Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw School of
Management and Law

eUmzug Kanton Bern

Aufgabe 5: Persistenz-Layer eUmzug-Plattform

Modul Geschäftsprozessintegration im BSc Wirtschaftsinformatik

Autor
Björn Scheppler
scep@zhaw.ch

Herausgeber
Zürcher Hochschule für Angewandte Wissenschaften
School of Management and Law
Institut für Wirtschaftsinformatik (IWI)
Stadthausstrasse 14
CH-8041 Winterthur
www.zhaw.ch/iwi

Vorliegende Version
1.0.0 (19.10.2018)

Änderungshistorie

Version (Datum)	Wer	Was
1.0.0 (19.10.2018)	scep	Initiale Fassung

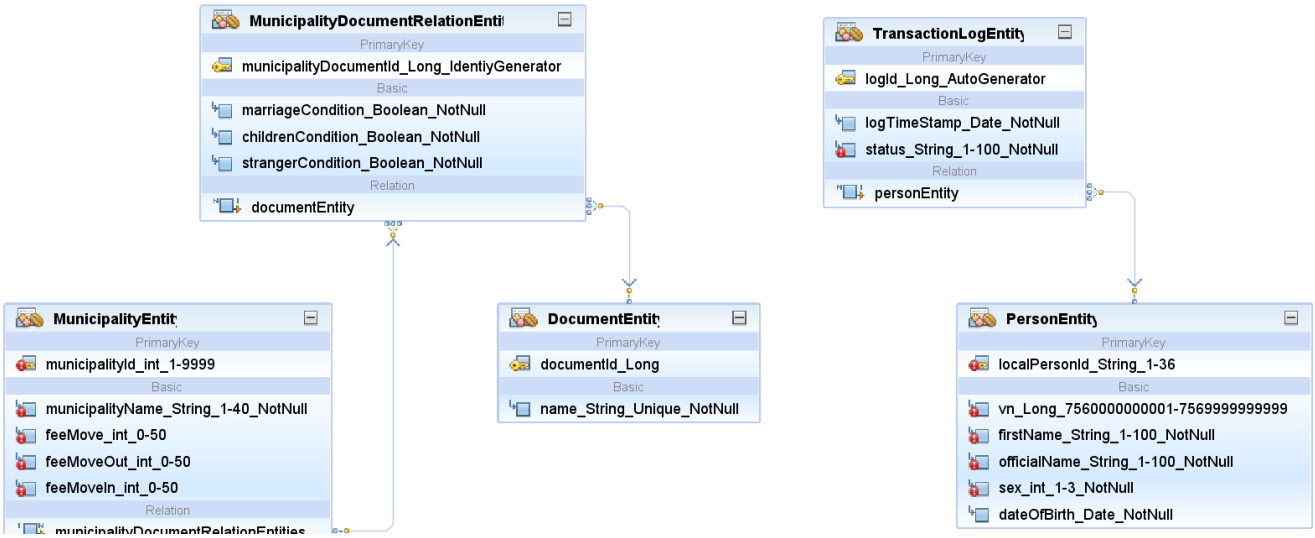
Hinweis: Aus Gründen der Lesbarkeit wurde im Text stets die männliche Form gewählt.

1 Zielsetzung

1. Am Beispiel der eUmzug-Plattform habt ihr gelernt, wie man mit einer relationalen Datenbank kommuniziert ohne SQL-Befehle schreiben zu müssen, sondern stattdessen **ORM** verwendet, in unserem Fall Hibernate als Implementierung von JPA im Spring-Framework.
2. Gewisse **Stammdaten** (z.B. Liste aller Gemeinden) als auch ausgewählte **Bewegungsdaten** erfolgreich oder fehlgeschlagen abgeschlossener Prozessinstanzen sollen in einer **Umzugsplattform-Datenbank** gehalten werden. Für diese sind **alle Persistierungs-Komponenten vorhanden**, von den Entities über Repositories, Datenbank und Testdaten bis zu Teilen der JavaDelegates. Dazu zählen die in der Kleinklasse vermittelten Teile (*Liste aller Gemeinden* sowie *Liste hochzuladender Dokumente einer Gemeinde*) als auch die selbständig zu erarbeitenden Teile (*XYZ persistieren* sowie *Gebühren ermitteln*).

