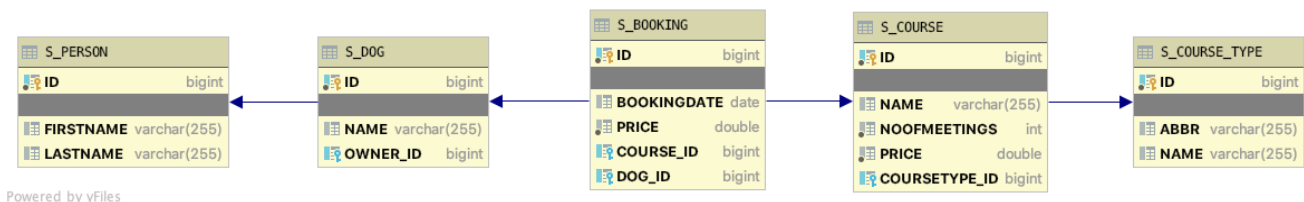


Figure 2. Entity Relationship Diagram



Beachten sie die Kommentare im source-code. Dort finden Sie weitere Anweisungen und Hilfestellungen

	ID	FIRSTNAME	LASTNAME
1	1	Matt	Murdock
2	2	Mathilda	Lando

Figure 3. Table S_PERSON

	ID	NAME	OWNER_ID
1	1	Timmy	1
2	2	Tino	1
3	3	Arko	2
4	4	Rex	2
5	5	Edi	2

Figure 4. Table S_DOG

	ID	ABBR	NAME
1	1	w	Welpenkurs
2	2	bg1	Begleithunde1
3	3	bg2	Begleithunde2

Figure 5. Table S_COURSE_TYPE

	ID	NAME	NOOFMEETINGS	PRICE	STARTDATETIME	COURSETYPE_ID
1	1	BG1 - Frühlingskurs	8	80	2020-03-07 10:00:00.000000000	2
2	2	BG1 - Herbstkurs	8	80	2019-08-16 10:00:00.000000000	2
3	3	Welpen - Frühlingskurs	8	70	2019-03-07 14:00:00.000000000	1
4	4	Welpen - Herbstkurs	8	70	2019-08-16 14:00:00.000000000	1
5	5	BG2 - Frühlingskurs	8	85	2020-05-09 10:00:00.000000000	3
6	6	BG2 - Herbstkurs	8	85	2019-10-18 10:00:00.000000000	3

Figure 6. Table S_COURSE

Aufgabe 2: Erstellen der Repositories

Sämtliche Zugriffe auf die Datenbank (mit Ausnahme der InitBean) sind über die Repositories zu

erfolgen.

Sie müssen **nicht** alle Methoden in den Repositories implementieren - nur jene, die sie zur Versorgung zB der Endpoints mit Daten benötigen.

Aufgabe 3: RESTful Endpoint für CourseType

Erstellen Sie einen Endpoint `CourseTypeEndpoint` für die Entität `CourseType` und implementieren Sie dort eine CRUD-Funktionalität. Das Datenformat ist sowohl bei den Requests als auch bei den Responses jeweils JSON.



In der Datei `/rest-requests/requests.http` sind schon einige Requests vorbereitet

Liste aller Kurstypen

```
GET http://localhost:8080/school/api/course_type
```

Erstellen eines neuen Kurstypen zB Schutzhunde-Ausbildung (mit Abkürzung "schutz")

```
POST http://localhost:8080/school/api/course_type
```

Wenn sie den vorbereiteten Request im File `/rest-requests/requests.http` verwenden, sollten Sie folgenden Response erhalten:

```
POST http://localhost:8080/school/api/course_type

HTTP/1.1 201 Created
Connection: keep-alive
Location: http://localhost:8080/school/api/course_type/4
Content-Length: 0
Date: Wed, 11 Dec 2019 23:59:32 GMT

<Response body is empty>

Response code: 201 (Created); Time: 7780ms; Content length: 0 bytes
```

Wichtig ist dabei der Header "Location" mit der id des erstellten Datensatzes

Ändern eines Kurstypen

Korrekte Anfrage

```
PUT http://localhost:8080/school/api/course_type/4
Content-Type: application/json
```

```
{
  "abbr": "such",
  "name": "Suchhunde-Ausbildung"
}
```

ev. auch PATCH

vgl. hierzu <https://medium.com/backticks-tildes/restful-api-design-put-vs-patch-4a061aa3ed0b>

Response bei korrekter Anfrage

```
PUT http://localhost:8080/school/api/course_type/4
```

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: application/json
Content-Length: 52
Date: Thu, 12 Dec 2019 00:42:47 GMT
```

```
{
  "abbr": "such",
  "id": 4,
  "name": "Suchhunde-Ausbildung"
}
```

```
Response code: 200 (OK); Time: 5396ms; Content length: 52 bytes
```

Fehlerhafte Anfrage

```
PUT http://localhost:8080/school/api/course_type/10
Content-Type: application/json
```

```
{
  "abbr": "such",
  "name": "Suchhunde-Ausbildung"
}
```

```
PUT http://localhost:8080/school/api/course_type/10
```

```
HTTP/1.1 400 Bad Request
```

```
Connection: keep-alive
```

```
Reason: courseType with id 10 does not exist
```

```
Content-Length: 0
```

```
Date: Thu, 12 Dec 2019 00:30:25 GMT
```

```
<Response body is empty>
```

```
Response code: 400 (Bad Request); Time: 29ms; Content length: 0 bytes
```

Beachten Sie, den Header **Reason**. Implementieren sie diesen genauso

Löschen eines Kurstypen

Korrekte Anfrage

```
DELETE http://localhost:8080/school/api/course_type/4
```

```
HTTP/1.1 204 No Content
```

```
Date: Thu, 12 Dec 2019 00:39:27 GMT
```

```
<Response body is empty>
```

```
Response code: 204 (No Content); Time: 3724ms; Content length: 0 bytes
```

Fehlerhafte Anfrage

```
DELETE http://localhost:8080/school/api/course_type/10
```

```
HTTP/1.1 400 Bad Request
```

```
Connection: keep-alive
```

```
Reason: courseType with id 10 does not exist
```

```
Content-Length: 0
```

```
Date: Thu, 12 Dec 2019 00:37:19 GMT
```

```
<Response body is empty>
```

```
Response code: 400 (Bad Request); Time: 24808ms; Content length: 0 bytes
```



Falls Sie Informationen über JPA-Queries brauchen, können Sie folgende Quelle verwenden <https://www.objectdb.com/java/jpa/query/jpql/expression>

Aufgabe 4: Erstellen von drei Buchungen

Hier können Sie die Methodennamen der Endpoints frei wählen

Viel Erfolg!