Multitenancy und JPA
Eine Bestandsaufnahme

Bernd Müller
Ostfalia
Hochschule Braunschweig/Wolfenbüttel
Salzdahlumer Straße 46/48
38302 Wolfenbüttel

Schlüsselworte:
Multitenancy, JPA, Java-EE 7, EclipseLink, OpenJPA, Hibernate.

Einleitung
Der JSR Java EE 7 (JSR 342) hat als großes Arbeitsthema die Cloud. Für JPA bedeutet dies, dass Anwendungen mandantenfähig sein müssen. Wir betrachten die augenblicklich verfügbaren Möglichkeiten der JPA-Provider *EclipseLink*, *OpenJPA* und *Hibernate* zur Mandantenfähigkeit und stellen diese vor.

1. Motivation und Standortbestimmung

Der Java Specification Request für Java EE 7 (JSR 342) hat als großes Arbeitsthema die Cloud. Damit wird Enterprise Java als Service (PaaS) möglich und sowohl Anwendungen als auch die EE-Infrastruktur müssen mandantenfähig sein, wie die folgenden Zitate verdeutlichen:

- „*... and deliver their functionality as a service with support for features such as multitenancy and elasticity*..."
- „*... Similarly, a single application can be used by multiple tenants with the same security, isolation, and quality of service guarantees*. ..."

Für einige im Umbrella-JSR 342 enthaltenen Spezifikationen bedeutet dies ebenfalls, eine Art von Mandantenfähigkeit einzuführen, insbesondere auch für die Persistenz. Der JSR Java Persistence 2.1 (JSR 338) nennt als eine Anforderung explizit

- „*Support for multitenancy*"

Wir wollen uns im Folgenden die bisherigen Möglichkeiten zur Mandantenfähigkeit der drei populärsten JPA-Implementierungen EclipseLink, OpenJPA und Hibernate anschauen. Der Autor ist nicht Mitglied der Expertengruppe des JSR 338. Die dargestellten Möglichkeiten sind daher als Bestandsaufnahme der aktuellen Code-Basen der Provider zu verstehen und bedeuten nicht, dass JPA 2.1 diese Möglichkeiten letztendlich enthalten wird.

Was bedeutet *Multitenancy*?
Unter Multitenancy versteht man die Möglichkeit, mehrere Mandanten in einer Anwendung verwalten zu können. Die Realisierung kann immer programmatisch erfolgen. Sinn von JSRs ist es jedoch, dies möglichst ohne explizite Programmierung, optimalerweise durch Deklaration zu erreichen. Im Java-EE-Umfeld sind von einer Mandantenfähigkeit mindestens Oberfläche (JSF), Anwendungslogik (Session Beans) und Persistenz (JPA) betroffen.

Für JPA kann eine Mandantenfähigkeit offensichtlich erreicht werden durch

1. getrennte/separate Datenbanken
2. getrennte/separate Schemata
3. gemeinsames Schema

Wir betrachten im Folgenden die Möglichkeiten der JPA-Provider EclipseLink, OpenJPA und Hibernate.

2. Multitenancy mit EclipseLink

EclipseLink führt mit der Version 2.3.0 (Indigo Release Train) 3 Annotationen und eine Enumeration zur Mandantenfähigkeit ein: `@Multitenant`, `@TenantDiscriminatorColumn`, `@TenantDiscriminatorColumns` und `MultitenantType`. Die alleinige Verwendung von `@Multitenant`, wie im folgenden Beispiel

```
@Entity
@Multitenant
public class Kunde {
    @Id @GeneratedValue
    private Integer id;
    private String vorname;
    private String nachname;
    @Temporal(TemporalType.DATE)
    private Date geburtsdatum;
    ...
}
```

führt zu einer zusätzlichen Spalte in der Kundentabelle. Nach unserer obigen Klassifikation wird also die dritte Alternative eines gemeinsamen Schemas verwendet. Abbildung 1 zeigt das Tabellenschema, das von EclipseLink bzw. der entsprechnden Tools auch generiert werden kann.

