Praktikum zu Fortgeschrittenes Web-Engineering, Wintersemester 2020/2021

Praktikum 4

Im Rahmen des Praktikums entwickeln wir eine Web-Anwendung, die wir Schritt für Schritt mit Anforderungen, Funktionen und Technologien erweitern.

Im vierten Praktikum wenden wir uns dem Backend zu. Wir beginnen mit dem Aufbau eines Spring-Basisprojektes und einer Fachlogik- und Persistenzschicht für unsere Web-Anwendung.

Stellen Sie sämtliche Ergebnisse und Änderungen in Ihrem Git-Repository zur Verfügung.

Aufgabe 1: Spring-Projekt mit Persistenzschicht aufbauen

1. Rahmenprojekt über Spring Initializr

Lassen Sie sich über den Spring Initializr (https://start.spring.io) ein Rahmenprojekt erzeugen. Verwenden Sie dazu folgende empfohlenen Einstellungen:

1. *Project:* Maven Project
2. *Language:* Java
3. *Spring Boot:* 2.4.0
4. *Projekt Metadata:*
    1. *Group:* de.fhdo
    2. *Artifact:* klausurapp
    3. *Packaging:* Jar
    4. *Java:* 11
5. *Dependencies:* Spring Web, Spring Data JPA, Spring Boot DevTools, H2 Database, Thymeleaf

Über *Generate* erzeugt der Spring Initializr ein Rahmenprojekt als ZIP-Archiv, welches schon vorkonfiguriert ist und direkt in Ihre IDE importiert werden kann (z.B. Eclipse: ZIP-Archiv entpacken und dann über *File → Import... → Existing Maven Projects* importieren).

2. Vorbereitete Persistenzschicht einbauen

Um nicht vollständig auf der grünen Wiese beginnen zu müssen, finden Sie in GitLab eine vorbereitete (wenn auch unvollständige) Persistenzschicht: https://git.inf.fh-dortmund.de/01/svjoe001/web2-klausurapp-basis-wise2021. Die Persistenzschicht verwendet Spring Data JPA (Java Persistence API) zur Definition der Datenbankentitäten und sorgt für eine initiale Befüllung der Datenbank mit Testdaten.

Folgende Packages finden Sie in der Persistenzschicht:

- `de.fhdo.klausurapp.domain` : Enthält die Datenbankentitäten. Diese Klassen nutzen Annotationen der Java Persistence API (JPA), um zu beschreiben, wie ein entsprechendes Objekt auf eine relationale Datenstruktur abgebildet werden soll (objektrelationales Mapping, ORM).
- `de.fhdo.klausurapp.bootstrap` : Beinhaltet die Klasse `DummyDataBootstrap` , die die initiale Befüllung der Datenbank übernimmt. Falls Sie weitere Testdaten benötigen, können Sie diese Klasse entsprechend erweitern.
- `de.fhdo.klausurapp.repositories` : Beinhaltet Interfaces für Repository-Beans, die Funktionalitäten zum Zugriff auf die Datenbank anbieten. Diese Interfaces verwenden nicht die in der Vorlesung erwähnte Stereotyp-Annotation `@Repository` , da wir Spring Data JPA einsetzen, welches Repositories auf eigene Weise verwaltet. Es mag irritieren, dass in diesem Package nur Interfaces enthalten sind und keine Implementierungen - dennoch muss hier nichts weiter implementiert werden! Die konkreten Repository-Beans werden von Spring Data JPA automatisch zur Laufzeit zur Verfügung gestellt.
- `de.fhdo.klausurapp.dto` : Enthält zu jeder Datenbankentität ein passendes *Data Transfer Object (DTO)*. **Im Sinne einer sauberen Entkopplung sollen oberhalb der (von Ihnen zu implementierenden) Fachlogikschicht nur DTOs verwendet werden und keine Datenbankentitäten!**
- `de.fhdo.klausurapp.converters` : Bietet Hilfsklassen zur Konvertierung von DTOs in Datenbankentitäten (und umgekehrt) an.

Zudem enthält die Persistenzschicht unter `/src/main/resources/application.properties` eine beispielhafte Properties-Datei, über welche Sie bei Bedarf das Logging von Datenbankzugriffen einstellen können.

