Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw School of
Management and Law

eUmzug Kanton Bern

Aufgabe 9: REST

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 (28.11.2018)

Änderungshistorie

Version (Datum)	Wer	Was
1.0.0 (28.11.2018)	scep	Initiale Fassung

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

1 Zielsetzung

1. Am Beispiel des VeKa-Dienstes, der Umzugsplattform und Camunda habt ihr **gelernt**, …
    a. … wie **IT-Systeme** in Prozessapplikationen **eingebunden** werden,
    b. … über eine von mehreren Protokollen/Architekturprinzipien, konkret **REST**,
    c. … im Kontext des Spring Frameworks über **RestController** und **RestTemplate**,
    d. … und wie REST-Services wie z.B. die VeKa-API und die Process Engine API effizient getestet/aufgerufen werden können über Webservice-Clients wie **soapUI**.
2. Für den **VeKa-Dienst** habt ihr den bestehenden Controller zu einem RestController ausgebaut.
3. In der **Umzugsplattform** nutzt ihr den VeKa-Dienst über ein RestTemplate.

2 Vorgeschlagenes Vorgehen

1. Besucht die **Kleinklasse** in der Semesterwoche 11 für die Kapitel 1 und 3 der Starthilfe und geht das Twitter-Review-Kapitel 9 durch (Anleitung und Videos). Damit wisst ihr …
    a. … wie ihr eure Projekte (VeKa, GWR und Umzugsplattform) mit den passenden **Musterlösungen** überschreibt.
    b. … in Kombination mit dem Know-How aus der Grossklasse, wie ihr einen **RestController** baut und ein **RestTemplate** nutzt.
2. Ergänzt den **VeKa-Dienst** so, dass er den Anforderungen gemäss den Bewertungskriterien 1 und 2 entspricht.
3. Optional: **Testet** die Funktionsweise eures VeKa-Dienstes, indem ihr über soapUI ein neues Testprojekt erstellt mit passenden Requests.
4. Besucht die **Kleinklasse** in der Semesterwoche 12 für das Kapitel 4.3 der Starthilfe. Damit seid ihr in der Lage, den VeKa-Dienst, das Personenregister und das GWR in der Umzugsplattform zu nutzen.
5. Ergänzt die **Umzugsplattform** so, dass sie den Anforderungen gemäss den Bewertungskriterien 3 und 4 entspricht.
6. Optional: **Testet** die Funktionsweise der angepassten Umzugsplattform, indem ihr über soapUI den Prozess von Beginn an testet oder gezielt nach der Aktivität *WohnverhaeltnisErfassen* und vor der neuen Aktivität *NeueAdresseVeKaCenterMitteilen*.
7. Gebt die Aufgabe gleichzeitig mit Aufgabe 10 ab.

3 Abgabe

Da ohnehin kein Feedback mehr während dem Semester gegeben werden kann, soll die Aufgabe 9 gleichzeitig mit Aufgabe 10 abgegeben werden auf Moodle, also am 24.12. Details folgen in der Aufgabenstellung zu Aufgabe 10.

4 Bewertungskriterien

4.1 Kriterien

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

1. Funktionierender und korrekter **CardController** in VeKa-Applikation (**1 Punkt**):
    a. Der CardController aus der Musterlösung aus Aufgabe 6 ist so umgebaut, dass er über **/vekaapi/v1/cards/{cardNumber}** aufrufbar ist und dabei entweder einen 404-Status zurückgibt oder eine card im JSON-Format sowie 200-Status.
    b. Bezüglich Aufbau orientiert euch am UserService aus dem Twitter-Review-Prozess.
2. Funktionierender und korrekter **PersonController** in VeKa-Applikation (**3 Punkte**):
    a. Ein neuer **PersonController** (und allenfalls weitere damit verbundene Klassen) erlaubt, dass über **/vekaapi/v1/persons/{id}/address** eine Methode per POST aufrufbar ist,
    b. die zum **Ziel** hat, die postalische Adresse einer Person zu ändern (**Adressänderung**),
    c. wobei der **Request** eine **Personen-Id** in der Url erwartet und als Request-Body ein JSON-Objekt mit dem Aufbau von ch.zhaw.gpi.veka.AddressEntity enthält, allerdings ohne id-Attribut,
    d. und wobei die **Response** stets nur ein http-Status ist, entweder 404, falls keine Person zur Id gefunden wurde oder ansonsten 200,
    e. und innerhalb der Methode mit folgenden Fällen **sinnvoll umgegangen** wird:
        i. Die übergebene (neue) Adresse entspricht bereits der aktuell zugewiesenen (alten) Adresse.
        ii. Die übergebene Adresse existiert bereits in der Datenbank der VeKa-Applikation, entspricht jedoch nicht der aktuell zugewiesenen Adresse.
        iii. Die übergebene Adresse existiert noch nicht in der Datenbank.
        iv. Die bisher einer Person zugewiesene Adresse ist nach der Änderung keiner weiteren Person mehr zugewiesen.
3. Funktionierende und korrekte Implementation der Aktivität **GrundversicherungPruefen** gemäss Element Documentation (**3 Punkte**)
4. Funktionierende und korrekte Implementation der Aktivität **NeueAdresseVeKaCenterMitteilen** gemäss Element Documentation (**2 Punkte**)
5. **Eleganz** und **Robustheit** des Codes: Erläuterungen bei Bedarf in Aufgabe 6 (**1 Punkt**)

4.2 Aufwandreduktion 4er-Gruppen

Kriterium 4 in Kapitel 4.1 ist nicht zu implementieren. Da diese Aufwandreduktion beträchtlich ist (2 Punkte geschenkt) wird es in Aufgabe 10 keine Aufwandreduktion geben.

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 [soweit 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 25.12.2018 10:00 ist die **Musterlösung** unterhalb der Aufgabenstellung auf Moodle freigeschaltet.

Spätestens am 2.1.2019 20:00 sieht diejenige Person, welche die Aufgabe eingereicht hat, die erreichte Punktzahl pro Kriterium und insgesamt inklusive allfälliger Kommentare, wenn sie erneut auf die **Moodle-Aufgabe** klickt.

6 Fortgeschrittene Anforderungen (optional)

6.1 Mögliche zu implementierende Anforderungen

6.1.1 Grundversicherungsprüfung fortgeschritten (vermutlich eher komplex)

Diese fortgeschrittene Anforderung wurde in Aufgabe 7 bereits beschrieben (siehe Kapitel 6.1.4), passt inhaltlich aber auch bestens zur aktuellen Aufgabe.

6.1.2 WebApplikation für VeKa-Center (vermutlich komplex)

Diese fortgeschrittene Anforderung wurde in Aufgabe 7 bereits beschrieben (siehe Kapitel 6.1.5), passt inhaltlich aber auch bestens zur aktuellen Aufgabe, sofern dort die Variante 2 mit REST gewählt wurde.

6.2 Abgabe

Die Regeln in Kapitel 5.2 der Hauptaufgabestellung einhalten. Die Abgabe erfolgt am 24.12.2018 gleichzeitig mit den Basis-Anforderungen. In der Abgabe auf Moodle daher nicht bloss die Commit Ids angeben, sondern auf einer weiteren Zeile auch der Hinweis: «Fortgeschrittene Anforderungen implementiert».