IMPLEMENTASI *OBJECT RELATIONAL MAPPING* (ORM) MENGGUNAKAN HIBERNATE (STUDI KASUS : APLIKASI PEMINJAMAN INVENTARIS PROGRAM STUDI INFORMATIKA UNSOED)

Bangun Wijayanto
Jurusan Teknik Informatika, Fakultas Sains dan Teknik,Universitas Jenderal Soedirman
Jl. Dr. Soeparno No.61 Kampus Unsoed Karangwangkal
Telp./Fax (0281) 638793
E-mail: bangun.wijayanto@gmail.com

ABSTRAK
Perkembangan pemrograman berorientasi objek saat ini sangat pesat. Banyak perangkat lunak saat ini yang dikembangkan dengan paradigma berorientasi objek. Kendala yang sering dihadapi pada saat ini adalah pengggunaan basisdata relasional dalam menyimpan data. Meskipun saat ini sudah terdapat sistem basisdata berorientasi objek namun yang menjadi kendala adalah ketika akan mengubah sistem yang menggunakan basisdata relasional. Object Relational Mapping (ORM) adalah salah satu teknik untuk memetakan basisdata relasional ke model objek. Dari hasil penelitian disimpulkan bahwa Hibernate mampu melakukan pemetaan serta menyederhanakan proses penyimpanan dan pengambilan data objek .

Kata Kunci: object relational mapping (ORM), Hibernate, basisdata berorientasi objek

1. PENDAHULUAN

Pada awal perkembangan pemrograman web berbasiskan Java, pengembang mengakses basisdata menggunakan berbagai macam *class* yang disediakan oleh java.sql *package*. Masalah yang sering muncul pada saat itu adalah ketika pengembang lupa untuk menutup koneksi basisdata, sehingga akan muncul eksepsi pada aplikasi setelah berjalan beberapa saat.

Beberapa tahun kemudian "*connection pools*" menjadi topic utama dimana pengembang tidak perlu lagi dipusingkan dengan pembuatan dan pengaturan koneksi basisdata dan lebih berfokus pada SQL dan pemrosesan *ResultSet*. Permasalahan koneksi basisdata dapat diatasi akan tetapi masih menjadi kewajaran pada saat itu ketika terdapat banyak baris program untuk menjalankan *query* serta melakukan parsing terhadap *ResultSets* yang dihasilkan oleh *query*.

Beberapa perusahaan mulai mengembangkan teknologi sistem basisdata baru yang dapat digunakan secara spesifik untuk melakukan penyimpanan terhadap objek. Menurut Taryana (2010) sistem basisdata berorientasi objek memiliki keunggulan diantaranya:

1. Penggunaan basis data objek untuk mengimplementasikan pengembangan perangkat lunak berorientasi objek tidak membutuhkan langkah pemodelan physical data model, pemodelan query dalam basis data.
2. Perancangan basis data pada pengembangan perangkat lunak berorientasi objek tidak dilakukan, oleh karena itu pengembangan lebih terfokus pada tahap analisis dan perancangan *class*.

Meskipun sistem basisdata berorientasi objek tersebut dapat secara transparan menyimpan dan memuat objek ke Java atau bahasa pemrograman berorientasi objek lainnya, tetapi terdapat isu ketika merubah sistem yang telah berjalan atau terdapat aplikasi lain yang ingin mengakses basisdata relasional yang telah tersedia (Pugh, 2004).

Perbedaan antara model relasional dengan model objek sering dikenal sebagai *Object-Relational Impedance Mismatch.* Teknologi *Object relational mapping* (ORM) muncul untuk menjembatani masalah diatas.

Object relational mapping melakukan pemetaan terhadap tabel-tabel pada basisdata relasional dengan suatu *class* yang ada pada bahasa pemrograman berorientasi objek.

Modul aplikasi peminjaman inventaris program studi informatika Unsoed dikembangkan pada *platform* Java menggunakan rangka kerja (*framework)* Strut dengan basisdata relasional menggunakan Interbase.

Tujuan penelitian ini adalah untuk mengimplementasikan *object relational mapping* menggunakan hibernate sebagai jembatan dalam mengatasi masalah *Object-Relational Impedance Mismatch* yang terjadi

2. TINJAUAN PUSTAKA

2.1 Hibernate

Hibernate dimulai pada tahun 2001 oleh Gavin King sebagai alternatif dari penggunaan EJB2. Hibernate adalah sebuah pustaka pemetaan objek-relasional (*object-relational mapping library*) untuk bahasa pemrograman Java.

