Hochschule RheinMain
H. Arega, W. Weitz

Studiengang Medieninformatik
Sommersemester 2024

Webbasierte Anwendungen

Web-Ueb-05 (22.05.2024)

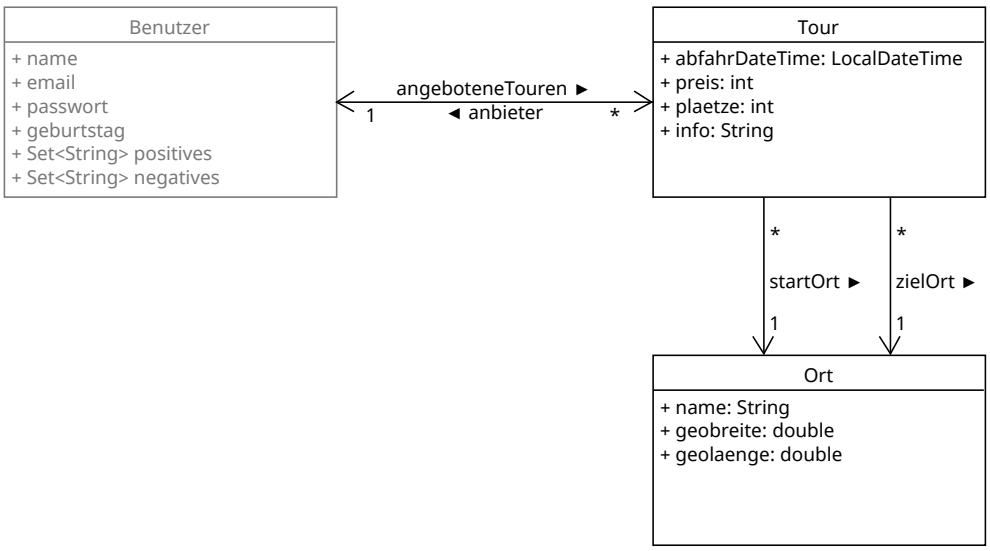
Unsere MIfahrzentrale kann bisher nur `Benutzer` verwalten. Das ist schön, genügt aber auf den zweiten Blick nicht, denn letztlich wollen wir auch Mitfahrmöglichkeiten (Touren) anbieten. Jede Tour startet zu einem bestimmten Abfahr-Zeitpunkt (Datum und Uhrzeit), hat eine begrenzte Zahl von Plätzen für Mitfahrer und einen Preis. Wenn gewünscht, kann der Anbieter auch einen Info-String für spätere Interessenten hinterlegen.

Jeder Benutzer kann beliebig viele Touren anbieten, jede Tour hat einen eindeutigen Anbieter (schon aus rechtlichen Gründen). Eine Tour hat einen Start- und einen Ziel-Ort, deren Geo-Koordinaten bekannt sind (geogr. Breite und Länge). Wir betrachten zunächst einmal nur die "Angebots-Seite", Such- und Buchungsmöglichkeitencd /Volumes/tank/wwe für Mitfahrer usw. kommen später dazu.

Die Übersetzungsdatei `uebersetzungen.csv` wurde im read.MI aktualisiert, bitte bieten Sie auch die neuen Seiten (mindestens) in den drei dort hinterlegten Sprachen an.

Aufgabe 1 (Entitäten und Repositories: Tour und Ort)

Bitte legen Sie im `entities`-Unterpaket Ihres Projekts wie schon beim `Benutzer` geübt ein Unterpaket `ort` mit Entität `Ort` und zugehörigem Repository `OrtRepository` sowie ein Unterpaket `tour` mit Entität `Tour` und zugehörigem Repository `TourRepository` an, und statten Sie die Entitäten gemäß dem nachfolgenden Klassendiagramm mit Attributen und Beziehungen aus. Verwenden Sie bitte die vorgegebenen Namen (auch für die Beziehungs-Attribute; Ausnahme sind die mengenwertigen Attribute in `Benutzer` für gemochte oder ungemochte Sachen, falls Sie diese anders benannt haben, ist das ok). Die Pfeile an den Beziehungen drücken die benötigten Navigationsrichtungen aus, so ist die Beziehung zwischen `Benutzer` und seinen `Tour`en bidirektional, während man von einer `Tour` an deren Start- und Ziel-Ort kommt, aber nicht vom `Ort` an die referenzierenden `Tour`en.



