Persistenz unter Kontrolle mit JDBI für Java

Alexander Schwartz
msg systems ag
Eschborn

Schlüsselworte
Java, JDBI, Persistenz, Datenbank.

Einleitung

JDBI ist eine Java-Bibliothek, die typsicheren Datenbank-Zugriff und Transaktionssteuerung für SQL-Datenbanken anbietet. Es ist als Open-Source-Bibliothek verfügbar und wird seit mehreren Jahren kontinuierlich weiterentwickelt.

Im Zentrum von JDBI steht die effiziente Ausführung von SQL-Statements aus Java heraus. Hierfür bietet die Bibliothek die notwendigen Abstraktionen an, die JDBC fehlen.

Im Gegensatz zu JPA wird das SQL dabei nicht versteckt, sondern kann bewusst genutzt werden, um effizient auf die SQL-Datenbank zuzugreifen.

Im Gegensatz zu JDBC sind benannte Parameter und eine robuste Transaktionssteuerung möglich. Java-Objekte können sowohl für die Parameterübergabe als auch für die Antworten angebunden werden.

Dieser Vortrag gibt einen ersten Einstieg in die Welt von JDBI und zeigt, wie es die Aufgaben des Alltags effizient bewältigen kann.

Persistenz für Java

SQL bietet einen effizienten Zugriff auf relationale Datenbanken. Mit INSERT, UPDATE und DELETE können Daten eingefügt, geändert und gelöscht werden.

Standard für das Interface zwischen Java und relationalen Datenbanken ist JDBC. Teilweise gibt es sogar für NoSQL Datenbanken JDBC Treiber.

Mit JDBC lassen sich SQL-Statements direkt ausführen. Sobald den SQL-Statements Parameter übergeben werden sollen, sind diese Parameter mit Fragezeichen als Platzhalter in den Statements vorzusehen. Dies birgt ein hohes Fehlerpotential und führt zu viel sich wiederholenden Java-Quellcode.

Objektrelationale Mapper wie JPA versprechen hier Abhilfe und verstecken dabei das erzeugte SQL hinter Java Objekten. Einfache Datenmapper erfordern dabei keinen Aufwand mehr. Selektiver Zugriff auf einzelne Spalten ist damit aber nicht mehr möglich. Nutzung von datenbankspezifischem SQL ist hier nicht gewünscht.

Als minimale Abstraktionsschicht für JDBC bietet sich JDBI an. Es erlaubt die effiziente Verwendung von SQL, vermeidet redundanten Java-Code, und erlaubt einen sicheren Umgang mit Parametern. Auch die Transaktionssteuerung kann von JDBI übernommen werden.

Ein Zugriff auf eine Tabelle sieht dann wie folgt aus:

```
@RegisterMapper(SightingJDBIMapper.class)

public interface SightingJDBI {



    @SqlQuery("SELECT sight_id, sight_vessel_name, sight_date_time,
        sight_timez_name, version FROM sighting ORDER BY sight_id")

    List<Sighting> findAll();



    @SqlUpdate("INSERT INTO sighting (sight_vessel_name, sight_date_time,
        sight_timez_name, version) VALUES (:sightVesselName,
        :sightDateTime, :sightTimezName, :version)")

    @GetGeneratedKeys

    long insert(@BindBean Sighting sighting);



    @SqlQuery("SELECT sight_id, sight_vessel_name, sight_date_time,
        sight_timez_name, version FROM sighting WHERE sight_id = :sightId")

    Sighting findNameById(@Bind("sightId") long id);

}
```

Über die Annotation @SqlQuery können SQL-Abfragen direkt übergeben werden. Parameter können direkt aus Java-Beans ausgelesen werden, wenn diese mit @BindBean als Parameter übergeben werden. Einzelne Werte können mit @Bind übergeben werden. Mit @SqlUpdate können Werte geändert werden. Mit @GetGeneratedKeys werden die beim INSERT von der Datenbank automatisch erzeugten Schlüssel zurückgegeben.

Die Annotation @RegisterMapper bewirkt, dass für das Ergebnis einer Abfrage, das bei JDBC zunächst ein ResultSet ist, ein Mapper hinterlegt werden kann. Dieser überführt die Einträge des ResultSets in Java-Objekte.

Über zusätzliche Annotationen können weitere SQL-Parameter übergeben werden, z. B. kann über @MaxRows die maximale Anzahl von zurückgelieferten Werten festgelegt werden, und über @QueryTimeOut die maximale Dauer der Abfrage festgelegt werden.

Über die Annotation @Transaction können mehrere Aufrufe in einem Transaktionskontext gebündelt werden.

Über die Annotationen @SqlBatch und @BatchChunkSize können Daten effizient geschrieben werden. Ebenso können über Abfrage-Ergebnisse über Java-Iteratoren Datensatz für Datensatz entgegengenommen werden. Damit ist eine effiziente Batch-Verarbeitung möglich.

Stored Procedures können mit den Annotationen @SqlCall und @OutParameter eingebunden werden.

Damit bietet sich JDBI als minimale Abstraktion für den Zugriff auf relationale Datenbanken an. Es hilft, redundanten und fehleranfälligen Code zu vermeiden und erlaubt eine effiziente Nutzung von SQL.

Mehr Informationen zu JDBI finden sich unter http://jdbi.org/.

Kontaktadresse:
Alexander Schwartz
msg systems ag
Mergenthalerallee 73-75
65760 Eschborn
Deutschland

Tel.: +49 6196 99845-5585
Fax: +49 6196 99845-5451
Mobil: +49 171 5625767
E-Mail: alexander.schwartz@msg-systems.com
Internet: http://www.msg-systems.com