EXPERLOG

XAPool 1.2.2

XAPool is Another Pool :-)

http://xapool.experlog.com
http://forge.objectweb.org/projects/xapool
email: xapool-public at lists.debian-sf.objectweb.org

Xavier.Spengler@experlog.com

July 1st, 2003

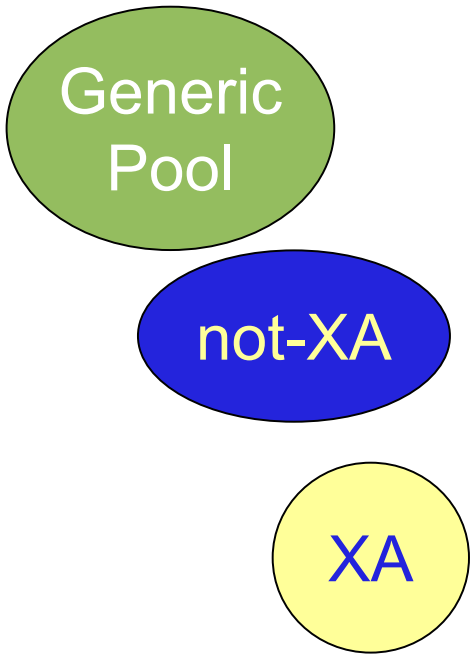
History

History:

-2000: Lutris asks for a new Database Manager for JOnAS
-2003: Experlog proposes XAPool as a new Enhydra project to the Objectweb Community
-in the future: ...

License:
EPL: Enhydra Public License

Table of Content

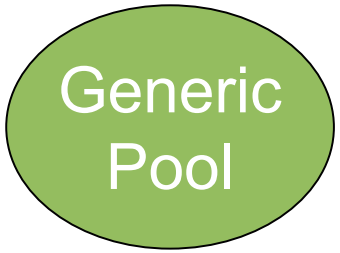


XAPool

What is XAPool ?

XAPool is a software component which allows to:
-Store objects with a Generic Pool
-Export a DataSource[1]
-Export a XADataSource[2]

[1] javax.sql.DataSource
[2] javax.sql.XADataSource



Generic Pool

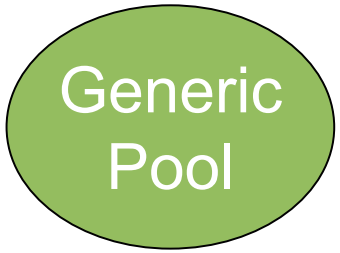
What can I store in the Generic Pool ?

Answer: anything!

You just need to implement an interface and give it to the Generic pool.

With this interface, the pool will be able to create your objects.

The pool exports the following methods:
-create(s) methods to build your objects
-expire method to destroy properly your objects
-check(s) methods to verify your objects before using them !
(5 levels of object checking)



Generic Pool (2)