Alle Entitäten haben ein `id`- und ein `version`-Attribut vom Typ `long` (nicht im Diagramm enthalten).

Validierungsbedingungen:

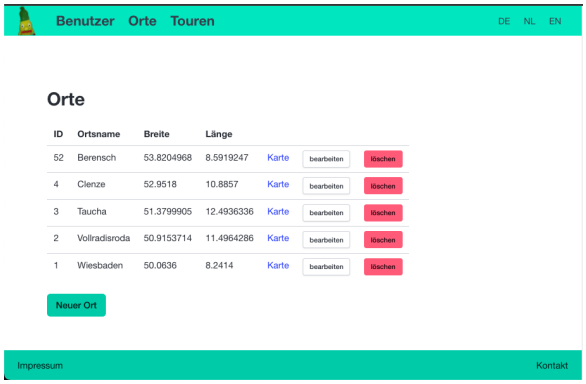
Bei der `Ort`-Entität ist zu beachten, dass der `name` ausgefüllt sein muss (also nicht NULL und mindestens ein Nicht-Leerzeichen).

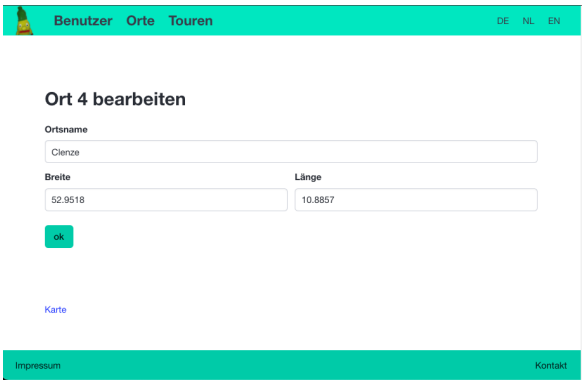
Bei einer `Tour` darf der Preis nicht negativ werden (Gratis-Reisen sind also möglich), es muss mindestens ein Platz für Mitfahrer angeboten werden und sowohl Start- als auch Ziel-Ort müssen angegeben sein. Der `info`-Text für die Fahrgäste darf bis zu 400 Zeichen lang sein.

Aufgabe 2 (Orte bearbeiten)

Wie beim `Benutzer` schon praktiziert möchten wir nun eine Möglichkeit zum Anlegen, Bearbeiten und Löschen von `Orte`n haben:

