

# Presentation

about objects and storing

by Arkadiusz Hiler



**JPA 100%**

@Entity



@ManyToOne  
@JoinColumn  
owner

@Column  
colour

@Id  
@GeneratedValue  
id

**Why so ORMious?**

Object



**this presentation is about**

**db4objects**

**BY VERSANT**

~~class must be EJB~~

~~class needs to implement interfaces~~

~~XML configuration~~

~~schema~~

reflection

POJOs

Java && .NET

[dll, jar] ~ 1MB

GNU GPL

commercial licensing (+support)

since year 2000 (now 8.0)

Java ME

# Who is using it?

Boeing

BOSCH

BMW

INDRA Sistemas (Spanish IT and Defense Systems)

Postbank

Seagate & Intel & IBM

**step back for a moment**

# Object-Oriented Database Model

Invention of 1980's.

It's hard to keep complex structures in relational DBs.

Faster relations (no tables, pointers!).

Pointers...

Same representation (language and DB).

-1 layer (ORM-less)

Lacks mathematical model.

There is OQL (but nobody uses it).

**Let's start!**

# Open

```
ObjectContainer database = Db4oEmbedded.openFile(  
    Db4oEmbedded.newConfiguration(),  
    "database.db4o"  
);  
  
try {  
    //queries!  
} finally {  
    database.close();  
}
```

# What happened?

We opened database file.

If file doesn't exist it is created.

There is possibility to connect to remote DB over TCP.

Operations in `try` block. Closing in `finally`.

# Storing & ...

```
database.store(object);
```

```
//retrieve
```

```
object.setFoo(bar);  
database.store(object);
```

```
//retrieve
```

```
database.delete(object);
```

# What you need to know?

Yes, storing is so simple.

Structured objects are saved implicitly (start from top-level).

If you have saved something before explicitly it makes no difference.

You **always** need to retrieve object before update or delete.

Not doing so (ex. sequence of store) will add new objects.

No problem with collections and inheritance.

Retrieves 5 levels of depth by default (then refs are null).

Can fetch more. `database.activate()`. Can be done transparently.

# Transactions

Transaction is started when you open container.

Committed when closing.

You can do `database.commit()`.

There is `database.rollback()`.

`database.ext().refresh()` for refreshing live objects.

# Native Queries

Fun way to search your datase.

You are using your language syntax!

One language less to use/learn/understand.

Create predicate (inner anonymous class with one method).

Rarely executed. Analyzed and compiled to SODA.

```
//get list of matched elements  
List<Type> list = database.query(Predicate<Type> p);  
  
//and predicate?  
new Predicate<Student> {  
    public boolean match(Student s) {  
        return s.getAvg() > 4.5  
            && s.getMawtykiWDziewkanacie();  
    }  
}
```

# SODA

Native Queries is all you need.

In most cases.

Not always.

Under the hood they are translated to SODA.

SODA is db4o internal *language*.

```
//from official tutorial
```

```
//retrieveByConjunction
```

```
Query query=db.query();  
query.constrain(Pilot.class);  
Constraint constr=query.descend("name").constrain("Michael");  
query.descend("points").constrain(99).and(constr);  
ObjectSet result=query.execute();
```

```
//retrieveByDisjunction
```

```
Query query=db.query();  
query.constrain(Pilot.class);  
Constraint constr=query.descend("name").constrain("Michael");  
query.descend("points").constrain(99).or(constr);  
ObjectSet result=query.execute();
```

# What else?

You can create indexes.

You can retrieve instance by internal ID.

Configurable in Object Manner.

OME - GUI for browsing DB & quering.

Groovy, Scala, JRuby...

# Not as great...

Lack of portability.

No full-text indexing -> poor performance.

Be careful with Native Queries! (can be  $O(n)$ ).

# Bad, bad query...

```
final String[] names = {"Hiler", "Kowalski"};
List<Student> result=database.query(
    new Predicate<Student>() {
        public boolean match(Student s) {
            for (String name : names) {
                if (s.getLastName().compareTo(name))
                    return true;
            }
            return false;
        }
    }
);
```

# Que?

Flowgraph is hard to analyze.

Query won't be converted to SODA.

It will actually run query on objects.

Complexity is  $O(n)$ .

They are working on it.

# But!

DB4O has many advantages.

Convenience.

Glory and epicness.

Extremely easy to setup and use.

There are a lot of project where it's enough  
or it's even better solution.

Very low learning curve.

Questions?

# Resources

[developer.db4o.com](http://developer.db4o.com)

[neodatis.org](http://neodatis.org)

[en.wikipedia.org/wiki/JavaDataObjects](http://en.wikipedia.org/wiki/JavaDataObjects)

[en.wikipedia.org/wiki/Object\\_database](http://en.wikipedia.org/wiki/Object_database)

[github.com/ivyl/presentations](https://github.com/ivyl/presentations)

[presentations.hiler.pl](http://presentations.hiler.pl)

[www.cs.iusb.edu/~danav/teach/i310/hw1.html](http://www.cs.iusb.edu/~danav/teach/i310/hw1.html)