**Praktikum Blatt 09 Webbasierte Anwendungen – Hertel, Withöft 17/06/19**

Was bewirkt die Methode **persist()**vom Interface **EntityManager?** 1P

Persist() nimmt eine Instanz einer Entität und fügt sie dem PersistenceContext hinzu.

* Das Objekt wird dem EntityManager hinzugefügt
* Die Instanz wird gemanaged (also weitere Updates an der Entität werden wahrgenommen)
* Die Entität ist darauf vorbereitet in die Datenbank eingefügt zu werden

Zitat:

*"A newly created instance of the entity class is passed to the persist method. After this method returns, the entity is managed and planned for insertion into the database. It may happen at or before the transaction commits or when the flush method is called. If the entity references another entity through a relationship marked with the PERSIST cascade strategy this procedure is applied to it also."*

*Quelle:* <https://vladmihalcea.com/a-beginners-guide-to-jpahibernate-flush-strategies/>

Vererbung mit JPA:

JPA unterstützt drei Arten von Vererbung:

SINGLE\_TABLE

JOINED\_TABLE

TABLE\_PER\_CONCRETE\_CLASS

**Single Table:**

Single-Table-Strategie nimmt alle Klassen Felder (beide Super-und Unterklassen) und ordnen Sie sie nach unten in einer einzigen Tabelle als SINGLE\_TABLE Strategie. Bei SINGLE\_TABLE müssen alle nicht PK-Felder zudem nullable sein, da alle Unterklassen in dieser Tabelle gespeichert werden und einige Typen diverse Spalten nicht verwenden. INGLE\_TABLE hat den Vorteil des perfomanten Zugriffs, da alles in einer Tabelle gespeichert wird. Unterscheiden sich die konkreten Unterklassen stark und ist die Anzahl hoch, ergibt sich eine sehr breite Tabelle.

dtype|id|common|specificA|specificB

**Joined Table:**

Beigetreten Tabelle Strategie ist es, die referenzierte Spalte, die eindeutige Werte, um die Tabelle zu beitreten und einfache Transaktionen machen teilen.

**Table per concrete Class:**

Tabelle pro Klasse Strategie ist es, eine Tabelle für jede Sub-Entität erstellen.

Beim Standard(SINGLE\_TABLE) werden alle Unterklassen der Vererbungshierarchie auf eine Datenbank-Tabelle gemapped.

Bei JOINED werden alle abstrakten Klassen und konkreten Klassen der Vererbungshierarchie in einer separaten Tabelle persistiert.

Bei TABLE\_PER\_CLASS wird für jede konkrete Klasse der Vererbungshierarchie eine eigene dedizierte Tabelle angelegt.

Realisierung

Es gibt eine abstrakte Klasse. Die Subklassen erben davon und sind @Entity.

Bei SINGLE\_TABLE und JOINED muss nach JPA-Standard eine @DiscriminatorValue angegeben werden. Dieser ist vom Typ String und trägt im Default den Namen dtype und dient dazu die konkreten Klassen voneinander zu unterscheiden: In der Datenbank wird in der Tabelle der jeweilige String in die Spalte dtype geschrieben.

Wäre es TABLE\_PER\_CLASS sind polymorphe Abfragen nicht möglich.

Bei JOINED werden bei Abfragen nach konkreten Subklassen inner joins generiert,

bei polymorphen Abfragen outer joins.