Hibernate menyediakan rangka kerja (*framework*) untuk melakukan pemetaan dari model

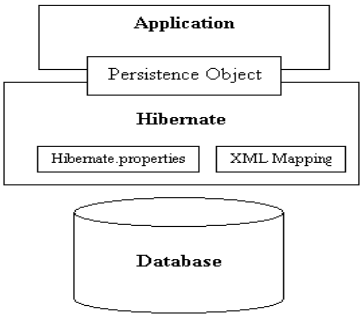
berbasiskan objek (*object-oriented domain model*) ke basisdata relasional. Tujuannya adalah untuk menyederhanakan serta memberikan kemampuan *persistence* yang lebih baik dibandingkan dengan EJB2 yaitu dengan menyederhanakan kompleksitas serta menyempurnakan kekurangan fitur pada EJB2.

2.2 Arsitektur Hibernate

Arsitektur Hibernate terdiri atas tiga komponen utama yaitu:

a. Manajemen Koneksi (*Connection Management*). Manajemen koneksi menyediakan pengaturan yang efisien terhadap koneksi basisdata. Koneksi basisdata adalah bagian yang paling penting karena banyak bagian dari program yang membuka dan menutup koneksi ke basisdata.
b. Manajemen Transaksi (*Transaction management*). Manajemen transaksi menyediakan kemampuan kepada pengguna untuk mengeksekusi lebih dari satu perintah ke basisdata dalam satu waktu.
c. Pemetaan Objek-Relasional (Object relational mapping) Pemetaan Objek-Relasional adalah teknik untuk memetakan representasi data dari model objek ke model relasional. (Roseindia, 2011)

Terdapat tiga bagian penting dalam melakukan proses pemetaan basisdata relasional ke model objek yaitu *class* Java, berkas xml untuk pemetaan (hbm.xml) dan berkas Hibernate.properties yang berisi pengeturan koneksi ke basisdata. Arsitektur Hibernate digambarkan pada Gambar 1.

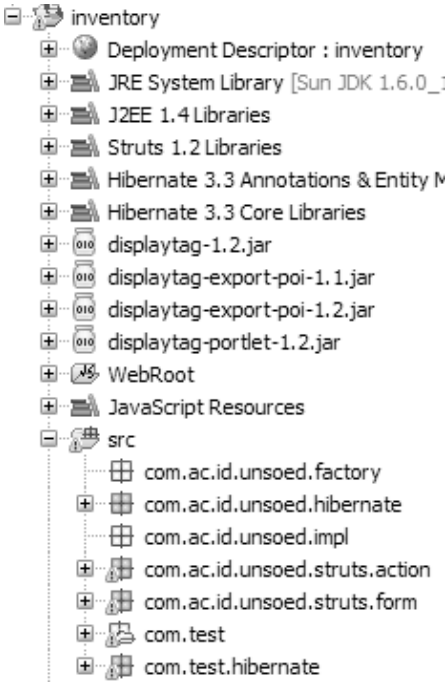


Gambar 1. Arsitektur Hibernate

3. PERANCANGAN DAN IMPLEMENTASI

3.1 Perancangan

Aplikasi peminjaman inventaris program studi informatika Unsoed dikembangkan pada *platform* Java menggunakan rangka kerja (*framework)* Strut dengan basisdata relasional menggunakan Interbase. Gambar 2 memperlihatkan struktur *package* yang terdapat pada kode sumber aplikasi peminjaman inventaris program studi informatika .

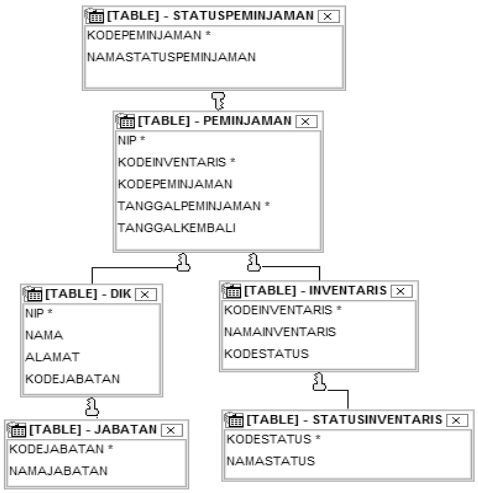


Gambar 2. Struktur *package*

Hibernate akan menggunakan berkas-berkas *class* dan xml yang akan ditampung dalam *package* com.test.hibernate.