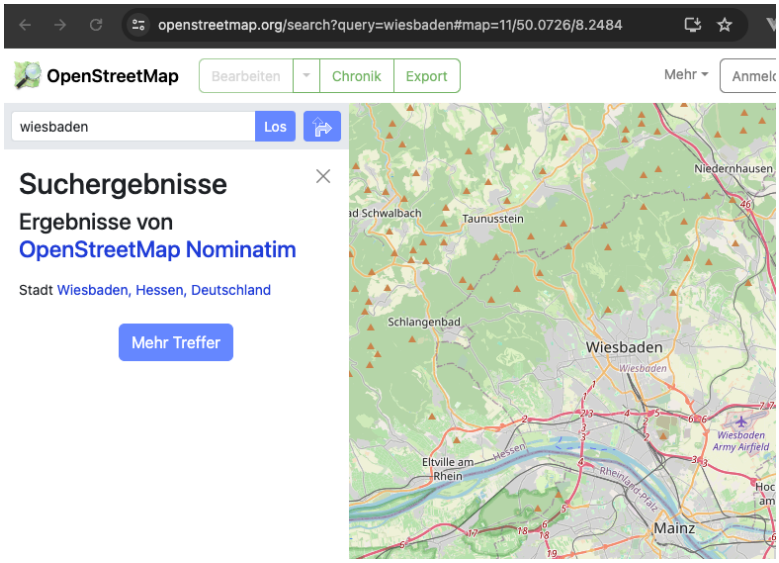
- Bitte erschaffen Sie im vorhandenen `ui`-Package ein Unterpaket `ort` mit einem `OrtController`, der für die Formulardaten auf ein `OrtFormular` im selben Package zurückgreift.
- Im `templates`-Unterordner `ort` legen wir Thymeleaf-Templates `ortliste.html` und `ortbearbeiten.html` an, welche die Orte mit ihren Attributen auflisten bzw. bearbeitbar machen (s. Screenshots). Wie üblich gibt es für jeden Eintrag der Orts-Liste einen "bearbeiten" und einen "löschen"-Link. In der Ortsliste sollen die Einträge nach Namen aufsteigend sortiert erscheinen.
- Der Controller sorgt dafür, dass unter dem URI-Pfad `/ort` die Liste aller Orte und unter `/ort/{id}` die Bearbeitungsseite für den Ort mit der ID `id` zugänglich ist.
- Auch legen wir im Thymeleaf-Fragment für die Kopfzeile einen Link "Orte" auf den Pfad `/ort` an.
- Im vorhandenen `services`-Package legen Sie bitte ein Unterpackage `ort` mit Interface `OrtService` und zugehöriger Klasse `OrtServiceImpl` an, welche die fachliche Verarbeitung für ihren Controller übernimmt.





Wie bei der zugehörigen `Ort`-Entität muss im Formular der Ortsname ausgefüllt sein (die Geo-Angaben sind defaultmäßig 0.0, können aber natürlich geänder werden).

Zur Ermittlung der Koordinaten eines gewünschten Ortes kann z.B. OpenStreetMap `https://openstreetmap.org` benutzt werden.



Die Koordinaten sind dann auf der Seite bzw. oben im Link ablesbar (hier: (50.0726, 8.2484)).

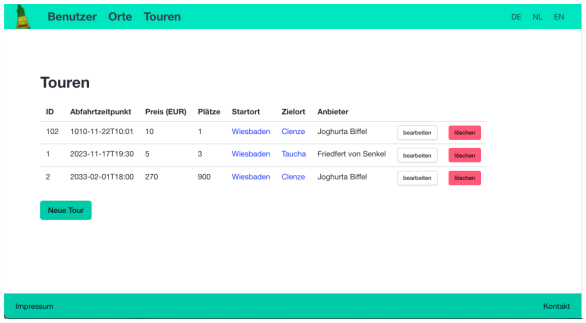
In der Ortsliste sehen Sie in jeder Zeile vor den Bearbeitungs-/Lösch-Links einen Link “Karte”, der (in einem separaten Browser-Tab) die OpenStreetMap-Kartenansicht für den betreffenden Ort öffnet. Wenn man die Koordinaten hat, braucht man dazu in `ortliste.html` dazu je Ort-Objekt `o` nur einen Link dieser Gestalt zu erzeugen:

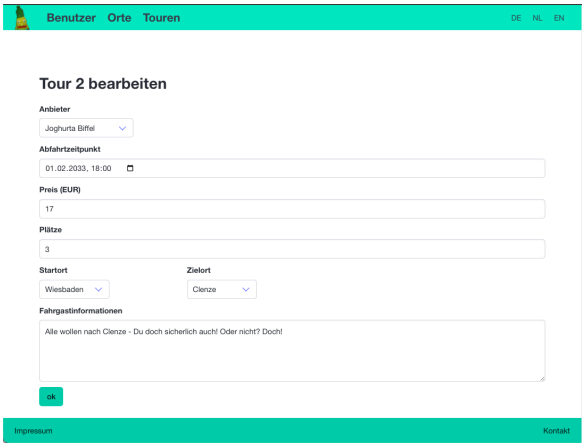
```
<a
 th:href="|https://www.openstreetmap.org/#map=14/${o.geobreite}/${o.geolaenge}|"
 target="_blank">Karte</a>
```

