Insight into the jBOSS Project

# SCALING THE HEIGHTS

DANIEL SCHULZE

**JBoss is a free project for an Enterprise Java beans application server. The extensive and lively developer scene and consequent orientation towards modern Java technologies such as JMX could help jBoss scale the heights of its class.**

The proposal sounds fascinating: The basis for applications no longer consists of the operating system, but a more abstract, clearly-drawn, well-documented and component-oriented platform, an Internet Operating System - or J2EE, as Sun Microsystems unpretentiously calls it. And now this new infrastructure, just like Linux, is to be made available free to everyone who needs it. Mesmerised by this vision and for the fun of programming, Marc Fleury, (then employed at SUN as a Java evangelist), founded the EJB Open Source Server (EJBOSS) project. Very quickly, high-calibre coders such as Richard Monson Haefel, who are helping with their contribution to the design and code, got the young project off the ground. The project was given an additional impetus when Richard Öeberg came across it and added the latest available Java technologies, which lent jBoss in its current, second generation, standard-setting characteristics. JMX - Guarantee for simple interfaces and expandabilty

The technical kernel consists of a JMX-(Java Management eXtension) server. JMX, a management API introduced by SUN as an official J2EE component, will soon become the standard for the management of distributed Java applications, so that commercial suppliers such as BEA can copy with Weblogic and integrate JMX. The advantage of this technology is a modular structure with clear interfaces, making it simple to integrate additional components and expansions. What originally began as an EJB container is now well on the way to becoming a proper J2EE server. JBoss is 100 per cent developed in Java, has an extremely narrow memory footprint, which makes it ideal for integration into other systems, and is licenced under the LGPL.

At present there are JMX-wrappers for two Web server/servlet containers (Tomcat3.2, Jetty3). There is also an integrated JMS (Java Messaging Service) implementation and an integration module for Castor, a free JDO implementation. JBoss speaks SOAP and there is even an integration module available for the well-known commercial O-R-mapping utility Cocobase.

The secret of this rapid success lies in the way in which Fleury manages the community 24 hours a day, seven days a week and makes sure that no-one loses focus. There is a very friendly, helpful atmosphere in the mailing lists. Anyone can contribute code and there is also constant encouragement to do so. So a frequent reply to queries as to when features will become available is.

"When you start your IDE and begin to make them happen!"

Many developers integrate their own projects into jBoss, examples worth mentioning include SpyderMQ (JMS) and Minerva, the implementation of a database connection pool.

The main focus of the whole project is ease of use; the motto goes: Hide as much complexity from the user as possible.

## Simple usability main objective of the project

This condition has brought with it, among other things, features like dynamic proxies and hot deploy/hot redeploy. Anyone who wants to deploy an EJB archive under jBoss, does not need to know how to create skeletons and stubs and which ones must then go into which class path. Simply copy the desired archive into the deploy folder and voila - beans deployed! If one integrates one of the two available servlet machines, one can also start Web archives (*.war*) or entire enterprise archives (*.ear*) simply by means of Drag and Drop. At present, though, *.web* and *.ear* integration is not yet 100 per cent J2EE-compatible. There are still problems in the JNDI and security integration.

## Unpack and go – Installation is no problem

Ease of use begins from when the server starts. Download jBoss from the Internet, no matter if it's CVS or binary distribution, simply execute *sh run.sh* (in the case of CVS, *sh build.sh* first) and it runs. An additional feature accompanying the distribution is EJX, a GUI-based XML file editor for simple assembly of EJB archives and the metadata necessary for this (*ejb-jar.xml*). This tool is being completely revised and will reappear in a few weeks with renewed sparkle.

The EJB container itself currently supports the EJB specification Version 1.1. It integrates a transaction monitor, JAWS, a persistence manager, and a JAAS (Java Authentication and Authorisation Service) compatible security layer. In principle any relational database can be used for the persistence layer (BMP/CMP), as long as there is a JDBC driver available for it. The distribution already includes two pre-configured pure Java databanks (Hypersonic, InstantDB). Mapping data for the following additional databases is available for JAWS, so that they can be integrated with ease: Oracle, PostgreSQL, Pointbase, Solid, mySQL, MS SQLServer, DB2/400 and SAPDB.

## Those who keep on reading are smarter

Despite simple installation and what is, in principle, simple usability, some questions remain unanswered. But it is also clear that programmers are reluctant to write documentation. Though efforts are being made to increase the supply of documentation and especially to keep the existing documentation up to date. But the rapid progress of the project means this is not always so simple. If the documentation provided on the Web site, the *Getting Started* guides and Howtos do not come up with the answer for a certain problem, the best source for information is the jBoss mailing list. Here can be found what seems to be an answer to every question, in most cases actually from the author of the code. It's certainly easier to look it up in the smartly designed ring binder provided, out new - this is open source!

Frequently asked questions are: Can I implement jBoss in a production environment? Do any benchmarks exist? How does the server scale? When will such and such a feature become available? Now, jBoss can be (and is) used in production environments for example: http://www.liquidwit.com. It may be a little early yet for company-critical environments, but in a month or two this hurdle may be overcome on the basis of the present momentum of the community. A minus point for use in larger distributed systems is the transaction manager, which currently does not support any distributed transactions, and 2-Phase Commit (committing data to more than one database simultaneously). The design of the container itself scales superbly. No matter how many clients are connected at the same time, all queries run via a central, very fast *container Invoker* authority, from where they are passed on to the corresponding beans.
The container itself is statusless, all necessary data is extracted from the incoming client invocation.

## The future – Clustering, Jini, EJB 2.0

Developers are now working hard on plans for clustering. In the good old jBoss tradition, you can look forward eagerly to some tasty technical morsels – one thing which can be disclosed is that JINI will play a leading role in this. And of course EJB 2.0 is also in the pipeline, *message driven beans* are said to be already running under laboratory conditions. For CMP2.0, though, there are no volunteers as yet.

Also in areas such as configuration tools, more fault-tolerant application deployment, and also the aforementioned clusters and EJB2.0 plans, there is still a lot to be done. But in general this is the right direction, the momentum on the part of the community is great enough to sustain the driving force for development on the part of the community, and now everyone knows how successful open source projects can be. ■


## The author

*Daniel Schulze is actually studying computer science in Leipzig, but at present he is living and working in San Francisco at Olliance, an open source consultancy. He became a jBoss developer as the result of work experience at Teksel.*