The JDO Persistence Model

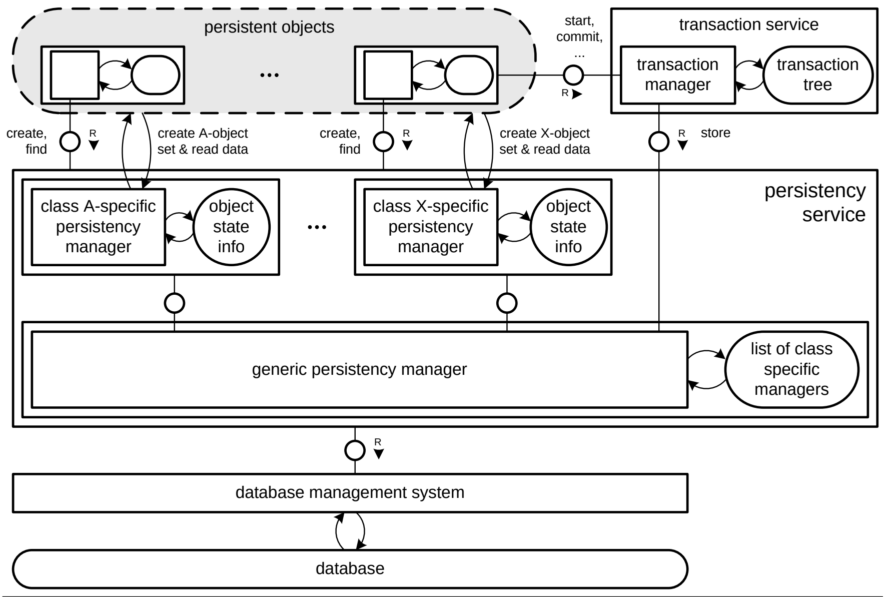
Stefan Marr

Agenda

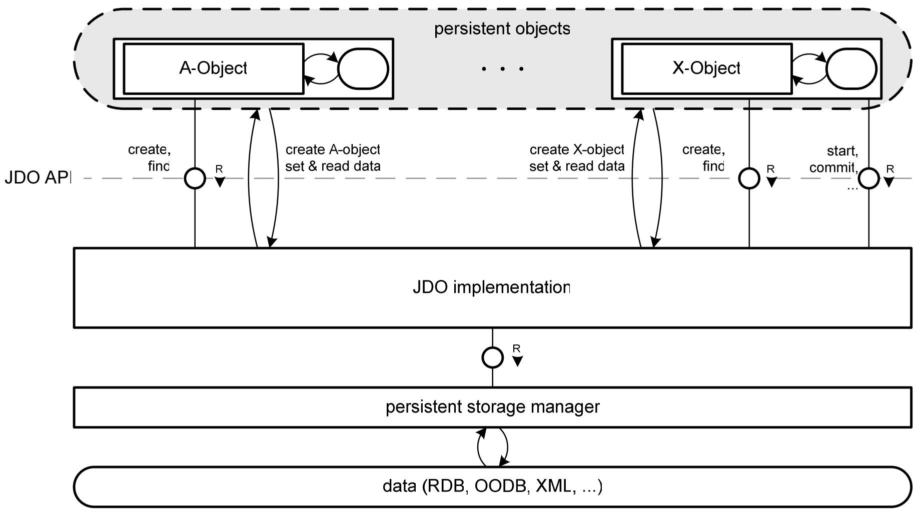
What is JDO?

- Java Data Objects
- interface-based definition of object persistence
- describes
  - storage
  - querying
  - retrieval
- transparent features
  - mapping of JDO instances to data storage
  - transparent to Java objects being persisted
  - independent to storage type
- implicit updates of persisted object states

Persistency of an application using a relational DBMS


persistent objects
...
start,
commit,
...
R
transaction service
transaction manager
transaction tree
create, find
R
create A-object set & read data
create, find
R
create X-object set & read data
R
store
persistency service
class A-specific persistency manager
object state info
...
class X-specific persistency manager
object state info
generic persistency manager
list of class specific managers
R
database management system
database

Persistency of an application using JDO


persistent objects
A-Object
. . .
X-Object
create, find
R
create A-object set & read data
create X-object set & read data
create, find
R
start, commit, ...
R
JDO API
JDO implementation
R
persistent storage manager
data (RDB, OODB, XML, ...)

