OJB - References and Testimonials

by Thomas Mahler

1. References and Testimonials

1.1. projects using OJB

Jakarta JetSpeed
Jetspeed is an Open Source implementation of an Enterprise Information Portal, using Java and XML.
OJB will be the default persistence model within Jetspeed 2.

The Tammi project
Tammi is a JMX-based Java application development framework and run-time environment providing a service architecture for J2EE server side Internet applications that are accessible from any device that supports HTTP including mobile (wireless) handsets.
Future plans include integration of Apache OJB based persistence services to the framework.

The Object Console project
The Object Console is an open web based application meant for the administration of objects via the web. Any object that is persistable by the ObJectRelationalBridge (OJB) framework can be managed through this tool. In addition, this tool provides administration functionality for the ObJectRelationalBridge (OJB) framework itself.
Object Console uses Struts and OJB. It ships with full sourcecode and is thus a great source for learning Struts + OJB techniques.

The IntAct project
The IntAct project establishes a knowledgebase for protein-protein interaction data. It's hosted at EBI - European Bioinformatics Institute, Cambridge.
IntAct uses OJB as its persistence layer.

Network for Earthquake Engineering Simulation
The NEES program will provide an unprecedented infrastructure for research and education, consisting of networked and geographically distributed resources for experimentation, computation, model-based simulation, data management, and communication.
OJB is used as the O/R mapping layer.

The OJB.NET project
OJB.NET is an object-to-relational persistence tool for the .NET platform. It enables applications to transparently store and retrieve .NET objects using relational databases.
OJB.NET is a port ojb Apache OJB to the .NET platform

The OpenEMed project
OpenEMed is a set of distributed healthcare information service components built around the OMG distributed object specifications and the HL7 (and other) data standards and is written in Java for platform portability.
OpenEMed uses ODMG as its persistence API. OJB is used as ODMG compliant O/R tool.

1.2. user testimonials

"We're using OJB in two production applications at the Northwest Alliance for Computational Science and Engineering

(NACSE). One is a data mining toolset, and the other is a massive National Science Foundation project that involves huge amounts of data, and about 20 or 25 universities and research groups like mine.
In fact, I've begun making OJB sort of a de-facto standard for NACSE java/database development. I've thrown out EJB's for the most part and I've tried JDO from Castor, but I'm sticking with OJB. Maybe we'll reconsider JDO when the OJB implementation is more complete."

"We are planning a November 2003 production deployment with OJB and WE LOVE IT!! We have been in development on a very data-centric application in the power industry for about 5 months now and OJB has undoubtedly saved us countless hours of development time. We have received benefits in the following areas:
-> Easily adapts to any data model that we've thrown at it. No problems mapping tables with compound keys, tables mapping polymorphic relationships, identity columns, etc.
-> Seemlesly switches between target DB platforms. We develop and unit test on our local workstations with HSQLDB and PostgreSQL, and deploy to DB2 using the Type 4 JDBC driver from IBM. Works great!
-> Makes querying a breeze with the PersistenceBroker API
Overall we have found OJB to be very stable (and we've really tested it out quite a bit). The only issues we've got outstanding at the moment is support for connections to multiple databases, but I've noticed in CVS that the OJB guys are already fixing this for OJB 0.9.9."

"We've been using it in "production" for a long time now, from about version 0.9.4, I believe. It has been very robust. We don't use all of its features. We've only see to failures of our persistent store in about 9 months, and I'm not sure they were due to OJB."

"So yes, we have made a quite useful mediumsized production website based on OJB (with JBoss, Jakarta Jetspeed, Jakarta Turbine and Jakarta Jelly, three Tomcats, OpenSymhony OSCache and for the database MSSQL server, all running on Win2000.) It is attracting between 600 and 9000 (peak) users a day, and runs smoothly for extended periods of time. And no, I can not actually show you the wonders of the editorial interface of the content management system, because it is hidden behind a firewall.
I feel OJB is quite useful in production, but you certainly have to know what you are doing and what you are trying to achieve with it. And there have been some tricky aspects, but these could be solved by simple workarounds and small hacks.
The main thing about OJB is that AFAIK it has an overall clean design, and it far beats making your own database abstraction layer and object/relational mapper. We certainly do not use all of it, only the Persistence Broker parts, so there was less to learn. We love the virtual proxy and collection proxy concepts, the criteria objects for building queries, and the nice little hidden features that you find when you start to learn the system."

"My Company is building medium to large scale, mission critical applications (100 - 5.000 concurrent users) for our customers. Our largest customer is KarstadtQuelle, Europes largest retail company. The next big system that will go in production (in June) is the new logistics system for the stationary logistics of Karstadt.
Of course we are using OJB in those Systems! We have several OJB based systems now in production for over a year. We never had any OJB related problems in production.
Most problems we faced during development were related to the learning curve developers had to face who were new to O/R mapping."

"I've also worked with OJB on high-load situations in J2EE environments. We're using JRun and/or Orion with OJB in a clustered/distributed environment. This is a National Science Foundation project called the Network for Earthquake Engineering Simulation (NEES).
The only major problem that we ran into was the cache. JCS just isn't good, and hasn't seemed to get much better over the last year. We ended up plugging in Tangosol's Coherence Clustered Cache into the system. We can also do write-behinds, and buffered data caching that is queued for transaction. That's important to us because we're dealing with very expensive scientific data that _can't_ get lost if a db goes down. Some of these Tsunami experiments can get pretty expensive.
Otherwise, we use mostly the PersistenceBroker, and a little of the ODMG. Performance seems better on PB, but less functional. It's not really that much of a problem anyway, because we can cheaply and quickly add app-servers to the cluster."