
Zwischenbericht Großer Beleg

Vergleich Persistenzkonzepte Newtron Framework – J2EE

Robert Vogel

Inhalt

- Persistenz in J2EE
- Persistenz Newtron Framework
- Beispielprojekt
- Vergleich
- Migration

Persistenz in J2EE

Entity Beans 2.0

- Local Interfaces
- CMP / CMR
- Home Methods
- Select Methods
- EJB QL
- Transaktionen

Persistenz Newtron Framework

- Überblick
- Architektur
- Syntax
- Funktionen von DBObject
- Relationen
- Transaktionen
- Beispiele

Überblick

- Objektorientierter Abstraktionslayer über relationalem Datenbankschema
- JDBC basiert
- Klassen werden aus Beschreibung automatisch generiert
- Features
 - Caching synchronisiert über mehrer Applikationen
 - Unterstützung von Relationen (1:1, 1:n, n:m)
 - Transaktionssteuerung, verschachtelte Transaktionen

Syntax

- Java-Klasse von DBObject/DBIdentObject abgeleitet
- Attributbeschreibung in einem Kommentar

```
/*  
    #begindescription  
    addPrimaryKey("ident", "int", "");  
    ...  
    #enddescription  
*/
```

Syntax

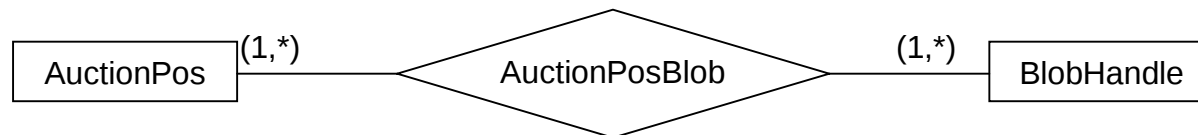
- `addAttribute(„value“, „int“, „unique number“);`
 - wie `addPrimaryKey`, fügt aber ein Attribut hinzu
- Datentypen
 - `byte`, `int`, `double`, `long`
 - `String`, `String50`, `String250`, `String4000`
 - `Date`
 - `java.util.Date`

Funktionen von DBObject

- insert()
 - fügt ein Objekt in die Datenbank ein
- delete()
 - löscht das Objekt aus der Datenbank
- lock()
 - sperrt das Objekt optimistisch

Relationen

- `$mtonrelation=1;`
 - Stellt dieses Objekt als n:m Relation dar
- 1:1-Methode liefert DBObject
- 1:n-Methode liefert Liste von DBObjects
- n:m-Methode liefert Liste von DBObjects (readonly)



- `@relations=qw( blobhandles )`
 - Liefert Relation-Klasse (read/write)

Transaktionen

- Änderungen können nur innerhalb einer Transaktion erfolgen
- wird von DBManager erzeugt und verwaltet
- Objekte innerhalb der gleichen Transaktion sehen Änderungen sofort, außerhalb erst nach commit
- verschachtelte Transaktionen
`Transaction childTransaction = new Transaction(parentTransaction)`
 - Parent-Transaktion sieht Änderungen innerhalb Child-Transaktion erst nach deren commit
 - Commit der Parent-Transaktion bewirkt kein commit der Child-Transaktion

Beispiele (1)

- Anlegen eines neuen Logins

```
Transaction t = new Transaction(dbm);  
Login l = new Login(t);  
l.insert();  
l.setName("Test");  
t.commit();
```

Beispiele (2)