Aufgabe 3 (Touren)

Nun fehlt noch die Bearbeitungsmöglichkeit für `Tour`en. Analog zu eben wird angelegt:

- ...im vorhandenen `ui`-Package ein Unterpaket `tour` mit einem `TourController`, der für die Formulardaten auf ein `TourFormular` im selben Package zurückgreift.
- Im `templates`-Unterordner `tour` legen wir Thymeleaf-Templates `tourliste.html` und `tourbearbeiten.html` an, welche die Tour mit ihren Attributen auflisten bzw. bearbeitbar machen (s. Screenshots). Wie üblich gibt es für jeden Eintrag der Tour-Liste einen “bearbeiten” und einen “löschen”-Link. In der Tour-Liste sollen die Einträge zeitlich aufsteigend sortiert sein.
- Der Controller sorgt dafür, dass unter dem URI-Pfad `/tour` die Liste aller Touren und unter `/tour/{id}` die Bearbeitungsseite für die Tour mit der ID `id` zugänglich ist.
- Auch legen wir im Thymeleaf-Fragment für die Kopfzeile einen Link “Touren” auf den Pfad `/tour`.
- Im vorhandenen `services`-Package wächst ein Unterpackage `tour` mit Interface `TourService` und zugehöriger Klasse `TourServiceImpl` nach, welche die fachliche Verarbeitung für ihren Controller übernimmt.





Auf der Bearbeitungsseite werden abgefragt:

- der Anbieter der Tour (Select-Box, gefüllt aus der Liste der in der Datenbank vorhandenen `Benutzer`),
- der Abfahrtzeitpunkt (Java `LocalDateTime`, HTML `INPUT`-Feldtyp `datetime-local`),
- der Preis (≥ 0) und die Anzahl angebotener Plätze (> 0),
- Start- und Ziel-Ort (Select-Boxen, gefüllt aus den in der Datenbank verfügbaren `Ort`-Einträge)
- `info`-String, max. 400 Zeichen.

Der Benutzer muss eine Auswahl aus allen drei Select-Boxen (Anbieter der Tour, Start- und Zielort) treffen. Um einen "ungültigen" Startwert zu haben kann man z.B. im Formular-Objekt für die betreffenden Wertelisten für Benutzer, Start- und Zielort (also die Datenquellen für die zugehörigen Select-Boxen) am Anfang einen "Pseudoeintrag" mit dem Namen "–" anbringen, dessen Auswahl beim Abschicken des Formulars als ungültig interpretiert wird. Wenn man als Auswahl-Wert der Select-Box die `id` zur dargestellten Entity (z.B. hier `Benutzer`) verwendet, kann man die Validierung leicht dadurch erreichen, indem man für den Auswahlwert der Selectbox `@Positive` fordert und dem Pseudoeintrag als `id`-Wert z.B. die 0 gibt.

Hinweis: Die `POST`-Handlermethode für die Tour ist vermutlich die bisher interessanteste, weil hier drei Beziehungen zu pflegen sind (zu `Benutzer` als Anbieter der Tour und zweimal zu `Ort`). Sicherlich sollen die zugehörigen JPA-Operationen zum Anlegen/Aktualisieren eines `Benutzer`-Objekts in einer Transaktion stattfinden. Es ist daher sinnvoll, in der `@Controller`-Methode nur die Daten einzusammeln und den Auftrag zum Anlegen/Updaten der `Tour` in eine transaktionale `@Service`-Methode auszulagern.