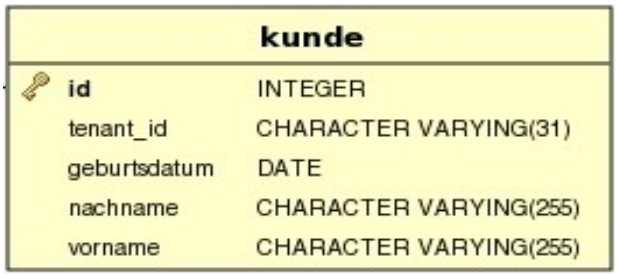

Abb. 1: Kundentabelle mit Tenant-Spalte *(Times New Roman, 10 Punkt, kursiv)*

Die Annotation `@Multitenant` erlaubt im optionalen Attribut `value` die Auswahl der beiden Enum-Werte `SINGLE_TABLE` und `TABLE_PER_TENANT`, wobei der zweite Wert im Augenblick nicht unterstützt wird und zu einer Exception führt. Die zweite Alternative ist also in Arbeit.

Der Default für die zusätzliche Spalte kann mit `@TenantDiscriminatorColumn` überschrieben werden, also etwa

```
@Entity
@Multitenant
@TenantDiscriminatorColumn(name = "mandant")
public class Kunde {
    @Id @GeneratedValue
    private Integer id;
    private String vorname;
    private String nachname;
    @Temporal(TemporalType.DATE)
    private Date geburtsdatum;
    ...
}
```

Die Verwendung mehrerer Spalten ist über `@TenantDiscriminatorColumns` möglich.

Verwendet wird die Mandantenfähigkeit durch das Property `eclipselink.tenant-id`, das beim Erzeugen der `EntityManagerFactory` in der `persistence.xml` oder in Java definiert werden muss. Es sind Strings als auch Integer als Typ möglich.

Es werden die üblichen CRUD-Opterationen des EntityManagers inklusive alle drei Query-Arten unterstützt.

3. Multitenancy mit OpenJPA

OpenJPA besitzt keine speziellen Features.für Mandantenfähigkeit. Die Erweiterung *Slice* erlaubt es jedoch, Enities auf verschiedene Datenbanken zu verteilen. Wir verwenden den aktuellsten Snapshot der Version 2.2.0. Die Hauptziele von Slice sind Verteilung und Replikation, es kann jedoch auch zur Mandantenfähigkeit nach unserer Klassifikationsart 1 verwendet werden.

Dazu muss Slice in der Persistenzeinheit konfiguriert werden:

```
<property name="openjpa.BrokerFactory" value="slice" />
<property name="openjpa.slice.Names" value="mandant1,mandant2" />
<property name="openjpa.slice.Master" value="mandant1" />
<property name="openjpa.slice.Lenient" value="true" />

<property name="openjpa.ConnectionDriverName"
          value="org.postgresql.Driver"/>
<property name="openjpa.ConnectionUserName" value="..."/>
<property name="openjpa.ConnectionPassword" value="..."/>
<property name="openjpa.slice.mandant1.ConnectionURL"
          value="jdbc:postgresql://localhost/bank1"/>
<property name="openjpa.slice.mandant2.ConnectionURL"
          value="jdbc:postgresql://localhost/bank2"/>
```

Die Verteilung auf die Datenbanken erfolgt aufgrund verschiedener Call-Back-Methoden, die bestimmte Policy-Interfaces implementieren müssen:

- Data Distribution Policy
- Query Target Policy
- Finder Target Policy

Die Verteilung selbst kann mit der ersten Policy sowohl auf Umgebungs-Property-Ebene analog zu EclipseLink als auch über ein Entity-Feld erfolgen. Für das Lesen über Queries bzw. `em.find()` sind die beiden anderen Policies zu implementieren.

4. Multitenancy mit Hibernate

Hibernate enthält in der Version 4.0.0 CR2 die Klasse `MultiTenantConnectionProvider`, die die Mandantenfähigkeit der Klassifikationsart 1 analog zu OpenJPAs Slice realisieren soll. Leider konnten wir keine Dokumentation oder Beispiele finden, die eine sinnvolle Verwendung erlaubt hätten, so dass wir zu Hibernate keine weiteren Ausführungen geben können.

5. Zusammenfassung

Die Arbeiten zu JPA 2.1 befinden sich im Fluss. Die drei populären Provider EclipseLink, OpenJPA und Hibernate arbeiten am Thema Mandantenfähigkeit, wobei EclipseLink am weitesten fortgeschritten scheint. Man darf gespannt sein, welche (oder keine ?) Ansätze sich durchsetzen und letztendlich in JPA 2.1 aufgenommen werden.

Kontaktadresse:

Bernd Müller
Ostfalia
Hochschule Braunschweig/Wolfenbüttel
Salzdahlumer Straße 46/48
D-38302 Wolfenbüttel

Telefon: +49 (0) 5331-939 31160
Fax: +49 (0) 5331-939 31004
E-Mail bernd.mueller@ostfalia.de
Internet: www.ostfalia.de