3.2 Pemetaan Basisdata

Basisdata aplikasi telah tersedia sebelumnya menggunakan DBMS (*database management system*) Interbase. Gambar 3 memperlihatkan skema basisdata relasional yang digunakan pada aplikasi peminjaman inventaris program studi informatika Unsoed.



Gambar 3. Skema basisdata

Hubungan antara tabel DIK dengan table Inventaris adalah banyak ke banyak (*many to many).* Gambar 4 menunjukan DDL dari table Peminjaman.

```
/* Table: PEMINJAMAN */
CREATE TABLE PEMINJAMAN (
NIP VARCHAR (14) CHARACTER SET NONE NOT NULL
       COLLATE NONE,
KODEINVENTARIS INTEGER NOT NULL,
KODEPEMINJAMAN INTEGER NOT NULL,
TANGGALPEMINJAMAN DATE NOT NULL,
TANGGALKEMBALI DATE);
/* Primary keys definition */
ALTER TABLE PEMINJAMAN ADD CONSTRAINT
       PK_PEMINJAMAN PRIMARY KEY (NIP,
       KODEINVENTARIS, TANGGALPEMINJAMAN);
/* Foreign keys definition */
ALTER TABLE PEMINJAMAN ADD CONSTRAINT
       FK_PEMINJAMAN FOREIGN KEY (NIP)
       REFERENCES DIK (NIP);
ALTER TABLE PEMINJAMAN ADD CONSTRAINT
       FK_PEMINJAMAN1 FOREIGN KEY
       (KODEINVENTARIS) REFERENCES
       INVENTARIS (KODEINVENTARIS);
ALTER TABLE PEMINJAMAN ADD CONSTRAINT
       FK_PEMINJAMAN2      FOREIGN      KEY
       (KODEPEMINJAMAN)          REFERENCES
       STATUSPEMINJAMAN (KODEPEMINJAMAN);
```

Gambar 4. DDL tabel DIK

Tabel Peminjaman adalah tabel yang berfungsi untuk menyimpan data-data peminjaman barang inventaris di programstudi informatika unsoed, satu pegawai dapat meminjam lebih dari satu alat pada satu waktu. Table tersebut kemudian dipetakan ke model objek melalu berkas hbm.xlm seperti digambarkan pada Gambar 5.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD3.0//EN"
"http://hibernate.sourceforge.net/hibernate-
mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.test.hibernate.Peminjaman"
table="PEMINJAMAN">
<composite-id name="id"
class="com.test.hibernate.PeminjamanId">
<key-many-to-one name="dik"
class="com.test.hibernate.Dik">
<column name="NIP" length="14" />
</key-many-to-one>
<key-many-to-one name="inventaris"
class="com.test.hibernate.Inventaris">
<column name="KODEINVENTARIS" length="25" />
</key-many-to-one>
<key-property name="tanggalpeminjaman"
type="java.util.Date">
<column name="TANGGALPEMINJAMAN" length="10"
/></key-property> </composite-id>
<many-to-one name="statuspeminjaman"
class="com.test.hibernate.Statuspeminjaman"
fetch="select"><column name="KODEPEMINJAMAN"
not-null="true" /></many-to-one>
<property name="tanggalkembali"
type="java.util.Date">
<column name="TANGGALKEMBALI" length="10" />
</property> </class></hibernate-mapping>
```

Gambar 5. Berkas peminjaman.hbm.xml

Berkas peminjaman.hbm.xml melakukan pemetaan dari tabel relasional ke *class* Java. *Class* yang dituju pada berkas peminjaman.hbm.xml adalah com.test.hibernate.Peminjaman. *Class* Peminjaman ditunjukan pada Gambar 5.

