

db4objects



db4o
The Open Source Object Database
for Java and .NET

Ogi Pishev
Sydney, 1/18/06

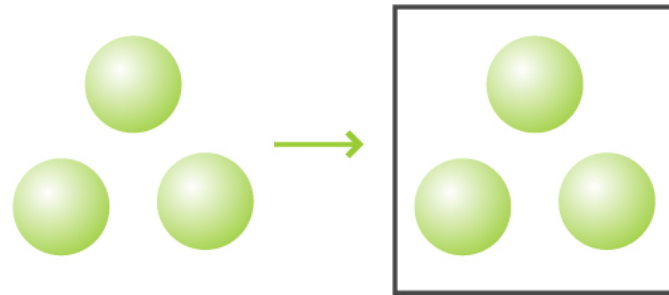


Introduction

Product Features, Benefits

Summary

db4o | Open Source Object Database



- | Native Java and .NET object database engine
- | Ideal to be embedded in products
- | Free GPL and commercial license available

db4o Key Benefits

| **Slashes 90% of cost** to develop persistence

| **10% faster to market** with your application

| **Runs up to 44x faster** than conventional systems

| Deployable in large volumes without local administration

| Build distributed, fully synchronized data architectures

| Build lean and truly object-oriented software

| Fewer errors, better refactorability and software longevity



db4o Key Features: The One-Line-of-Code-Database

- | One line of code stores any object
- | Class model = object schema
- | Smooth production process

```
public void store(Car car){
    ObjectContainer db =
        Db4o.openFile("car.yap");
    db.set(car);
    db.commit();
    db.close();
}
```



Code Demo

| Need to store instances of this class:

```
public class Pilot {
    private String name;
    private int points;

    public Pilot(String name,int points) {
        this.name=name;
        this.points=points;
    }
    ...getters and setters...
}
```

| No persistence code - POJO



Storing Objects

| The ObjectContainer is your primary interface to db4o

```
ObjectContainer db=Db4o.openFile("f1.yap");
try {
    // do something with db4o
}
finally {
    db.close();
}
```



Storing Objects

| To store an object, we simply call `set()` on our database, passing the object as a parameter

```
Pilot pilot1=new Pilot("Jenson Button",99);  
db.set(pilot1);  
System.out.println("Stored "+pilot1);
```

```
Pilot pilot2=new Pilot("David  
    Coulthard",100);  
db.set(pilot2);  
System.out.println("Stored "+pilot2);
```



Retrieving Objects

- | Basic query method is Query-by-example (QBE)
- | To retrieve objects, create prototype object and call `get()` on our database, passing the prototype object as a parameter
- | More advanced query mechanisms also available
-> Native Queries (Carl Rosenberger)

```
Pilot proto=new Pilot(null,0);
ObjectSet result=db.get(proto);
while(result.hasNext()) {
    System.out.println(result.next());
}
```



Updating Objects

- | Retrieve objects, make changes to object and call set() to store updated object
- | Deleting similar, except use delete() method

```
ObjectSet result=db.get(new Pilot("Jenson  
Button",0));  
Pilot found=(Pilot)result.next();  
found.addPoints(10);  
db.set(found);  
System.out.println("Added 11 points for  
"+found);
```



Storing Structured Objects

- | To store a car with its pilot, we just call set() on our top level object, the car
- | The pilot will be stored implicitly
- | Retrieving car will also retrieve pilot
- | Structured objects can also contain collections/arrays

```
Car car1=new Car("BAR Honda");  
Pilot pilot1=new Pilot("Jenson Button",100);  
car1.setPilot(pilot1);  
db.set(car1);
```



Structured Objects

| To store this data in a relational database, you would need to:

- | Map car and pilot classes to separate tables
- | Define key relationship between tables
- | Create separate INSERT statements for car and pilot

| To retrieve this data from a relational database, you would need to:

- | Create SELECT statement with join
- | Marshall data to construct car object
- | Marshall data to construct pilot object



Inheritance

| Can easily store and retrieve classes and subclasses

```
public class WorldChampion extends Pilot{  
    ...  
  
    Pilot pilot1=new WorldChampion("Michael  
        Schumacher",110, 7);  
    db.set(pilot1);  
    System.out.println("Stored "+pilot1);  
}
```



Inheritance

| To store this object in a relational database, you would need to:

- | Map classes and subclasses to table(s)
- | May need to distribute attributes of subclass between tables
- | Define key relationship between table(s)
- | Create INSERT statement(s)

| To retrieve this object from a relational database, you would need to:

- | Create SELECT statement(s)
- | May need to join tables to get all attributes
- | Marshall data to construct object



Transactions

- | Implicitly using transactions
- | Committed when ObjectContainer is closed
- | Can also explicitly commit and rollback transactions

```
db.commit();
```

```
db.rollback();
```

Introduction

Product Features, Benefits

Summary

Client/Server

- | Can use db4o as a client-server database
- | Handle concurrently executing transactions
- | Embedded server
- | Network server



Summary of db4o Features and Benefits

Key features:

- | The One-Line-of-Code-Database
- | Embeddable
- | Zero Administration
- | Multiple platform support
- | Brings OO development to the database

Key benefits:

- | Slashes 90% of cost to develop persistence
- | 10% faster to market with your application
- | Runs up to 44x faster than conventional systems

Introduction

Product Features, Benefits

Summary



Thank You



download now

| More information and free download: www.db4o.com