- Löschen eines Logins

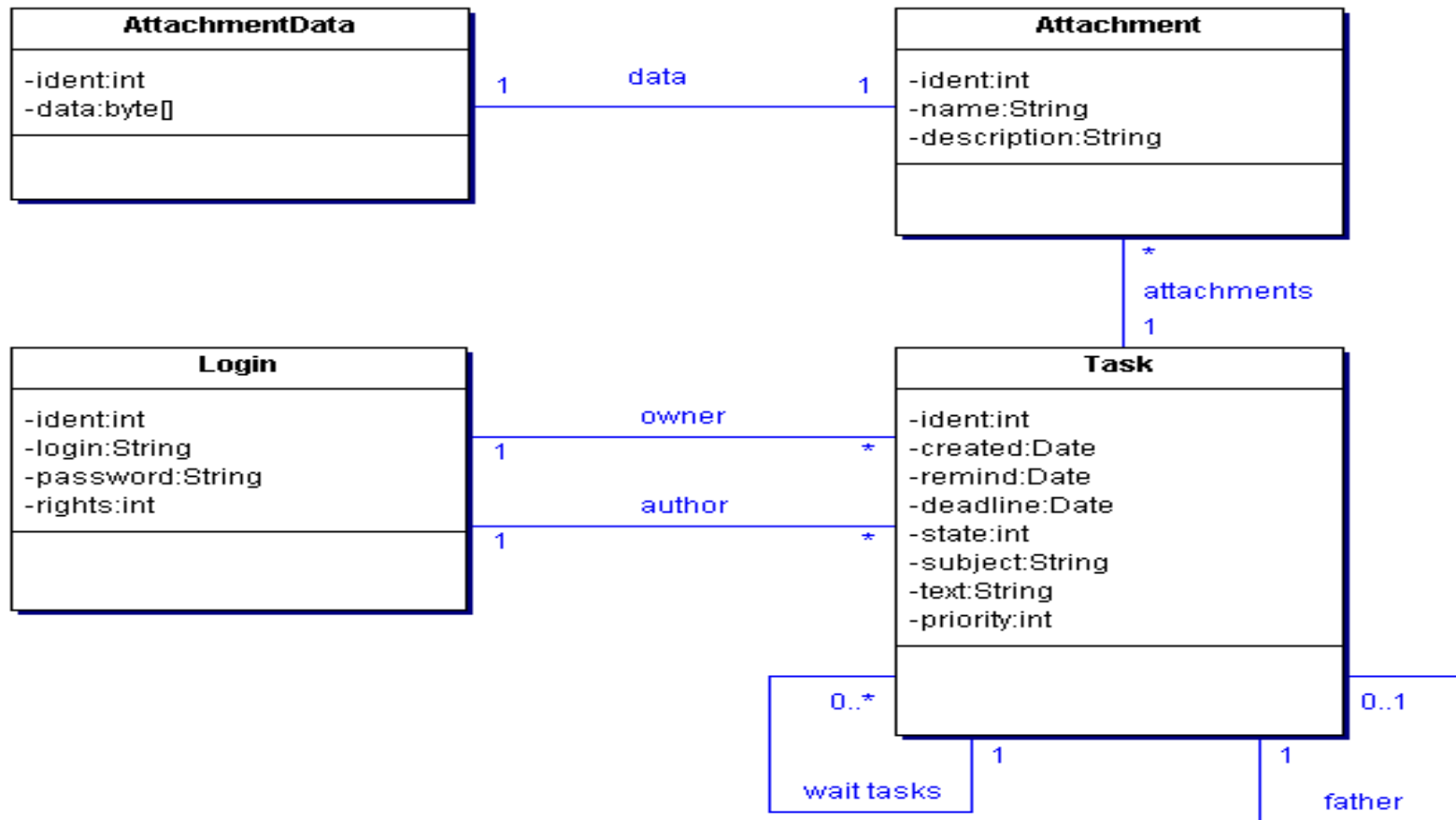
```
Transaction t = new Transaction(dbm);  
new Login(t, 1).delete();  
t.commit();
```

Beispiel (3)

- Erzeugen von Logins aus einem ResultSet und Änderung aller Logins

```
Transaction t = new Transaction(dbm);
ResultSet rs;
rs = t.executeQuery("Select * from Login
                    where ...");
while (rs.next())
{
    Login l = new Login(t, rs);
    l.setName("Test");
}
t.commit();
```

Beispielprojekt



Vergleich

- starke Abweichung in der Architektur
 - Entity Bean werden im EJBContainer verwaltet (RMI)
 - DBObjects können lokal benutzt werden (Caching)
- höheres Abstraktionsniveau bei Entity Beans
- Kritikpunkte Entity Beans 2.0
 - mangelnde Akzeptanz Entity Beans (Impl., Tools)
 - EJB QL
 - Performance
 - Deployment
- große Ähnlichkeiten Newtron Framework mit JDO

Migration (Fragen)

- Gründe für Migration ?
 - Marketing ? / höhere Kundenakzeptanz ? / Nutzung von Standardtechnologien ? / Akzeptanz des Frameworks ?
- Kosten ? / Zeit ?
 - worst case: Neuimplementierung Applikationen + Framework
 - Lizenzen ? / Aneignung J2EE Know-How
- Effizienzsteigerung ?
 - Vereinfachungen durch den Einsatz von J2EE ?
 - Tools ?
- Flexibilität ?
 - Anpassung an eigene Bedürfnisse ?

Migration (Szenarien)

- komplette Neuimplementierung auf J2EE-Basis
 - bestehende Applikationen + Framework
- Implementierung einer Zwischenschicht (1)
 - Kompatibilitätsschicht für bestehende Applikationen
 - Neuentwicklungen auf J2EE
- Implementierung einer Zwischenschicht (2)
 - Weiternutzung des Frameworks
 - Zwischenschicht realisiert Umsetzung auf J2EE

Migration (Probleme)

- Generator zur Erzeugung der Wrapper
- Steuerung der Transaktionen / Nested Transactions
- Realisierung der lokalen Caches (Synchronisation)
- Verstöße gegen J2EE-Patterns
 - z.B. Session Bean wraps Entity Bean, Value Object
- Umsetzung der JDBC-Statements auf J2EE
 - bei Verwendung CMP 2.0 Abbildung auf EJB QL
- Umsetzung Optimistisches-Locking
- Performance

Stand / Ausblick

- Einarbeitung Newtron-Framework ✓
- Beispielprojekt
 - Newtron-Framework ✓
 - J2EE ✓
- Vergleich ✓
- Migrationsvorschläge erstellen
- Tools / Applikationsserver
 - Portierung Beispielprojekt nach
 - BEA WebLogic 7.0
 - JBoss 3.0
 - Erstellung App.Server-Requirements
 - weitere Tools (z.B. Together 6.0)