Functionalities and properties of the Generic Pool

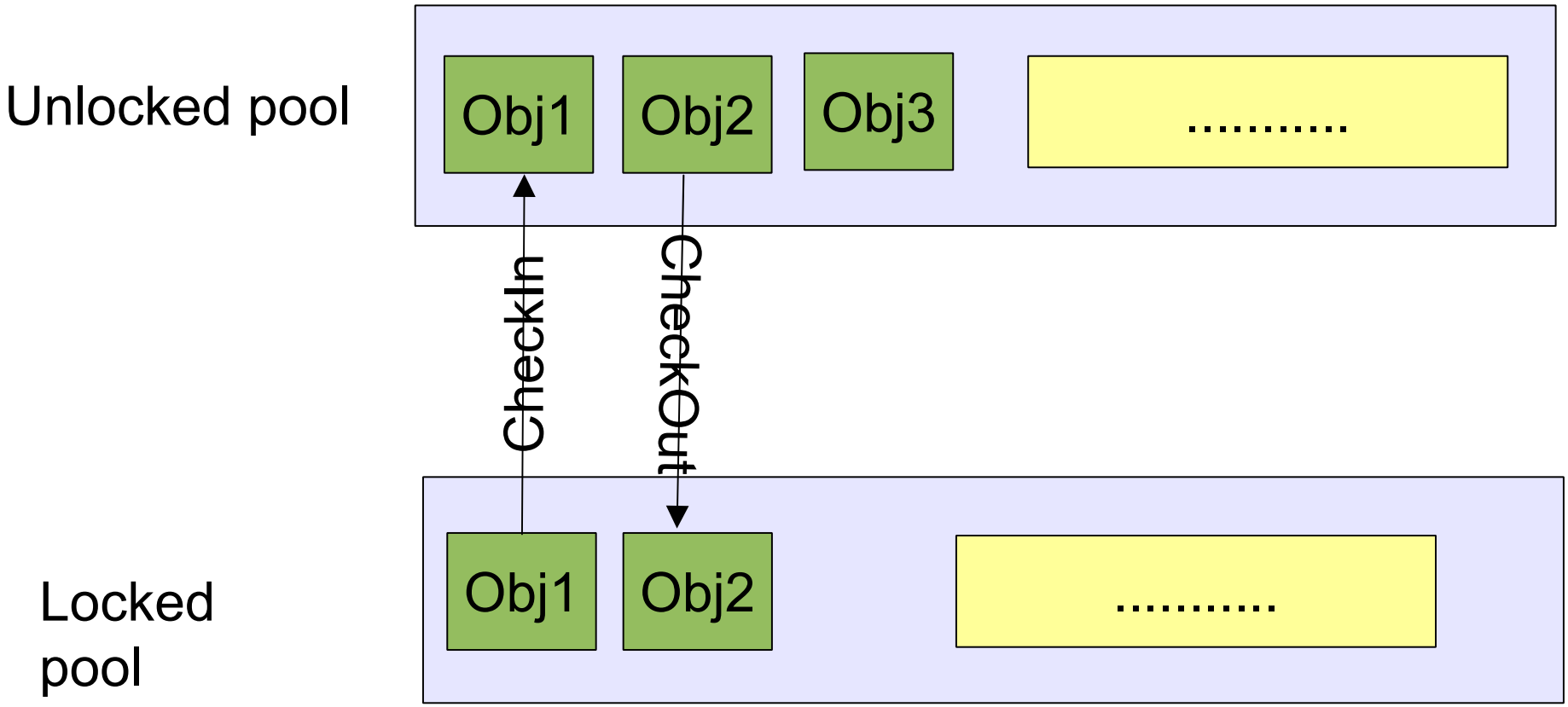
-minimum pool size (and initial number of pooled objects)
-maximum pool size
-log writer (with commons-logging[1])
-life time of objects (lifetime in the free objects pool)
-time to wait until an object (eg. a Connection) is available
-garbage collector option(run gc upon pool cleanup)
-”generation” concept when problems occurs on objects (upon error, check all objects of current or previous generation, with optional reset).

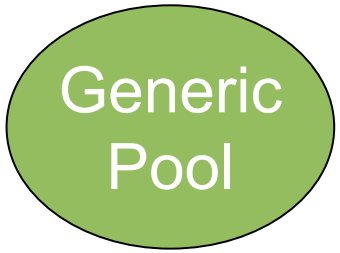
[1] http://jakarta.apache.org/commons/logging.html

Generic Pool

Pool internals

2 pools inside the Generic Pool:





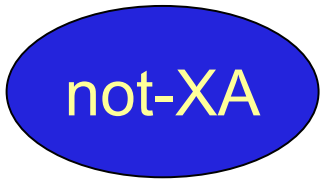
Main methods

How to communicate with the Generic Pool ?

-start method to initialize the pool
-checkout method to get an object from the pool
-checkin method to return an object to the pool
-stop method to delete all objects and pool activity
-administrative methods (eg. get/set min/max size…)

The Generic Pool is used by :
-the PooledConnection[1] factory
-the XAConnection[2] factory

[1] javax.sql.PooledConnection
[2] javax.sql.XAConnection



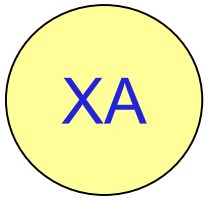
Datasource

The PooledConnection Factory: the Datasource
(StandardPoolDataSource[1])

This object is able to build PooledConnection objects and to store them into the pool.

Main methods from DataSource are:
-Connection[2] getConnection()
-Connection getConnection(String user, String password)
-PrintWriter[3] methods (setter/getter) for logging
-timeOut methods (setter/getter)

[1] org.enhydra.jdbc.pool.StandardPoolDataSource
[2] java.sql.Connection
[3] java.io.PrintWriter



XA DataSource

The XAConnection Factory: the XADataSource
(StandardXAPoolDataSource[1])

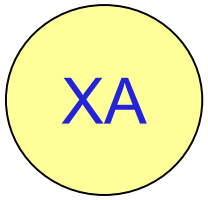
-This object is able to build XAConnection objects and to store them into the pool.

