Platforms

by Thomas Mahler

1. how to use OJB with a specific relational database

OJB has been designed to smoothly integrate with any relational database that provides JDBC support. OJB can be configured to use only JDBC 1.0 API calls to avoid problems with restrictions of several JDBC drivers.
It uses a limited SQL subset to avoid problems with restrictions of certain RDBMS. This design allows to keep the OJB code generic and free from database specifics.

This document explains basic concepts and shows how OJB can be configured to run against a specific RDBMS.

If you not already have done so, then you also might want to have a look at the Getting Started section which presents a sample skeleton project.

2. Basic Concepts

2.1. OJB internal tables

For certain features OJB relies on several internal tables that must be present in the target rdbms to allow a proper functioning. If those features are not needed OJB can be safely run without any internal tables.

The following table lists all tables and their specific purpose.

Tablename	Purpose
`OJB_HL_SEQ`	Table for the high/low sequence manager. If the built-in OJB sequence manager is not used, this table is not needed.
`OJB_LOCKENTRY`	This table is used to store Object locks if the LockManager is run in distributed mode. Not needed in singlevm mode.
`OJB_NRM`	The "Named Roots Map". ODMG allows to bind persistent objects to an user defined name. The Named roots map is used to store these bindings. It has NAME (String of arbitrary length) as primary key and keeps the serialized OID of the persistent object in the field OID (String of arbitrary length). If bind() and lookup() are not used in client apps, this table is not needed
`OJB_DLIST`	The table used for the ODMG persistent DList collections. If ODMG DLists are not used, this table is not needed.
`OJB_DLIST_ENTRIES`	stores the entries of DLists (a wrapper to objects stored in the DList)

Copyright © All rights reserved.

	If ODMG DLists are not used, this table is not needed.
OJB_DSET	The table used to store ODMG persistent DSET collections If ODMG DSets are not used, this table is not needed.
OJB_DSET_ENTRIES	This table stores the entries of DSets. If ODMG DSets are not used, this table is not needed.
OJB_DMAP	The table use to store the ODMG persistent DMap tables If ODMG DMaps are not used, this table is not needed.
OJB_DMAP_ENTRIES	The table containing the DMap entries. The Keys and Values of the map can be arbitrary persistent objects. If ODMG DMaps are not used, this table is not needed.

OJB uses Torque to create all required tables and data. Thus there is no SQL DDL file, but an XML file describing the tables in format readable by Torque. The Torque DDL information for the internal tables resides in the file `src/schema/ojbcore-schema.xml`.

The o/r mappings for these tables are contained in the file `repository_internal.xml`.

If you want to have a look at how these files could be used, have a look at the the ojb-blank sample project which is already prepared to use these files.

2.2. Tables for the regression testbed

It is recommended to run the OJB JUnit regression tests against your target database. Thus you will have to provide several more tables, filled with the proper testdata.

The DDL information for these tables resides in the file `src/schema/ojbtest-schema.xml`.

The testdata is defined in the file `src/schema/ojbtest-data.xml`.

The o/r mappings for these tables are contained in the file `repository_junit.xml`.

2.3. Tables for the tutorial applications

If you intend to run the OJB tutorial applications against your target database you will have to provide one extra table.

The DDL information for this table also resides in the file `src/schema/ojbtest-schema.xml`.

The testdata is also defined in the file `src/schema/ojbtest-data.xml`.

The o/r mappings for this table is contained in the file `repository_user.xml`.

3. The setup process

OJB provides a setup routine to generate the target database and to fill it with the required testdata. This routine is based on Torque scripts and is driven from the build.xml file. This section describes how to use it.

Copyright © All rights reserved.