2 Vorgeschlagenes Vorgehen

1. Besucht die **Kleinklasse** in der Semesterwoche 6, in welcher Björn mit euch die Starthilfe durchgeht. Damit habt ihr …
    a. … im Idealfall bereits **Punkt 1 der Kriterien** aus Kapitel 4.1 erfüllt sowie
    b. … gemeinsam mit dem Stoff von Peter aus der Grossklasse das **praktische Rüstzeug**, für die selbständig zu erarbeitenden Teile.
2. Beginnt mit dem Anpassen der **BPMN-Modelle**:
    a. **Öffnet die BPMN-Modelle** *UmzugMelden.bpmn*, *UmzugsmeldungErfassenUnd-Bezahlen.bpmn* sowie *PersonErfassenUndIdentifizieren.bpmn*.
    b. Lest die **Element Documentations** aller *XYZPersistieren*-Tasks (sind identisch bis auf den letzten Abschnitt) sowie des *GebuehrenErmitteln*-Tasks. Damit wisst ihr, was erreicht werden soll. Noch ein Hinweis fürs Verständnis: Die *feesTotal* und *feeMap* werden später im Semester benötigt beim *AlleAngabenPruefen*-Task, beim *Fallen Gebühren an?*-Gateway und natürlich im *ZahlungDurchfuehren*-Task.
    c. **Konfiguriert** die genannten Tasks, wie in der Element Documentation beschrieben (Implementation & Input Parameter).
    d. Erweitert den **VariablenStatischSetzen**-Script Task um weitere statische Variablen für localPersonId, firstName und officialName (vn, sex und dateOfBirth gehen einfacher anders -> siehe weiter unten) für eine x-beliebige Person eurer Wahl, damit später beim eigentlichen Persistieren einer Person entsprechende Angaben vorhanden sind.
3. Nun zu den **Entitäten**:
    a. Ergänzt die **MunicipalityEntity** um die noch fehlenden Attribute gemäss dem folgenden Klassendiagramm. Die Angaben zu Typ, möglichem Werte-/Grössenbereich, usw. sind jeweils pro Attribut nach dem Unterstrich aufgeführt.
    b. Erstellt die neuen Entitäten **PersonEntity** und **TransactionLogEntitiy** gemäss dem folgenden Klassendiagramm.



4. Löscht eure bisherige **umzugsplattform.mv.db**-Datenbank, startet die Applikation, loggt euch in der Konsole ein, führt die SQL-Statements aus initialData.sql aus.
5. Erstellt nun **PersonRepository** und **TransactionLogRepository** analog zu MunicipalityRepository.
6. Nun geht es an die Geschäftslogik und damit an die Verbindung zwischen Prozessapplikation und Datenbank mittels Service Task und der Implementation mit den **JavaDelegates**, zunächst die **PersistUserEntriesAndStatusDelegate**:
    a. Versucht, die Beschreibung in der Element Documentation von ErfolgreichenAbschlussPersistieren in einer neuen JavaDelegate-Klasse **PersistUserEntriesAndStatusDelegate** zu implementieren.
    b. Tipp: Überlegt euch zunächst, wie ihr dies implementieren wollt, schreibt dies Schritt-für-Schritt als **Zeilenkommentare** in die execute-Methode. Erst jetzt versucht ihr, eure Zeilenkommentare mit entsprechendem Code zum Leben zu erwecken.
    c. Und dann **testet** ihr euren Code im **Debug**-Modus, indem ihr eine neue Prozessinstanz erstellt, beim Schritt «Identifikationsmerkmale erfassen» klickt ihr auf «Variablen laden» und fügt neue Variablen hinzu für sex (integer, 1-3), dateOfBirth (Date, Format ist) und allenfalls vn (Long). Geht auch mit der H2-Konsole prüfen, ob wirklich eine neue Person und ein neuer TransactionLog-Eintrag erstellt wurde.
7. Nun geht es an die zweite Delegate-Klasse: **GetFeesDelegate**.
    a. Auch hier empfehlen wir **das gleiche Vorgehen**.
    b. Damit ihr diese Klasse **testen** könnt, wiederum im Debug-Modus durchgehen und eine neue Prozessinstanz starten. Wiederum bei «Identifikationsmerkmale erfassen» auf «Variablen laden» klicken, aber nun geht es einerseits darum, «municipalityIdMoveOut/In» auf Integer zu setzen und anderseits Ids einzugeben, die es auch wirklich in der Datenbank gibt (z.B. 305 für Kappelen und 302 für Bargen). PS: Es wird nicht erwartet, dass ihr Fehler abfängt, wie zum Beispiel, dass eine municipalityIdMoveOut erfasst wird, welche nicht in der Datenbank existiert.
    c. Um sicher zu sein, dass alles läuft, wie es soll, geht ihr nun vor dem Abschliessen von «Abschlussbestätigung anzeigen» ins **Camunda Cockpit** und prüft, ob dort feeMap und feesTotal als Variablen für die laufende Prozessinstanz vorhanden sind und die von euch erwarteten Werte haben.