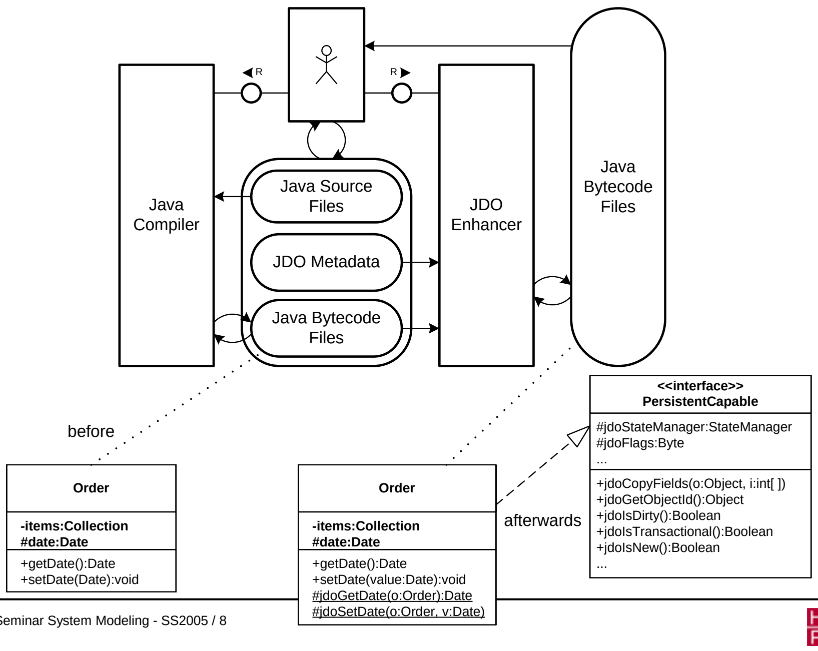
Benefits

- **Reduced modeling efforts**
  - **focus on domain specific aspects**
  - **transparent persistence by automated enhancement**
  - **exploit object-oriented capabilities of Java without any limitation**
- **Abstraction of specific data storage**
  - **exchange of data storage without recompile**
  - **accessing persistent data with object model information only**
  - **no knowledge of SQL, JDBC or underlying data store necessary**
- **Useable in co-operation with other J2EE technologies**

Non-Managed and Managed Environments

- Non-Managed Environment
  - typical two-tier application
  - direct connection to resources
  - developer responsible for interactions with persistence services
    - configuring
    - invoking
- Managed Environment
  - typical J2EE-based multi-tier application
  - J2EE container responsible
    - pooling of the persistence service
    - transactions
    - configuration done declaratively

The Class Enhancement

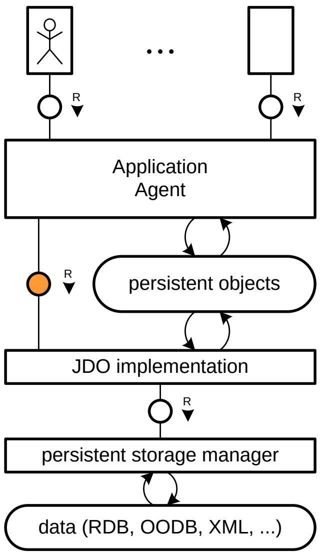


The Class Enhancement

- JDO implementation provides enhancer tool
  - modifies data classes
  - metadata identifies data classes
  - describes managed fields
- Effects of Enhancement
  - `PersistenceCapable` interface added
  - a few methods and fields
  - new Getter and Setter for managed fields
  - access to managed fields changed to Getter/Setter usage
  - don’t use added interface, methods, and fields
- Enhancement Metadata
  - XML-Files
  - identifies and describes the application data classes
  - used by enhancer and JDO runtime
    - don‘t change after enhancement, behavior unspecified

Data access with JDO

- working with persistent objects
  - Persistence Manager
  - Transactions
  - Object Identities
  - Object Lifecycle
- access persistent objects
  - Extends
  - JDO-QL



