

Systemtechnik Labor

4AHIT 2017/18

Komponentenbasierte Programmierung

Laborprotokoll

Marc Rousavy

22. April 2018

Bewertung:

Betreuer: Michael Borko

Version: 1.0

Begonnen: 12.4.18

Beendet: 20.4.18

Inhaltsverzeichnis

1	Einführung	3
1.1	Ziele	3
1.2	Voraussetzungen	3
1.3	Aufgabenstellung	3
1.4	Bewertung	5
1.5	Quellen	5
2	Lösung	6
2.1	Vorbereitung	6
2.2	Entities	7

1 Einführung

Diese Übung zeigt die Anwendung von komponentenbasierter Programmierung mittels Webframeworks.

1.1 Ziele

Das Ziel dieser Übung ist die automatisierte Persistierung und Verwendung von Objekten eines vorgegebenen Domänenmodells mittels eines Frameworks. Dabei sollen die CRUD-Operationen der verwendeten API zur Anwendung kommen.

Die Persistierung soll mittels der Java Persistence API (JPA) realisiert werden.

1.2 Voraussetzungen

- Grundlagen zu Java und das Anwenden neuer Application Programming Interfaces (APIs)
- Verständnis über relationale Datenbanken und dessen Anbindung mittels höherer Programmiersprachen (JDBC/ODBC)
- Verständnis von UML und Build-Tools

1.3 Aufgabenstellung

Erstellen Sie von folgendem Modell Persistenzklassen und implementieren Sie diese mittels JPA:

Suche Die Suche nach Zügen muss auf jeden Fall die Auswahl des Abfahrts- und Ankunftsortes (nur folgende Bahnhöfe sind möglich: Wien Westbhf, Wien Hütteldorf, St. Pölten, Amstetten, Linz, Wels, Attnang-Puchheim, Salzburg) ermöglichen. Dies führt zur Anzeige der möglichen Abfahrten, die zur Vereinfachung an jedem Tag zur selben Zeit stattfinden. Des weiteren wird auch die Dauer der Fahrt angezeigt.

In dieser Liste kann nun eine gewünschte Abfahrtszeit ausgewählt werden. Die Auswahl der Zeit führt zu einer automatischen Weiterleitung zum Ticketshop.

Um sich die Auslastung der reservierten Sitzplätze anzusehen, muss bei dem Suchlisting noch das Datum ausgewählt werden. Dieses Service steht jedoch nur registrierten Benutzern zur Verfügung.

Ticketshop Man kann Einzeltickets kaufen, Reservierungen für bestimmte Züge durchführen und Zeitkarten erwerben. Dabei sind folgende Angaben notwendig:

Einzeltickets: Strecke (Abfahrt/Ankunft), Anzahl der Tickets, Optionen (Fahrrad, Großgepäck) Reservierung: Strecke (Abfahrt/Ankunft), Art der Reservierung (Sitzplatz, Fahrrad, Rollstuhlstellplatz), Reisetag und Zug (Datum/Uhrzeit) Zeitkarte: Strecke, Zeitraum (Wochen- und Monatskarte)

Um einen Überblick zu erhalten, kann der Warenkorb beliebig befüllt und jederzeit angezeigt werden. Es sind keine Änderungen erlaubt, jedoch können einzelne Posten wieder gelöscht werden.

Die Funktion „Zur Kassa gehen“ soll die Bezahlung und den Ausdruck der Tickets sowie die Zusendung per eMail/SMS ermöglichen. Dabei ist für die Bezahlung nur ein Schein-Service zu verwenden um zum Beispiel eine Kreditkarten- bzw. Maestrotransaktion zu simulieren.

Prämienmeilen Benutzer können sich am System registrieren um getätigte Käufe und Reservierungen einzusehen. Diese führen nämlich zu Prämienmeilen, die weitere Vergünstigungen ermöglichen. Um diese beim nächsten Einkauf nützen zu können, muss sich der Benutzer einloggen und wird beim „Zur Kassa gehen“ gefragt, ob er die Prämienmeilen für diesen Kauf einlösen möchte.

