OJB - Features

by Thomas Mahler, Armin Waibel

Table of contents

1. Features

- Support of standard and non-standard API's:
  - PB api (non-standard)
  - OTM api (non-standard)
  - ODMG api (standard)
  - JDO api (standard)
- The PersistenceBroker kernel api and all top-level api (ODMG, OTM, JDO) allows Java Programmers to store and retrieve Java Objects in/from (any) JDBC-compliant RDBMS
- Transparent persistence: Persistent classes don't have to inherit from a persistent base class or to implement an interface.
- Scalable architecture that allows to build massively distributed and clustered systems.
- Configurable persistence by reachability (docu/guides/basic-technique.html) : All Objects associated to a persistent object by references can made persitent too.
- Extremly flexible design with pluggable implementation of most service classes like *PersistenceBroker*, *ObjectCache*, *SequenceManager*, *RowReader*, *ConnectionFactory*, *ConnectionManager*, *IndirectionHandler*, *SQLGenerator*, *JdbcAccess*, ... and so on.
- Quality assurance taken seriously: More than *600 JUnit-TestCases* (docu/testing/testsuite.html) for regression tests. JUnit tests integrated into the build scripts.
- Mapping support for 1:1, 1:n and m:n associations (docu/guides/basic-technique.html) .
- Configurable collection queries to control loading of relationships. See QueryCustomizer (docu/guides/advanced-technique.html) .
- Automatic (docu/guides/basic-technique.html#cascading) and manual (docu/guides/basic-technique.html#linking) assignment of foreign key values.
- The Object / Relational mapping is defined in an XML Repository. The mapping is completely dynamic and can be manipulated at runtime (docu/guides/metadata.html) for maximum flexibility
- Easy use of multiple databases (docu/faq.html#multipleDB) .
- Configurable Lazy Materialization through Proxy (docu/guides/basic-technique.html#using-proxy) support in the PersistenceBroker. The user can implement specific Proxy classes or let OJB generate dynamic Proxies.
- Support for Polymorphism and Extents (docu/guides/advanced-technique.html#polymorhism) . You can use Interface-types and abstract classes as attribute types in your persistent classes. Queries are also aware of extents: A query against a baseclass or interface will return matches from derived classes, even if they are mapped to different DB-tables
- Support for Java Array- and Collection-attributes in persistent classes. The attribute-types (docu/guides/advanced-technique.html#types-for-associations) can be Arrays, java.util.Collection or may be user defined collections that implement the interface

`ojb.broker.ManageableCollection`.

- Sequence-Managing (docu/guides/sequencemanager.html) . The SequenceManager is aware of "extents" and maintains uniqueness of ids accross any number of tables.
  Sequence Numbering can be declared in the mappping repository.
  Native Database based Sequence Numbering is also supported.
- Reusing Prepared Statements, internal connection pooling (docu/guides/connection.html) .
- Integrates smoothly in controlled environments like EJB containers (docu/guides/deployment.html#j2ee-server)
- Full JTA and JCA (in progress) Integration.
- Support for prefetched relationships (docu/guides/query.html#prefetched-relationships) to minimize the number of queries.
- ODMG compliant API, a Tutorial (docu/tutorials/odmg-tutorial.html) , and TestCases are included.
- JDO 1.0.1 compliant API (based on jdori, native implementation in progress), a Tutorial (docu/tutorials/jdo-tutorial.html) , and TestCases are included.
- The Lockmanagement (docu/guides/lockmanager.html) supporting four pessimistic Transaction Isolation Levels (uncommited or "dirty" reads, commited reads, repeatable reads, serializable transactions) - distributed locking is possible.
- Optimistic locking (docu/guides/lockmanager.html#optimistic-locking) support. Users may declare `int` or `long` fields as version attributes or `java.sql.Timestamp` fields as timestamp attributes.
- Support for persistent object caching. Different caching strategies and distributed caches (docu/guides/objectcache.html#distributed-cache) .
- Comes along with fully functional demo applications running against HSQLDB.
- Provides Commons-Logging (http://jakarta.apache.org/commons/logging/) and Log4J (http://logging.apache.org/log4j/) logging facilities.
- 100 %: pure Java, Open Source, Apache License

Note:

- OQL is currently not fully implemented (Aggregations and Method Invocations)
- ODMG implicit locking is partly implemented but does currently not maintain transaction isolation properly. To achieve safe transaction isolation client application must use explicit lock acquisition