Controller sollten eigentlich nur die zu verarbeitenden Daten aus der Benutzereingabe bündeln und dann an *eine* Service-Methode weiterreichen (also *nicht* mehrere DB-Zugriffe zum Zusammenbau der aktualisierten `Tour` incl. der referenzierten anderen Entitäten im Controller anstoßen, um die "fertige" `Tour` an den Service zur reinen Abspeicherung weiterzugeben).

Dass man überhaupt JPA-Operationen im Spring-Controller anstoßen kann (und dazu gehört auch das "lazy" Nachladen von Beziehungs-Objekten), ist einer Default-Einstellung zu "verdanken", die nicht unumstritten ist (OSIV, siehe z.B. diesen Artikel bzw. diese Diskussion im Spring-Boot-Projekt). Eine "richtige", *zusammenhängende* Transaktion haben Sie in der Controller-Handlermethode nicht, wenn Sie mehrere DB-Zugriffe (über mehrere Service-Aufrufe) ausführen.

Es ist also völlig ok, z.B. zum Anlegen/Pflegen einer `Tour` z.B. eine Service-Methode

```
Tour speichereTourangebot(
        long anbieterid,
        Tour tour,
        long startortid, long zielortid);
```

zu verwenden, nachdem der Controller das zur Bearbeitung gemerkte `Tour`-Objekt bzgl. der einfachen (`int`, `String`)-Felder aktualisiert hat (wie auch beim `Benutzer` praktiziert). Die Service-Methode übernimmt das `Tour`-Objekt und stellt vor dem Abspeichern anhand der übergebenen IDs die benötigten Beziehungen her, die dazu benötigten Entitäten sind anhand ihrer ID leicht abzurufen (alles bis hin zur Speicherung der `Tour` in einer Transaktion).

Aufgabe 4 (Benutzer weg – Touren weg)

Was passiert, wenn Sie einen Benutzer, der Touren im Angebot hat, löschen? Sinnvollerweise sollten die Touren in diesem Fall mit verschwinden. Tun sie das bei Ihnen? Falls nein, erreichen Sie das leicht mit einem passenden `cascade`-Zusatz an der Beziehungsannotation für das Attribut `angeboteneTouren` in Ihrer `Benutzer`-Entität – mehr ist nicht nötig.cd /Volumes/tank/wwe

---

Akzeptanzkriterien für diesen Projektstand:

- Unter dem URI-Pfad `/ort` erscheint die Ortsliste mit der Möglichkeit, gewünschte Orts-Datenbankeinträge zu löschen, zu bearbeiten oder neue zu erfassen.
- Beim Bearbeiten ist das zugehörige Formular gemäß der zuletzt gespeicherten Version des Ortes vorbelegt. Das Namesfeld darf beim Abschicken des Formualars nicht unausgefüllt sein. Validierungsfehler werden im Formular gekennzeichnet.
- Unter dem URI-Pfad `/tour` erscheint die Tourenliste mit der Möglichkeit, gewünschte Tour-Datenbankeinträge zu löschen, zu bearbeiten oder neue zu erfassen.
- Beim Bearbeiten ist das zugehörige Formular gemäß der zuletzt gespeicherten Version der gewählten Tour vorbelegt.
- Die Tour-Bearbeitungsseite bietet in Selectboxen die in der Datenbank erfassten Benutzer (als Tour-Anbieter), Start- und Zielorte an, alle sind aus der DB (wie die anderen Felder auch) vorausgewählt/vorbelegt. Per Auswahl lässt sich die jeweilige Zuordnung ändern und abspeichern (auch sonstige Bearbeitung von Tourfeldern wie bisher).
- Beim Abschicken des Formulars muss für alle drei Select-Boxen eine Auswahl getroffen und ein positiver Wert für Preis und Anzahl der angebotenen Plätze angegeben sein. Validierungsfehler werden im Formular gekennzeichnet.
- Löschen eines Benutzers löscht automatisch auch seine angebotenen Touren.