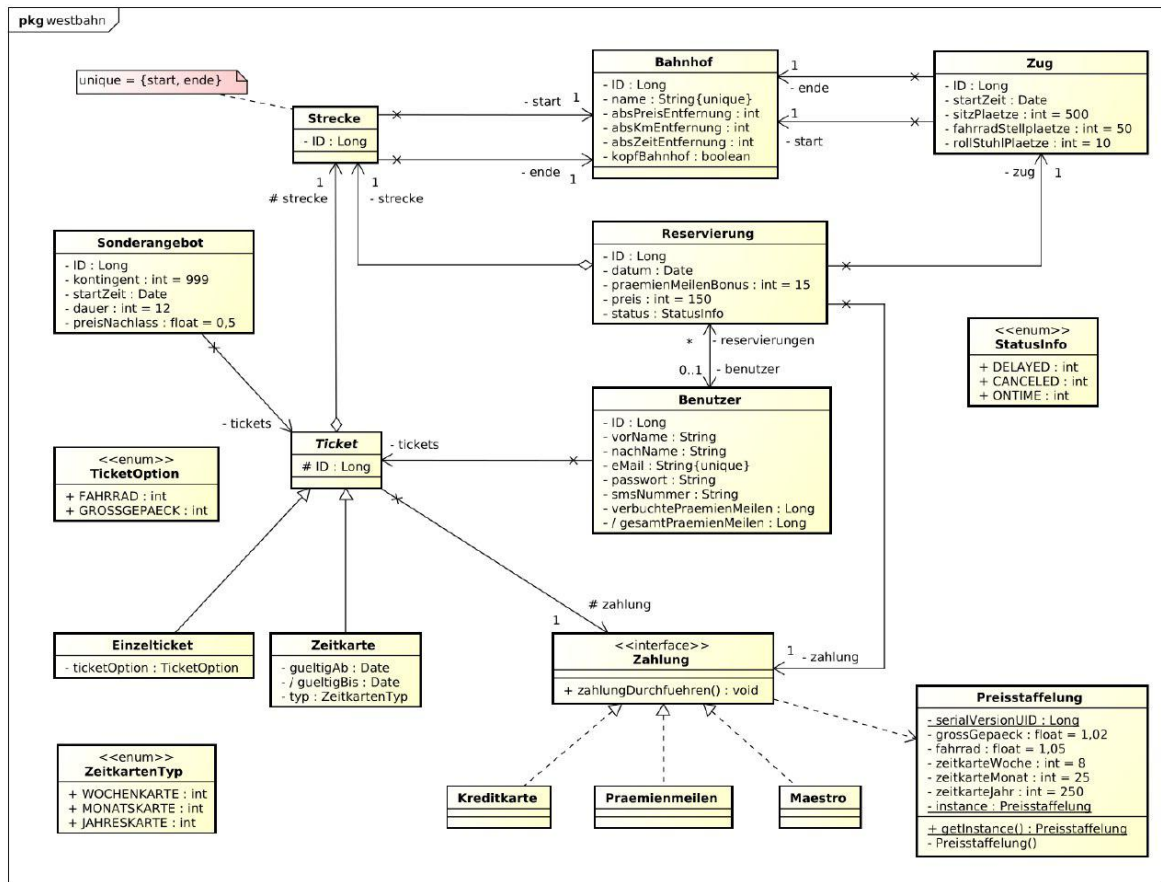


Abbildung 1: Die Westbahn Datenstruktur (UML)

Instant Notification System der Warteliste Der Kunde soll über änderungen bezüglich seiner Reservierung (Verspätung bzw. Stornierung) mittels ausgesuchtem Service (eMail bzw. SMS) benachrichtigt werden. Bei ausgelasteten Zügen soll auch die Möglichkeit einer Anfrage an reservierte Plätze möglich sein. Dabei kann ein Zuggast um einen Platz ansuchen, bei entsprechender änderung einer schon getätigten Reservierung wird der ansuchende Kunde informiert und es wird automatisch seine Reservierung angenommen.

Sonderangebote Für festzulegende Fahrtstrecken soll es ermöglicht werden, dass ein fixes Kontingent von Tickets (z.b.: 999) zu einem verbilligten Preis (z.b.: 50)

Task 1 - Mapping Schreiben Sie für alle oben definierten Klassen und Relationen entsprechende Hibernate JPA Implementierungen (javax.persistence.*). Bis auf die Klasse Reservierung sollen dafür die Annotationen verwendet werden. Die Klasse Reservierung soll mittels XML Mapping definiert werden.

