Datenmanagement - Datenbankanwendungen mit JDBC

Seminar 2, am 07.12.2017

Level 1

- Neues Java-Projekt „JDBC“ anlegen
- JDBC Treiber im Java-Projekt referenzieren - ihr habt diese Referenzen schon in der ersten LV über Maven eingebunden!
- Testklasse mit `Main`-Function erstellen
- `DatabaseFacade`-Klasse implementieren
- `getConnection`-Methode implementieren und im `main` ausprobieren

Level 2

- Neues Java-Projekt „Model“ erstellen
- Projekt im „JDBC“-Projekt referenzieren
- Eure bereits geschriebenen Modellklassen sollen in diesem Projekt liegen
- Für eure bereits erstellten POJOs die `toString()`-Methode überschreiben, sodass sie für die jew. Klasse sinnvoll ist

Level 3

- Im Projekt „JDBC“
- Methode `getAllReservations()` in der `Facade` implementieren und mit JDBC alle Objekte aus der Datenbank laden
- Im Main `getAllPersons()` aufrufen und die Personen mit `toString()` in die Konsole schreiben

Level 4

- Im Projekt „JDBC“
- Klasse `BrokerBase<T>` erstellen und abstrakte Methoden definieren(Was muss ein Broker können?)
- Beispiel-Broker (z.B. `ReservationBroker`) implementieren (von `BrokerBase` erben)
- Den Code von `getAllReservations` aus der `Facade` in den Broker runterziehen
- Die `Facade` hat weiterhin `getAllPersons`, delegiert das aber jetzt an den Broker
- Im `Main` wird weiterhin nur `DatabaseFacade.getAllPersons()` aufgerufen

Level 5

- Im Projekt „JDBC“
- Weitere Broker implementieren
- Facade erweitern mit abstrahierten Funktionen für verschiedene Datentypen
- z.B: `save<T>(T item)`
- Herausfinden, welchen Broker man braucht
- Broker erstellen/laden, Delegation aufrufen (`insert`, `update` oder `save`)

Level 6

- Im Projekt „JDBC“
- POJOs von MoJOs trennen (Broker erstellt POJO, Facade gibt MoJO zurück)
- Weitere Aufrufe implementieren (Speichern, Verändern, Löschen…) und in allen Schichten abbilden