Persistence Manager

- main interface for application developer
- for handling `PersistenceCapable` objects
  - added to persistent classes by enhancement
- state management methods
  - `makePersistent()`, `deletePersistent()`, `makeTransient()`, `evict()`, `refresh()`, etc.
- obtaining Query, Extent, and Transaction objects
- get from `PersistenceManagerFactory` by JDOHelper, JNDI, ...
  - `JDOHelper.createPersistenceManagerFactory(Properties p)`
  - `PersistenceManager getPersistenceManager()`

```
<<interface>>
PersistenceManager
---------------------------------------------------
close()
currentTransaction():Transaction
deletePersistent(pc:Object)
deletePersistentAll(pcs:Collection)
evict(pc:Object)
evictAll()
getExtent(pcc:Class, sub:boolean):Extent
getObjectById(oid:Object, val:boolean):Object
getObjectId(pc:Object):Object
getObjectIdClass(cls:Class):Class
getPersistenceManagerFactory():
            PersistenceManagerFactory
getTransactionalObjectId(pc:Object):Object
isClosed():boolean
makeNontransactional(pc:Object)
makeNontransactionalAll(pcs:Collection)
makePersistent(pc:Object)
makePersistentAll(pcs:Collection)
makeTransactional(pc:Object)
makeTransactionalAll(pcs:Collection)
makeTransient(pc:Object)
makeTransientAll(pcs:Collection)
newObjectIdInstance(pcc:Class, str:String):Object
newQuery():Query
refresh(pc:Object)
refreshAll()
retrieve(pc:Object)
retrieveAll(pcs:Collection)
```

Transactions

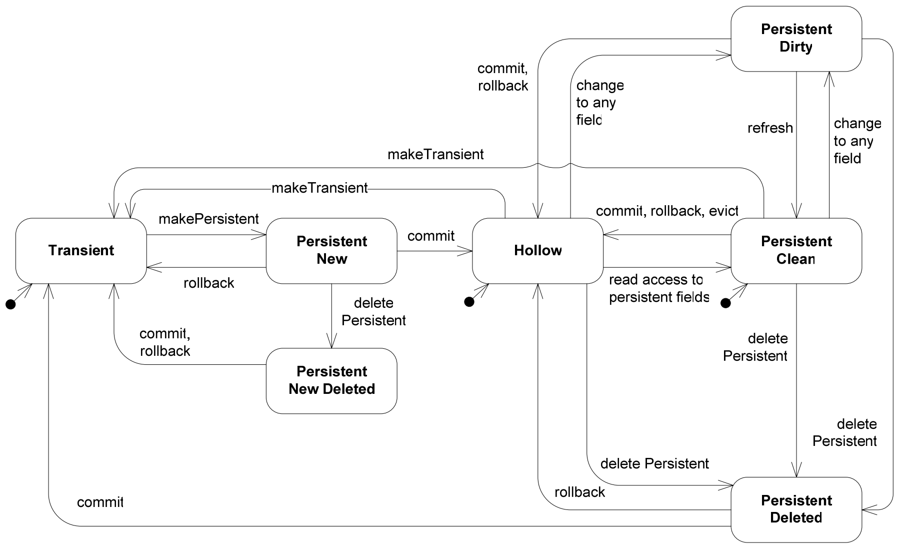
- ACID, atomic, ...
- not rely on isolation level greater than Read Committed
- at most one per `PersistenceManager`
- pooling used in J2EE environments
- Transaction Strategies
  - Pessimistic
    - default
    - suitable for short-living transactions
    - exclude other transactions from accessing data
  - Optimistic
    - for long-living transactions
- obtained from `PersistenceManager`
  - `Transaction currentTransaction()`
  - `begin()`, `commit()`, `rollback()`

```
<<interface>>
Transaction
---------------------------------------------------
begin()
commit()
rollback()

isActive():boolean

getNontransactionalRead():boolean
getNontransactionalWrite():boolean
getOptimistic():boolean
getPersistenceManager():
    PersistenceManager
getRestoreValues():boolean
getRetainValues():boolean
getSynchronization():
    javax.transaction.Synchronization

setNontransactionalRead(v:boolean)
setNontransactionalWrite(v:boolean)
setOptimistic(v:boolean)
setRestoreValues(v:boolean)
setRetainValues(v:boolean)
setSynchronization(sync:javax.
    transaction.Synchronization)
```