Task 2 - Named Queries Schreiben Sie folgende NamedQueries (kein plain SQL und auch keine Inline-Queries) für das Domänenmodell aus Task1. Die Queries sollen die entsprechenden Parameter akzeptieren und die gewünschten Typen zurückliefern:

1. Finde alle Reservierungen für einen bestimmten Benutzer, der durch die eMail-Adresse definiert wird.
2. Liste alle Benutzer auf, die eine Monatskarte besitzen.
3. Liste alle Tickets für eine bestimmte Strecke aus (durch Anfangs- und Endbahnhof definiert), wo keine Reservierungen durchgeführt wurden.

Task 3 - Validierung Alle Constraints der einzelnen Entitäten sollen verifiziert werden. Hierfür soll die Bean Validation API verwendet werden. Folgende Einschränkungen sollen überprüft werden:

1. Zug und Strecke können nicht denselben Start- und Endbahnhof besitzen.
2. Die eMail des Benutzers soll ein gängiges eMail-Pattern befolgen.
3. Die Startzeit eines Sonderangebotes kann nicht in der Vergangenheit liegen.
4. Der Name eines Bahnhofs darf nicht kürzer als zwei und nicht länger als 150 Zeichen sein. Sonderzeichen sind bis auf den Bindestrich zu unterbinden.

1.4 Bewertung

- Gruppengröße: 1 Person
- Anforderungen "überwiegend erfüllt"
 - Dokumentation und Beschreibung der angewendeten Schnittstelle
 - Task 1
 - Task 2
- Anforderungen für Gänze erfüllt"
 - Task 3
 - Ausreichende Testobjekte zur Validierung der Persistierung
 - Überprüfung der funktionalen Anforderungen mittels Regressionstests

1.5 Quellen

"The Java EE Tutorial - Persistence"; Oracle; online: <https://docs.oracle.com/javaee/7/tutorial/partpersist.htm#BNBPY>

"HTML5 - A vocabulary and associated APIs for HTML and XHTML"; W3C; 17.12.2012; online: <https://www.w3.org/TR/2012/CR-html5-20121217/forms.html#valid-e-mail-address>

"Hibernate ORM Documentation"; JBoss; online: <http://hibernate.org/orm/documentation/5.2/>

"Hibernate ORM 5.2.13 Final User Guide"; JBoss; 25.01.2018; online: <https://docs.jboss.org/hibernate/orm/5.2/userguide/>

2 Lösung

2.1 Vorbereitung

Als Projekt wird eine **Java EE 8** Application mit folgenden Frameworks erstellt:

- **Maven** - Package/Library Management System
- **Java Persistence API** - Annotation Declarations für Entities und das ORM System
- **Hibernate** - Die Implementierung von Java Persistence API (JPA)
- **Java Database Connectivity (JDBC)** - Java Datenbank Interface

IntelliJ lädt automatisch benötigte Libraries, andernfalls kann man dies manuell hinzufügen.

Für die Libraries habe ich **Maven** verwendet, das **Maven** Project Object Model (POM) sieht dementsprechend folgendermaßen aus:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4     <modelVersion>4.0.0</modelVersion>
5
6     <groupId>mrousavy</groupId>
7     <artifactId>Westbahnhof</artifactId>
8     <version>1.0</version>
9
10    <dependencies>
11        <dependency>
12            <groupId>mysql</groupId>
13            <artifactId>mysql-connector-java</artifactId>
14            <version>6.0.6</version>
15        </dependency>
16        <dependency>
17            <groupId>org.hibernate</groupId>
18            <artifactId>hibernate-core</artifactId>
19            <version>5.2.16.Final</version>
20        </dependency>
21        <dependency>
22            <groupId>log4j</groupId>
23            <artifactId>log4j</artifactId>
24            <version>1.2.17</version>
25        </dependency>
26        <dependency>
27            <groupId>javax.xml.bind</groupId>
28            <artifactId>jaxb-api</artifactId>
29            <version>2.3.0</version>
30        </dependency>
31    </dependencies>
32 </project>
```

2.2 Entities

Entities sind einfache Klassen, bzw. Plain Old Java Object (POJO), welche durch die JPA auf die Datenbank *gemapped* werden.

Für die Westbahn, müssen alle Entities in dem UML (Siehe: Figur ??) implementiert werden.

Je **Entity** muss ein POJO erstellt werden, welches eine von der JPA verwaltete Schnittstelle zwischen dem Java Code und der Datenbank repräsentiert.

Abbildungsverzeichnis

1	Die Westbahn Datenstruktur (UML)	4
---	--	---