Alle Dateien sind mit Kommentaren versehen, die ggf. zusätzliche Informationen zur Benutzung enthalten.

2.1 Persistenzschicht in das Rahmenprojekt kopieren und starten

Kopieren Sie die Dateien der Persistenzschicht in Ihr Rahmenprojekt (siehe 1.). Der Quellcode sollte kompilieren und problemlos ausführbar sein (z.B. Eclipse: Starten der Hauptklasse, die mit `@SpringBootApplication` annotiert ist, z.B. durch Rechtsklick auf die Klasse → *Run As* → *Java Application*). Beim Starten der Anwendung wird der Web-Container Apache Tomcat sowie die In-Memory-Datenbank H2 hochgefahren und über den `DummyDataBootstrap` mit Testdaten befüllt.

2.2 Funktionskontrolle über H2-Konsole

Kontrollieren Sie die Funktionstüchtigkeit Ihres Projektes durch einen Blick in die H2-Konsole. Öffnen Sie dazu nach Start der Anwendung in einem Web-Browser die URL http://localhost:8080/h2-console/.

Durchsuchen Sie zur Ermittlung der JDBC-URL die Konsolenausgabe Ihrer Anwendung. Sie finden die URL

in einer Ausgabe der Form:
`H2 console available at '/h2-console'. Database available at '$JDBC_URL".`
Kopieren Sie den Wert *$JDBC_URL*.

Geben Sie dann folgende Daten in das Login-Formular der H2-Konsole ein:

- JDBC URL: *$JDBC_URL*
- User Name: *sa*
- Password: leer lassen

Nach erfolgreichem Login können Sie die angelegten Tabellen und Daten inspizieren.

Weitere Informationen:

Falls Sie für Ihre Web-Anwendung die vorbereitete Persistenzschicht modifizieren oder erweitern wollen (dies ist *keine Pflicht, sondern freiwillig!*), so könnten Ihnen die folgenden Informationsquellen weiterhelfen:

- Accessing Data with JPA
- Spring Data JPA - Reference Documentation
- Introduction to the Java Persistence API

Aufgabe 2: Fachlogikschicht implementieren

In dieser Aufgabe implementieren Sie die Fachlogikschicht der Web-Anwendung. Diese bildet die Basis für die folgenden Praktika, in denen wir mit Spring auch die weiteren Architekturschichten der Anwendung realisieren werden.

1. Fachlogik implementieren

Konzipieren und implementieren Sie die Fachlogikschicht in Form fachlicher *Services*, die mindestens folgende Funktionalitäten ermöglichen:

1. Alle Klausuren lesen
2. Neue Klausur anlegen
3. Klausur anhand der ID lesen
4. Neue Aufgabe zu Klausur hinzufügen
5. Klausureinträge zu Klausur lesen
6. Neuen Studenten/neue Studentin anlegen
7. Neuen Klausureintrag zu einem Studenten/einer Studentin und einer Klausur anlegen
8. Bewertung(en) zu Klausureintrag hinzufügen

Verwenden Sie dazu die Mittel, die Ihnen die vorbereitete Persistenzschicht (siehe Aufgabe 1) bietet. Bei Bedarf dürfen Sie diese auch erweitern. Die Services sollen von Spring verwaltete Komponenten (Beans) sein. Verwenden Sie zu diesem Zweck geeignete *Stereotyp-Annotationen*. Sorgen Sie bei den

entstehenden Services für eine geeignete Trennung der Zuständigkeiten.

Hinweis: Nicht alle der o.g Funktionalitäten benötigen zwingend eine eigene Methode in einem Service!

2. Fachlogik ausprobieren

Testen Sie die Funktionstüchtigkeit der Fachlogikschicht durch Aufruf aller Service-Methoden, z.B. mit entsprechenden Konsolenausgaben in der Hauptklasse (mit `@SpringBootApplication` annotiert).

Hinweis: Bei Problemen mit Fehlern der Art `org.hibernate.LazyInitializationException` ist es typischerweise notwendig, die betroffene Service-Methode mit der Annotation `org.springframework.transaction.annotation.Transactional` zu versehen. Ursache für das Problem ist zumeist das Nachladen von Daten (*lazy loading*) außerhalb einer Transaktion, die mittels `@Transactional` explizit auf die Service-Methode festgelegt wird.