```
package com.test.hibernate;
import java.util.Date;
public class Peminjaman extends
AbstractPeminjaman implements
java.io.Serializable {
public Peminjaman() {}
/** minimal constructor */
public Peminjaman(PeminjamanId id,
Statuspeminjaman statuspeminjaman) {
 super(id, statuspeminjaman);}
/** full constructor */
public Peminjaman(PeminjamanId id,
Statuspeminjaman statuspeminjaman, Date
tanggalkembali) {
 super(id,statuspeminjaman,
tanggalkembali); }}
```

Gambar 6. *Class* Peminjaman

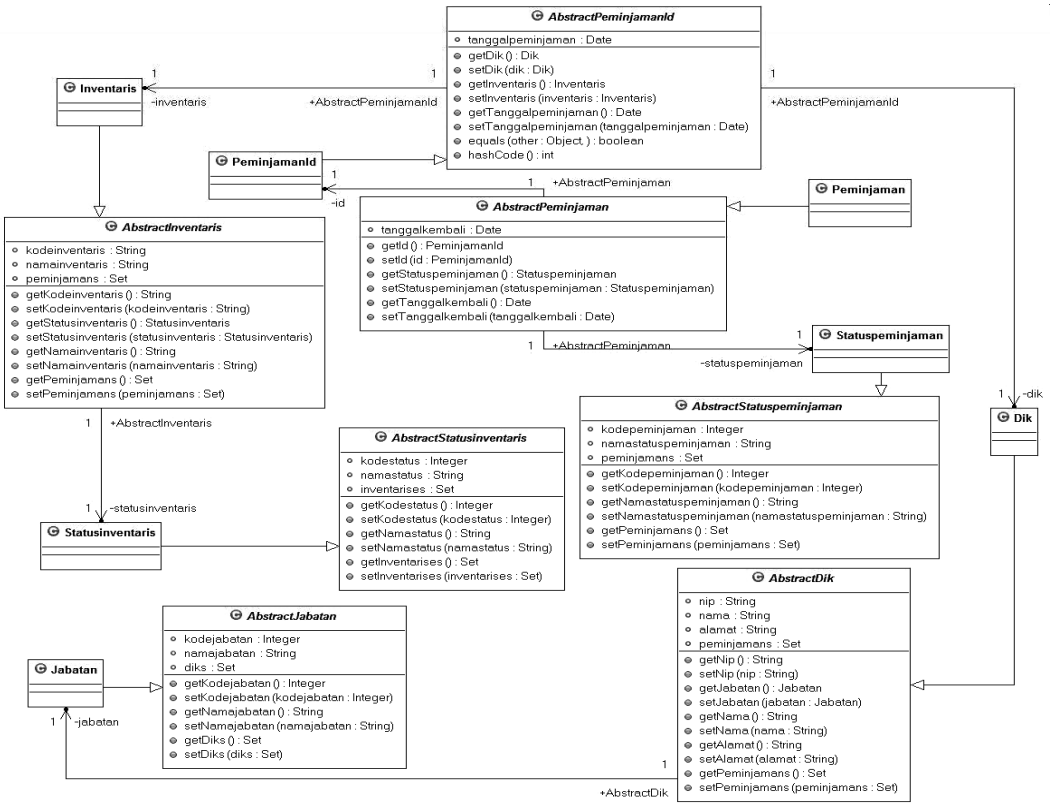
Berkas *class* Peminjaman pada Gambar 6 merupakan turunan dari *class* AbstractPeminjaman. *Class* AbstractPeminjaman merupakan *class* POJO (*plain old java object*), yakni class yang mendefinisikan atribut/*state* serta *method-method, getter* maupun *setter.* Gambar 7 menunjukan potongan kode sumber dari *class* AbstractPeminjaman.

