



# **OSGi Alliance Community Event**

## **A Persistence Service for the OSGi framework**

Carl Rosenberger  
Chief Software Architect  
db4objects Inc.



# What is an object?

- ?



## What is an object?

- a conceptual unit with
  - Identity
  - State
  - Behaviour
- an instance of a class
- `Car car = new Car("Ferrari");`



## Where do objects come from?

- `Car car = new Car("Ferrari");`
- from user entry
- from machine generated data
- over the network
- from a database



## “Databases”

- Flat files (Roll Your Own)
- Java Serialization
- `INSERT INTO car(name)VALUES("Ferrari")`



## “Databases”

- Flat files (Roll Your Own)
  - Is your development team experienced at writing database engines?
  - Will flat files be failsafe and will they provide ACID transactions?
  - Will you have querying functionality?
  - How much will the development cost?
  - How long will it take until the system stable?



# “Databases”

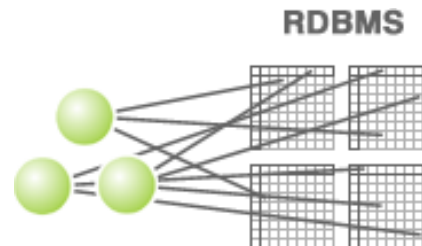
- Java Serialization
  - will break upon modifications to classes
  - requires loading complete graphs of objects to memory
  - is not transactional
  - does not provide querying functionality
  - is explicitly **not recommended** for longterm persistence by the Java Language Specification



## “Databases”

- `INSERT INTO car(name)VALUES ("Ferrari")`
- Is SQL the best choice for storing objects?

*Using tables to store objects is like driving your car home and then disassembling it to put it in the garage. It can be assembled again in the morning, but one eventually asks whether this is the most efficient way to park a car.*



*Esther Dyson*



## How much SQL do you want to write?

- `public class Car {`
- `Colour colour;`
- `Motor motor;`
- `List<Door> doors;`
- `List<Wheel> wheels;`
- `Brake brake;`
- `/*`
- `[ 50 more fields here ]`
- `*/`
- `}`



## Is there an easier way to store objects?

- `Car car = new Car("Ferrari");`



## Is there an easier way to store objects?

- `Car car = new Car("Ferrari");`
- `something.store(car);`



## Is there an easier way to store objects?

- `Car car = new Car("Ferrari");`
- `database.store(car);`
- Simply
  - store objects to the database
  - get objects back from the database
- Persistence by reachability
- Database engine can analyze class schema



## What about queries?

- `SELECT * FROM  
    car, car_brake, brake, car_motor, motor`
- `WHERE car.id = car_brake.car_id`
- `AND brake.id = car_brake.brake_id`
- `AND car.id = car_motor.car_id`
- `AND motor.id = car_motor.motor_id`
- `AND brake.type = 'Ceramic'`
- `AND motor.power > 400;`



## Is there an easier way to query for objects?

- `car.brake.type == "Ceramic"`
- `&& car.motor.power > 400;`



# Introducing Native Queries

- `return car.brake.type == "Ceramic"`
- `&& car.motor.power > 400;`



## Introducing Native Queries

- `public boolean match(Car car){`
- `return car.brake.type == "Ceramic"`
- `&& car.motor.power > 400;`
- `}`



## Introducing Native Queries

- `new Predicate <Car>() {`
- `public boolean match(Car car) {`
- `return car.brake.type == "Ceramic"`
- `&& car.motor.power > 400;`
- `}`
- `}`



## Introducing Native Queries

- `List<Car> cars =`
- `database.query(new Predicate <Car>(){`
- `public boolean match(Car car){`
- `return car.brake.type == "Ceramic"`
- `&& car.motor.power > 400;`
- `}`
- `});`



## Outlook: Native Queries using Closures

- `List<Car> cars =`
- `database.query( {`
- `Car car =>`
- `car.brake.type == "Ceramic"`
- `&& car.motor.power > 400;`
- `}`
- `);`



## Object-Oriented Persistence

- 100% pure Java
- 100% refactorable
- 100% typesafe
- 100% checked at compile-time
- no O-R impedance mismatch
- minimum code
- maximum performance



# Object-Oriented Persistence

- Store Objects
- Native Queries



# Introducing db4o

- **database**
- 4
- **objects**



# Introducing db4o

- **d**atabase
- **4**
- **OSGi**



## Introducing db4o

- Plain Object Persistence
- Zero Administration
- Automatic Schema Management
- Optimized Native Queries
  - compile time
  - load time
  - run time
- OSGi Service Interface
- OSGi ClassLoader aware



## Introducing **db4objects**

- Open Source Project db4o
- GPL License
- Registered Community of 20,000 developers



## Introducing db4objects

- Commercial db4o Licenses
- Customers include Boeing, RICOH, Bosch, Indra





## db4o for OSGi

- db4objects is a member of the OSGi alliance
- Dedicated db4o version for OSGi
- Partnership with ProSyst
  - ProSyst bundles db4o with mBedded Server





**Thank You!**



**download now**

**<http://www.db4o.com/OSGi>**