Object Identities

- Java: identity vs. equality
- Datastore Identity
  - default, nature internal by JDO implementation
  - corresponds to primary-key in RDBS
  - unique id internally handled by JDO
- Application Identity
  - handled by application, simple or compound primary-key
  - own Key-Class necessary
  - mentioned in JDO Metadata
- Non-durable JDO Identity
  - for objects without own identity necessary

Object Lifecycle

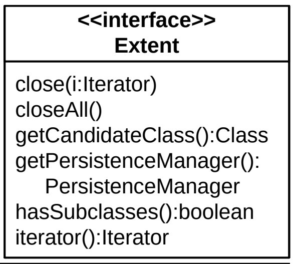


Object Lifecycle

- Transient
  - created with new(), without identity and not persisted
  - no transactional behavior
- Persistent-New
  - object made persistent during transaction
    - saves persistent and transac. non-pers. field values for rollback
    - assigns a JDO identity
- Persistent-New-Deleted
  - made persistent and be deleted within current transaction
- Hollow
  - persistent object, id loaded, data not accessed
- Persistent-Clean
  - data read, but not altered
- Persistent-Dirty
  - data changed or call to `makeDirty()` of `JDOHelper`
- Persistent-Deleted, deleted in current transaction

Extents

- represents complete set of all persistent instances of a class
- method of `PersistenceManager`
  - `public Extent getExtent(Class c, boolean getSubclasses)`
- provides an `Iterator`
  - data retrieval process started on first `next()`
  - no filtering, only decision whether Subclasses or not
- primary purpose
  - provide candidate collection of objects to a query
  - query uses extents
    - to produce an equivalent query in native datastore language
    - apply filter


<<interface>>
Extent

close(i:Iterator)
closeAll()
getCandidateClass():Class
getPersistenceManager():
PersistenceManager
hasSubclasses():boolean
iterator():Iterator

JDO-Query Language

- abstracts from datastore language
- capable of optimizations to underlying technology

- obtained from `PersistenceManager`
  - `Query newQuery(Extent cln, String filter)`
    - new query with the candidate class from Extent
- Filter
  - `“attrName == \"string\"“`
  - Supported Operators: !, &&, ||, <, >, ==, …
  - Methods: `isEmpty()`, `contains(Object o)`, `startsWith(String s)`, `endsWith(String s)`
- Ordering by `setOrdering(String s)`
- JDO-QL limited to basics in JDO 1.0.1

```
<<interface>>
Query
-----------------------------------
close(qr:Object)
closeAll()
compile()
declareImports(imports:String)
declareParameters(paras:String)
declareVariables(variables:String)
execute():Object
getIgnoreCache():boolean
getPersistenceManager():
            PersistenceManager
setCandidates(pcs:Collection)
setCandidates(pcs:Extent)
setClass(cls:Class)
setFilter(filter:String)
setIgnoreCache(v:boolean)
setOrdering(ordering:String)
```

Summary and outlook on JDO 2.0

- interface-based definition of object persistence
- reduced modeling efforts
- abstraction of specific data storage
- accessing persistent data with object model information only
- transparency by code enhancement

- `PersistenceManager`, `Transaction`, `Extent`, `Query`
- JDO Metadata in XML-files

- JDO 2.0
  - compatible with JDO 1.0, still binary compatible
  - enhanced JDO-QL, projections, aggregates, simplified programming of queries, paging of results
  - standardized mapping to relational databases
  - detached objects, for multi-tier application programming

References

- JDO 1.0
  - ‘Using and Understanding Java Data Objects’, David Ezzio, Apress, 2003
  - ‘Java Data Objects’, Robin M. Roos, Addison Wesley, 2003
  - ‘Java Data Objects’, David Jordan & Craig Russell, O’Reilly, 2003
  - ‘Java Data Objects in der Praxis’, Andreas Holubek, Javamagazin 06/2004
  - http://jcp.org/aboutJava/communityprocess/final/jsr012/index2.html JDO 1.0.1 specification
- JDO 2.0
  - http://www.theserverside.com/articles/article.tss?l=JDO2-Kickoff
  - http://www.jdocentral.com/JDO_Commentary_CraigRussell_4.html
- Additional Resources
  - http://www.jpox.org/
    - open source JDO implementation
    - will become JDO 2.0 Reference Implementation
  - http://www.jdocentral.com/
  - http://java.sun.com/products/jdo/javadocs/

The JDO Persistency Model

Q & A

Stefan Marr