```
public abstract class AbstractPeminjaman
extends ActionForm implements
java.io.Serializable {
  private PeminjamanId id;
  private Statuspeminjaman statuspeminjaman;
  private Date tanggalkembali;
  /** default constructor */
  public AbstractPeminjaman() { }
 // Property accessors
  public PeminjamanId getId() {
        return this.id;   }
  public void setId(PeminjamanId id) {
       this.id = id;  }
  public Statuspeminjaman
  getStatuspeminjaman() {
       return this.statuspeminjaman;  }
  public void
  setStatuspeminjaman(Statuspeminjaman
       statuspeminjaman) {
          this.statuspeminjaman =
          statuspeminjaman;  }
   public Date getTanggalkembali() {
       return this.tanggalkembali;  }
   public void setTanggalkembali(Date
       tanggalkembali) {
          this.tanggalkembali=
          tanggalkembali;  } }
```

Gambar 7. Kodesumber *class* AbstractPeminjaman

Pada Gambar 7 terlihat bahwa *primary key* pada tabel peminjaman dibuat dalam satu *class* yakni *class* PeminjamanId. PeminjamanId adalah *class* yang diturunkan dari *class* AbstractPeminjamanId.

Skema *class* diagram hasil pemetaan basisdata relasional ke model objek diperlihatkan pada Gambar 8.



Gambar 8. *Class* hasil pemetaan dari tabel basisdata relasional

3.3 Implementasi

Setelah pemetaan *class* dilakukan selanjutnya *clas-class* hasil pemetaan tabel basisdata relasional akan digunakan untuk menyimpan data berupa objek. Gambar 9 memperlihatkan tampilan *form* peminjaman inventaris .

Gambar 9. *Form* peminjaman inventaris .

Halaman *form* inventaris tersebut akan memanggil *action* yaitu pinjamInventaris. Gambar 10 memperlihatkan *method* Simpan yang terdapat pada *action* pinjamInventaris.

Pertama-tama seluruh *class* yang diperlukan untuk pemanggilan konstruktor *class* Peminjaman dibuat *instance* (objek) untuk kemudian diisi. *Instance* dari *class* Peminjaman selanjutnya kita sebut sebagai objPeminjaman.

```
public ActionForward Simpan(ActionMapping
mapping, ActionForm form,HttpServletRequest
request, HttpServletResponse response)
throws Exception {
PinjamInventarisForm pinjamInventarisForm =
(PinjamInventarisForm) form;
PeminjamanId key=new PeminjamanId();
Dik dik=new Dik();
Statuspeminjaman statuspeminjaman= new
     Statuspeminjaman();
Inventaris objinventaris=new Inventaris();
objinventaris.setKodeinventaris(pinjamInvent
arisForm.getKodeinventaris());
dik.setNip(pinjamInventarisForm.getNip());
statuspeminjaman.setKodepeminjaman(new
Integer(pinjamInventarisForm.getKodepeminjam
an()));
key.setDik(dik);
key.setTanggalpeminjaman((DateUtils.DateNow(
)));
key.setInventaris(objinventaris);
Peminjaman objPeminjaman=new Peminjaman(key,
statuspeminjaman,
DateUtils.strTodate(pinjamInventarisForm.get
Tanggalkembali()));
PeminjamanDAO z=new PeminjamanDAO();
Transaction tx =
z.getSession().beginTransaction();
z.save(objPeminjaman);
tx.commit();
z.getSession().close();
return this.unspecified(mapping,
pinjamInventarisForm, request, response);}
```

Gambar 10. *Method* simpan pada action pinjam Inventaris .

Penyimpanan dari objek objPeminjaman dilakukan dengan sederhana. Objek akan otomatis terpetakan dan tersimpan ke table basisdata relasional. Dengan menggunakan teknik ini objek yang digunakan dapat langsung disimpan ke basisdata.

4. SIMPULAN

Dari hasil penelitian implementasi Hibernate dapat diambil beberapa kesimpulan sebagai berikut:

1. Hibernate mampu menjadi jembatan antara model objek dengan model basisdata relasional
2. Penggunaan Hibernate mampu menyederhanakan proses penyimpanan dan pemuatan data objek.
3. Penggunaan teknik *object-relational mapping* dapat mempersingkat waktu untuk melakukan pengembangan perangkat lunak.
4. Diperlukan penelitian lanjutan untuk menguji kinerja web server ketika menggunakan hibernate sebagai *object relational mapping.*

PUSTAKA

MyEclipse (2011), *MyEclipse Hibernate Tutorial* Diakses pada 1 april 2011 dari http://www.myeclipseide.com/documentation/quickstarts/hibernate/

Roseindia (2011), *Hibernate Architecture*
Diakses pada 1 april 2011 dari http://www.roseindia.net/hibernate/hibernate_architecture.shtml

Taryana, Acep (2010). Implementasi Basis Data Berorientasi Objek pada Pengembangan Perangkat Lunak. *SESINDO*

Pugh, Eric. dan Gradecki, Joseph D. (2004). *Profesional Hibernate*. Canada: Wiley Publishing Inc.