3 Abgabe

1. Die inhaltliche Abgabe ist ein Commit und Push im Github-Repository der Gruppe.
2. Die formale Abgabe erfolgt auf Moodle hier durch EINE Person im Team.
3. Dies erfolgt spätestens bis zum im Betreff der verlinkten Aufgabe aufgeführten Zeit.
4. Die Abgabe besteht darin, die Id des von uns zu korrigierenden Commits als Text einzufügen und auf «Abgabe einreichen» zu klicken.

4 Bewertungskriterien

4.1 Kriterien

Maximal können **10 Punkte** erreicht werden, die sich auf die folgenden **Kriterien** aufteilen:

1. Teile aus **Kleinklassen-Unterricht-Starthilfe** (*Liste aller Gemeinden* sowie *Liste hochzuladender Dokumente einer Gemeinde*) korrekt implementiert (2 Punkte)
2. Korrekt angepasste **BPMN-Modelle** (1 Punkt), also
    a. *VariablenStatischSetzen* erweitert um sinnvolle Werte, damit beim Durchführen einer neuen Prozessinstanz im Service Task *ErfolgreichenAbschlussPersistieren* eine Person und ein Transaktions-Logbucheintrag in der Datenbank erstellt wird.
    b. Alle *…Persistieren*»-Tasks so konfiguriert, dass sie so funktionieren, wie in der *Element Documentation* beschrieben
3. Korrekte **Entitäten** (MunicipalityEntity, PersonEntity und TransactionLogEntitiy) gemäss den Anforderungen aus dem Klassendiagramm (2 Punkte)
4. Korrekte **Repositories** (*PersonRepository* und *TransactionLogRepository*) (1 Punkt)
5. **PersistUserEntriesAndStatusDelegate** implementiert wie in der *Element Documentation* beschrieben (2 Punkte)
6. **GetFeesDelegate** implementiert wie in der *Element Documentation* beschrieben (2 Punkte)

4.2 Aufwandreduktion 4er-Gruppen

In dieser Aufgabe erfolgt keine Aufwandreduktion.

4.3 0 Punkte

Keine Abgabe innerhalb der Abgabefrist auf Moodle und Github.

4.4 Punkteabzug

Für die folgenden formalen Fehler und fehlenden/falschen Dokumentationen gibt es die Abzüge auf die erreichte Punktzahl gemäss Kapitel 4.1. Diese Abzüge werden aufsummiert, wobei eine Gruppe nicht weniger als 0 Punkte erreichen kann.

1. **Keine fristgerechte Abgabe auf Moodle**, aber auf Github eine Lösung gepusht: 4 Punkte
2. **Auf Moodle formal eine falsche Abgabe** (z.B. keine vorhandene Commit-ID, etwas anderes als eine Commit-ID, usw.): 2 Punkte
3. Java-Klassen in falsch(en) (bezeichneten) **Packages**: 1 Punkt
4. Java-Klassen, Methoden und Variablen **falsch benannt** (Klassen anders als vorgegeben sowie Gross-/Kleinschreibung): maximal 2 Punkte
5. Klassen und Methoden ohne aussagekräftige Java Documentation (**JavaDoc**). Enthält eine Klasse nur eine Methode, so reicht eine Java Documentation der Klasse: maximal 4 Punkte
6. Jedes Java-Statement (mit Semikolon beendet) sowie if-then-else- oder for-Blöcke enthlaten einen oder mehrere zutreffende, selbst formulierte, deutschsprachige **Zeilenkommentare**. Davon ausgenommen sind Standard-Getter- & Setter-Methoden sowie wenn direkt untereinander mehrfach dasselbe gemacht wird (z.B. Auslesen von vier Prozessvariablen in lokale Variablen). Verletzung dieser Regel: maximal 4 Punkte.

5 Feedback

Spätestens am 8.11.2018 10:00 ist die **Musterlösung** unterhalb der Aufgabenstellung auf Moodle freigeschaltet.

Spätestens am 21.11.2018 20:00 sieht diejenige Person, welche die Aufgabe eingereicht hat, die erreichte Punktzahl insgesamt als auch die bewertete Excel-Datei mit allfälligen Kommentaren im Feld «Bemerkungen» ganz rechts, wenn sie erneut auf die **Moodle-Aufgabe** klickt.