-This object extends javax.sql.DataSource to use common getConnection method.

-create and close (expire) methods are specific for each object.

-A TransactionManager[2] (JOTM) is used to enlist and delist XAResource[3].

[1] org.enhydra.jdbc.pool.StandardXAPoolDataSource
[2] javax.transaction.TransactionManager
[3] javax.transaction.xa.XAResource



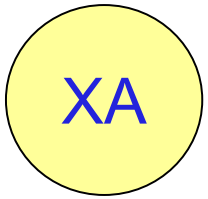
XAResource

XAConnection as XAResource:

Our implementation of XAConnection implements XAResource[1] interface:
-association between Transaction and the database

Main methods of XAResource are:
-commit, rollback, prepare (with a specific Xid[2])
-start, end, forget (with a specific Xid)
- ...
Xid = « X id » or « transaction id », given by the Transaction Manager

[1] java mapping of the industry standardXA interface based on the X/Open CAE Specification
[2] javax.transaction.xa.Xid

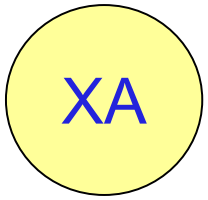


XA Statements

(XA)PrepareStatement[1] and (XA)Statement[2]:

-These 2 implementations are wrappers to core implementations.
-These objects are used to enlist resources ONLY when connections are used !
-Implements a cache of PreparedStatement

[1] java.sql.PrepareStatement
[2] java.sql.Statement



DB wrappers

Specific development for Databases:

Oracle: a special wrapper to XAConnection and XADataSource objects from Oracle driver (jdbc2.0 driver) exists to pool these connections.

Informix, Sybase: special wrappers have been done to hack some functionalities

InstantDB: a special wrapper has been done to support a full JDBC XA driver.

PostgreSQL, MySQL, SAPdb, HSQL,...: it works as is

External libs

Externals requirements:

Mains are:
-jotm.jar: Transaction Manager
-commons_logging.jar and log4j.jar: Logging
-rmiregistry: JNDI, to bind and lookup transaction, XADataSource and XAPoolDataSource

-other useful libs: carol.jar, idb.jar, jonas_timer.jar, jotm_jrmp_stubs.jar, jta-spec1_0_1.jar, p6spy.jar ...

Success stories

Where I can find Success Stories:
-ExperSHOP: Experlog's product embeds XAPool since 2001 (the PooledConnection part)
-JOTM: uses XAPool as XA Pool DataSource
-JettyPlus[1]: XAPool and JOTM are integrated into JettyPlus from MortBay.org

Downloads:
May: 36
June: 67

Recent news:
-XAPool works with C-JDBC for database clustering.

[1] http://jetty.mortbay.org/jetty/plus

```
// create an XA pool datasource with a minimum of 4 objects
StandardXAPoolDataSource spds = new StandardXAPoolDataSource(4);
spds.setUser(login);   spds.setPassword(password);
Jotm jotm = new Jotm(true, false);
spds.setTransactionManager(jotm.getTransactionManager());

// create an XA datasource which will be given to the XA pool
StandardXADataSource xads = new StandardXADataSource();
try {
  xads.setDriverName(driver);
  xads.setUrl(url);
  xads.setUser(login);
  xads.setPassword(password);
} catch (Exception e) { System.err.println("JOTM problem."); }

// give the XA datasource to the pool (to create future objects)
spds.setDataSource(xads);
```

```
Connection conn = spds.getConnection(login, password);
try {
  UserTransaction utx = jotm.getUserTransaction();
  utx.begin();
  PreparedStatement pstmt0 = conn.prepareStatement(SQL_QUERY);
  pstmt0.setInt(1, 13);
  pstmt0.executeUpdate();
  utx.commit();

  utx.begin();
  PreparedStatement pstmt = conn.prepareStatement(SQL_QUERY2);
  pstmt.setInt(1, 14);
  pstmt.executeUpdate();
  utx.rollback();
} catch (Exception e) {
  System.err.println("Exception");
}
conn.close();
```

XAPool 1.2.2

Conclusion:

-Reusable Generic Pool
-Fully (100%) compliant to the specification
-Real time instrumentation
-Now used in several products (yes, there was a need!)

To be done:

-reuse of the Generic Pool for other products
-implementation of Connector architecture
-JOnAS, back to the source :-)
-JMX ?