Proposal for Paper:
Implications of Java Data Objects for Database Application Architectures

Craig E. Ward
CMSI 698 SS:Advanced Topics in Database Systems
Loyola Marymount University, Spring 2004

1 Introduction

The set of technologies comprising the Java programming environment provides multiple approaches to programmatic access to database systems. All of these approaches can not be equally appropriate for all database systems. Choosing an approach depends on the requirements of the particular database system in question.

Java technologies allow for relatively simple, direct access to the database using the *Java Database Connectivity* (JDBC) API either by direct use of the API or the direct embedding of SQL code using the SQLJ wrapper. The *Enterprise Java Beans* (EJB) of *Java 2 Enterprise Edition* (J2EE) and the more recent *Java Data Objects* (JDO) provide the ability to create *n*-tiered architectures. There are also two kinds of EJBs. *Entity* beans represent objects or records that need to persist in a database and *Session* beans that only exist during the life cycle of one transaction or session of an application.

These technologies are not mutually exclusive. It is possible to mix parts of one with another, *e.g.* EJBs may use JDBC for accessing database records.

The purpose of this paper is to review how others have chosen to balance the needs of a database system with an architecture that utilizes the most appropriate mix of Java technologies. A particular emphasis for the paper is how the newer JDO changes the balance between direct JDBC calls and a tiered EJB approach.

2 Motivation

The creation of the World Wide Web has added the problem of incorporating a database system into multi-tiered applications. Java technologies provide an excellent means for creating web applications.

System designers in the Java realm can choose between Java *servlets* and *Java Server Pages* (JSP) for the front end of an application. Servlets and JSPs can use the

JDBC API directly or they can access a back end of EJBs. The designer must then choose between *Container Managed Persistence* (CMP) or *Bean Managed Persistence* (BMP) for any *Entity* EJBs. Back end containers that support the JDO API are now becoming available and provide another option.

The growing number of combinations of these technologies presents an interesting problem. Which technology or combination of technologies is best for any particular application?

3 Background Work

JDO technology is a relatively recent addition to the "Java suite." Most of the material currently available focuses on the balances between using JDBC directly or EJBs in a container. The issue of CMP vs. BMP for any EJBs has also been researched. How the market will respond to JDO technology is still an open question.

(Eisenberg and Melton, 1998) describe the SQLJ standard and illustrate how it can be used to embed SQL code directly into Java code. SQLJ is a wrapper around JDBC and does not seem to be in much use today. It is included for completeness.

(Salo and Hill, 2000) provides a comprehensive comparison of using EJBs in various combinations of servlets and JSP pages. (Tost, 2000) discusses the benefits of using Java Beans (an embeddable object not related to the later EJB standard) and EJBs together.

The performance and scalability of J2EE applications is addressed in (Cecchet, Marguerite, and Zwaenepoel, 2002) and (Gorton and Liu, 2003). The former implements an e-commerce application using different architectures comprised of Entity beans and Session beans with BMP and CMP. The latter focuses on a similar issue but reports on experiments using several different container systems.

One of the papers does explicitly address JDO. In (Baldwin, 2003), the issue hinges on how well these technologies (plus some non-Java specific ones) handle the extremely large data sets at the National Climate Data Center.

The other references serve as background for the JDO technology.

The goal of this research paper will be to synthesize the established results regarding direct access using JDBC in servlets or JSPs versus using the tiered architectures possible with EJBs with the new developments possible with JDO.

4 References

Almaer, Dion. (2002). Using Java Data Objects. *ONJava.com* 2/6/2002. URL: http://www.onjava.com/lpt/a/1372

Java Data Objects (JDO) is a recent addition to the suite of APIs available for accessing databases from a Java-based environment . This article provides some simple examples of how the JDO technology is used to build Java objects from relational databases.

Baldwin, Richard T. (2003). Views, Objects, and Persistence for Accessing a High Volume Global Data Set. *Digest of Papers IEEE Symposium on Mass Storage Systems (MSS'03)* p 77-81

This paper describes lessons learned by the National Climate Data Center (NCDC) as it decides how to deploy extremely large data sets for use by scientists and researchers world-wide. Prototype systems were developed using direct file access from Java programs, a key/value database bundled with Unix systems, and Java Data Objects (JDO) and Enterprise JavaBeans (EJB) interfaces to RDBMS and ODBMS.

Brown, Jeff. (2002). An Introduction To Java Data Objects. *Object Computing, Inc. - Java News Brief* June 2002. URL: http://www.ociweb.com/jnb/jnbJun2002.html

This web article provides a general overview of the Java Data Objects (JDO) API and how it relates to the prior Java Database Connectivity (JDBC) API. The XML descriptions of the RDBMS-to-Java object mappings are also discussed.

Cecchet, Emmanuel ; Marguerite, Julie and Zwaenepoel, Willy. (2002). Performance and Scalability of EJB Applications. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, OOPSLA, 2002, p 246-261

The paper describes a study of the performance of an e-commerce application utilizing different combinations of Enterprise JavaBeans. The conclusion of the study is that stateless session beans with bean managed persistence out perform alternate combinations of entity beans with container managed persistence.

Eisenberg, Andrew and Melton, Jim. (1998). SQLJ Part O, now known as SQL/OLB (Object-Language Bindings). *SIGMOD Record*, Vol. 27, No. 4, December 1998. p 94-100

SQLJ/OLB is a standard for embedding SQL code directly into Java source code. This paper describes the standard and illustrates how it merges with the Java Database Connectivity (JDBC) API.

Gorton, Ian and Liu, Anna. (2003). Evaluating the Performance of EJB Components. *IEEE Internet Computing*, v 7, n 3, May/June, 2003, p 18-23

A report comparing the performance of two common Java 2 Enterprise Edition's (J2EE) Enterprise JavaBean (EJB) application architectures. J2EE allows for management of object persistence either by the server container or the EJBs themselves.

Jordan, David and Russell, Craig. (2003). JDO or CMP? *ONJava.com* 05/21/2003. URL: http://www.onjava.com/lpt/a/3763

This web article provides a bullet comparison of the capabilities of object persistence using Java Data Objects (JDO) and Container Managed Persistence (CMP) of the Java 2 Enterprise Edition standard. It is excerpted from the book *Java Data Objects* published by O'Reilly & Company by the same authors.

Salo, T. and Hill, J. (2000). Building Enterprise Web Applications with Java. *JOOP - Journal of Object-Oriented Programming*, v 13, n 2, May, 2000, p 28-29+47

Five architectures for web applications utilizing different combinations of Java-based technologies are examined. Systems using components comprised of servlets, Java Server Pages (JSP), Java Beans and Enterprise JavaBeans, and combinations are compared.

Tost, A. and Johnson, V.M. (2000). Using JavaBeans Components as Accessors to Enterprise JavaBeans Components. *IBM Systems Journal*, v 39, n 2, 2000, p 293-300

The paper describes how using JavaBean components in a Enterprise JavaBean allows for a clean , three-tier architecture separating client-side and server-side implementations.