6 Fortgeschrittene Anforderungen (optional)

6.1 Mögliche zu implementierende Anforderungen

6.1.1 Separate Datenbank (vermutlich einfach)

Ausgangslage: Der Einfachheit halber sind die Tabellen der Umzugsplattform momentan in derselben Datenbank, welche die Camunda Process Engine für ihre eigenen Tabellen nutzt.

Anforderung: Die in den obigen Basis-Anforderungen aufgeführten Tabellen sind in einer separaten dateibasierten H2-Datenbank mit der Bezeichnung *umzugsplattform_stammdaten.mv.db*. Trotzdem soll alles weiterhin so funktionieren wie bei den Basis-Anforderungen.

6.1.2 Status-Typen als eigene Entität (vermutlich eher einfach)

Ausgangslage: Momentan wird in der TransactionLogEntity der Status als String gespeichert. Dies bringt mehrere Nachteile mit sich, wie z.B. keine Möglichkeit für Mehrsprachigkeit.

Anforderungen:

1. Der Status wird zu einer eigenen Entität mit den Attributen *id* (Autowert), *name* (z.B. *FINISHED_SUCCESSFUL*), *germanText* (z.B. *Erfolgreicher Abschluss des eUmzugsmeldung*) und *englishText* (z.B. *Successful completion of the eRelocation report*).
2. Für die für den eUmzug relevanten Stati gibt es eine-SQL-Datei mit den entsprechenden INSERT-Statements.
3. *PersistUserEntriesAndStatusDelegate* sowie der *Input Parameter «status»* in den «XYZ persistieren»-Service Tasks sind sinnvoll anzupassen

6.1.3 Basis für REST-Zugriff und Web-Applikation (vermutlich eher komplex)

Ausgangslage: Momentan können die Stammdaten (Gemeinden, Dokumente, Personen) und Bewegungsdaten (Transaktions-Log-Einträge) in der H2-Datenbank lediglich über SQL-Befehle verändert werden (z.B. in der H2-Console), respektive mit «PersistUserEntriesAndStatusDelegate» können neue Personen und Transaktions-Log-Einträge erstellt werden.

Aufgaben-übergreifende Zielsetzung: Eleganter wäre es, wenn Mitarbeitende des Kantons (und der Gemeinden) über eine separate Webapplikation (nicht Camunda) sowohl Stammdaten verwalten als auch den Status zu einer bestimmten Person abfragen könnten. Hierzu braucht es entsprechende Persistierungskomponenten und Controller-Klassen (diese Aufgabe), eine separate Webapplikation (Aufgabe 7) und noch eleganter REST-Controller (Aufgabe 9).

Anforderungen:

1. Folgende Funktionalität für die Verwaltung von **Dokumenten** sollen in einer DocumentController-Klasse angeboten werden: Hinzufügen eines neuen Dokuments, Suchen eines Dokuments über seinen Namen, Ausgeben einer Liste aller Dokumente, Umbenennen eines Dokuments, Löschen eines Dokuments sofern in keiner Beziehung mehr genutzt.
2. Dasselbe gilt für **Gemeinden** in einer MunicipalityController-Klasse mit folgenden Abweichungen/Ergänzungen: Verändern der drei Gebührenwerte für eine Gemeinde; beim Löschen einer Gemeinde soll automatisch auch die zugehörigen MunicipalityDocumentRelationEntities gelöscht werden.
3. Für das **Transaktions-Logbuch** sollen in einer TransactionLogController-Klasse folgende Funktionalität angeboten werden: Den aktuellsten Status-Eintrag (*status* und *logTimeStamp*) für eine bestimmte Person ausgeben; Alle möglichen Status-Einträge (status UNIQUE) zurückgeben (nur relevant, falls nicht 6.1.2 umgesetzt wird) alphabetisch sortiert; Eine Liste aller Personen mit einem bestimmten Status zurückgeben, sortiert nach *logTimeStamp*.

6.2 Abgabe

Die Regeln in Kapitel 5.2 der Hauptaufgabestellung einhalten. Falls die Abgabe nicht erst am 24.12.2018, sondern gleichzeitig mit den Basis-Anforderungen erfolgt, dann in der Abgabe auf Moodle (siehe Kapitel 3 oben) nicht bloss die Commit Id angeben, sondern auf einer zweiten Zeile auch der Hinweis: «Fortgeschrittene Anforderungen implementiert».