Übung 3: Spring Framework

- ## Übungaufgabe 3 (Spring)

  → **Deadline: 25. 3. 2016, 24:00 Uhr**

Vervollständigung der DB-Zugriffsschicht

Vervollständigen Sie die Implementierung der DAOs für die Domänenklassen *Employee*, *LogbookEntry*,*Project* und der Klassen, die im Zusammenhang mit Sprints, Anforderungen und Tasks in der 2. Übungsaufgabe erstellt wurden. Verwenden Sie für die Realisierung der Datenbankzugriffschicht die JPA und setzen Sie die zugehörigen Unterstützungsklassen des Spring-Frameworks ein. Es ist Ihnen überlassen, ob Sie die *Plain JPA API* oder *Spring Data JPA* verwenden.

Vervollständigung der Geschäftslogik

Folgende Funktionen sind zumindest zu implementieren. Aus den Anforderungen an den unten beschriebenen Konsolen-Clients können sich zusätzliche Funktionen ergeben.

- Einfügen/Aktualisieren eines Mitarbeiters,
- Zugriff auf einen Mitarbeiter per Schlüssel bzw. Zugriff auf alle gespeicherten Mitarbeiter.
- Einfügen eines Projekts.
- Definition eines Projektleiters.
- Zuordnung eines Mitarbeiters zu einem Projekt, Löschen einer Projektzuordnung.
- Speichern einer Anforderung.
- Speichern und Löschen einer Aufgabe.
- Suchen aller Anforderungen für ein Projekt.
- Suchen aller Aufgaben, die zu einer Anforderung gehören.
- Ermittlung aller Daten für die Ermittlung eines Burn-Down-Charts für einen Sprint.

Vergewissern Sie sich, dass die Operationen der Geschäftslogik in Transaktionen eingebettet sind.

Integrationstest

Implementieren sie für Ihre Geschäftslogik eine Test-Suite, die folgende Anforderungen erfüllt:

- Definieren Sie mindestens 15 Testfälle.
- Achten Sie darauf, dass die Tests voneinander unabhängig sind.
- Für die Integrationstests ist eine In-Memory-Datenbank zu verwenden.
- Verwenden Sie die Unterstützungsklassen zum Testen des Spring-Framworks

Unittest *(optional, für Beurteilung mit sehr gut erforderlich)*

Implementieren Sie für Ihre Geschäftslogik zusätzlich Unit-Test, indem Sie Ihre DAOs durch Mock-Objekte ersetzen. Die Wahl des Mocking-Frameworks ist Ihnen überlassen.

Dinu Marius-Constantin Aufwand: 17h
SEv16.6.SWT UE, Sebastian Pimminger

Interaktiver Konsolen-Client

Implementierung Sie einen interaktiven Konsolen-Client, der auf Basis der Geschäftslogik die im Folgenden angeführte Funktionalität realisiert:

- Anzeige aller Mitarbeiter.
- Anlegen eines neuen Projekts.
- Zuordnung von Mitarbeitern zu einem Projekt, Rücknahme dieser Zuordnung.
- Auflistung aller einem Projekt zugeordneten Mitarbeiter.
- Auflistung aller Anforderungen eines Projekts, gruppiert nach Sprints.
- Auflistung aller Aufgaben, die einer Anforderung zugeordnet sind.
- Hinzufügen einer Anforderung für ein Projekt.
- Zuordnung einer Anforderung zu einem Sprint.
- Hinzufügen einer Aufgabe, die einer bestehenden Anforderung zugeordnet werden muss.
- Ändern/Löschen einer Aufgabe.
- Ausgabe eines Burn-Down-Charts für einen bestimmten Sprint.

Der Konsolen-Client soll auf eine persistente Datenbank zugreifen, die mit sinnvollen Datensätzen in einem vernünftigen Umgang befüllt ist.

Allgemeine Hinweise

- Überlegen Sie sich gut, welche schreibenden Operationen Sie kaskadieren und welche Ladestrategien Sie bei den verschiedenen Beziehungen verwenden. Es ist nicht sinnvoll und auch nicht erlaubt, ausschließlich sofortiges Laden („eager fetching“) einzusetzen.
- Vermeiden Sie in der Geschäftslogik und der Präsentationsschicht so weit wie möglich Technologiecode. Nutzen Sie dafür die Konzepte des Spring-Frameworks.
- Wählen Sie für Ihre Anwendung eine Architektur, die sicherstellt, dass es bei Operationen auf Ebene der Präsentationsschicht (also im Konsolen-Client) beim verzögerten Nachladen zu keinen Ausnahmen kommen kann.

Lösungsidee:

Umbau der DAOs auf Spring Data Repositories. Applikationskontext wird beim Start mittels Spring Boot initialisiert und scanned die Pakete beginnend ab der Application Klasse.

Die Konfigurationen beziehen sich auf die AppConfig.java Datei, welche trotzdem das applicationContext.xml File referenziert. Die Beans werden mittels Component, Service oder Controller angesprochen, je nach Veranlagung der Klasse.

Die Service Klassen (Geschäftslogikklassen) besitzen transaktionale Annotationen und im Controller wird die Hibernate Sessions solange offengehalten um Lazy-Loading zu unterstützen bei einer bestimmten User Operation.

Dinu Marius-Constantin Aufwand: 17h
SEv16.6.SWT UE, Sebastian Pimminger

Die Unit-Tests wurden auf des Spring Framework angepasst und unterstützen SpringJUnit4ClassRunner, transaktionale Operationen und Rollbackfunktionalitäten.

Alle Referenzen der Applikation werden per Dependency Injection via Type Inference aufgelöst.

Die Applikation wurde nach dem MVC Pattern aufgebaut und beinhaltet eine Konsolen View, Konsolen Controller und die bereitgestellten Service Klassen.