3.1. Selecting a platform profile

OJB ships with support for several popular database platforms. The target platform is selected by the switch `profile` in the file build.properties. You can choose one out of the predefined profiles:

```
# With the 'profile' property you can choose the RDBMS platform OJB is using
# implemented profiles:
#
profile=hsqldb
# use the mssqldb-JSQLConnect profile for Microsoft SQL Server and
# you will automatically JSQLConnect driver, from http://www.j-netdirect.com/
# MBAIRD: This is my driver of preference for MS SQL Server, I find the OEM'd
# MS driver to have some problems.
#profile=mssqldb-JSQLConnect
#profile=mssqldb-Opta2000
#profile=mssqldb-ms
#profile=mysql
#profile=db2
#profile=oracle
#profile=oracle9i
#profile=oracle9i-Seropto
#profile=msaccess
#profile=postgresql
#profile=informix
#profile=sybase
#profile=sapdb
#profile=maxdb
```

The profile switch activated in `build.properties` is used to select a profile file from the `profile` directory.
If you set `profile=db2`, then the file `profile/db2.profile` is selected.
This file is used by the Torque scripts to set platform specific properties and to perform platform specific SQL operations.

3.2. editing the profile to point to your target db

The platform specific file `profile/xxx.profile` contains lots of information used by Torque. You can ignore most of it. The only important part in this file is the section where the url to the target db is assembled, here is an snip of the DB2 profile:

```
# -----------------------------------------------------------------
#
#  D A T A B A S E  S E T T I N G S
#
# -----------------------------------------------------------------
# JDBC connection settings. This is used by the JDBCToXML task
# that will create an XML database schema from JDBC metadata.
# These settings are also used by the SQL Ant task to initialize
# your Turbine system with the generated SQL.
# -----------------------------------------------------------------

dbmsName = Db2
jdbcLevel = 1.0
urlProtocol = jdbc
urlSubprotocol = db2
urlDbalias = OJB

createDatabaseUrl = ${urlProtocol}:${urlSubprotocol}:${urlDbalias}
buildDatabaseUrl = ${urlProtocol}:${urlSubprotocol}:${urlDbalias}
databaseUrl = ${urlProtocol}:${urlSubprotocol}:${urlDbalias}
databaseDriver = COM.ibm.db2.jdbc.app.DB2Driver
databaseUser = admin
databasePassword = db2
databaseHost = 127.0.0.1
```

These settings result in a database URL `jdbc:db2:OJB`. If your production database is registered with the name `MY_PRODUCTION_DB` you have to edit the entry `urlDBalias` to:

Copyright © All rights reserved.

`urlDbalias = MY_PRODUCTION_DB`.

In this section you can also set application user name and password. You can also enter a different jdbc driver class, to activate a different driver.

Before progressing, please check that the jdbc driver class, named in the `databaseDriver` entry is located on the classpath! You can either edit the global environment variable `CLASSPATH` or place the jdbc driver jar file into the `jakarta-ojb-xxx/lib` directory.

3.3. Executing the build script

Now everything should be prepared to launch the setup routine. This routine can be invoked by calling **`ant prepare-testdb`** .

If you are prompted with a `BUILD SUCCESSFUL` message after some time, everything is OK.

If you are prompted with a `BUILD FAILED` message after some time, something went wrong. This may have several reasons:

- You entered some incorrect settings. Please check the log messages to see what went wrong.
- Torque does not work properly against your target database. Torque is very flexible and should be able to work against a wide range of databases. But the code templates for each database may not be accurate. Please check the ojb-user mailinglist archive if there are any failure reports for your specific database. Please also check if some contributed a fix already. If you don't find anything please post your problem to the ojb user-list.

As a last resort you can try the following: Switch back to the default hsqldb profile and execute `ant prepare-testdb` This will setup the default hsqldb database. And it will also generate SQL scripts that you may use to generate your database manually.

The SQL scripts are generated to `jakarta-ojb-xxx/target/src/sql`. You can touch these scripts to match your database specifics and execute them manually against your platform.

3.4. Verifying the installation

Now everything is setup to run the junit regression tests against your target database.

Execute

```
ant junit
```

to see if everything works as expected. more information about the OJB Test Suite here. If you did not manage to set up the target database with the `ant prepare-testdb` you can use
`ant junit-no-compile-no-prepare` to run the testsuite without generation of the test database.

Copyright © All rights reserved.