Building Clustered Enterprise Applications
with JBoss Application Server on the Dell PowerEdge 1855 Blade Server

JBoss Application Server (JBoss AS) is a standards-based Java platform for scalable enterprise applications. The Dell™ PowerEdge™ 1855 blade server can provide a cost-effective system for hosting scaled-out applications on the JBoss AS platform. To demonstrate the ease of migrating applications to this platform, a team of engineers from Dell and JBoss ported a Web application to JBoss AS on a PowerEdge 1855.

The certification of JBoss AS on the Dell PowerEdge 1855 exemplifies Dell support for open source software. The combination of JBoss AS and MySQL database from MySQL AB, both of which are included with Novell® SUSE LINUX Enterprise Server 9, can provide a cost-effective, Dell-supported platform for enterprise applications. The Dell and JBoss relationship further extends the Professional Open Source model, which combines the cost-savings benefits of open source with the development methodologies, support, and accountability expected from leading hardware and software vendors.

To demonstrate the ease of building applications on the JBoss AS platform, in July 2005 a team of engineers from Dell and JBoss ported a Web application—which uses JavaServer Pages (JSP) to implement the front end of an online DVD store—to JBoss AS running on two PowerEdge 1855.

---

1 JBoss Inc. certified JBoss AS software to run on Dell PowerEdge servers, including the Dell PowerEdge 1855.
blade servers in less than a week. In the process, the team added features to the Web application, including secure sign-on, shopping-cart persistence, clustering, failover, and internationalization.\(^2\) The MySQL database back end required no modifications to be ported to the JBoss AS platform. This article provides an overview of the features provided by JBoss AS—with a focus on the features added to the DVD store application when ported to JBoss AS—as well as details of the JBoss AS DVD store implementation.

**Understanding JBoss AS**

JBoss AS is a standards-based, J2EE 1.4 certified application server providing the foundation upon which enterprises, software vendors, integrators, and solution providers can build applications and Web services. Application platforms like JBoss AS are typically referred to as middleware and are designed to simplify the development of business applications by providing a layer of software infrastructure that abstracts the low-level operating systems, communication protocols, and hardware details.

Companies rely on JBoss for deploying scalable and secure Web applications that are composed of complex business logic and serve thousands of concurrent requests. Because JBoss AS is founded on a service-oriented microkernel architecture, it is designed to provide services in a plug-and-play fashion, including all J2EE 1.4 services such as Enterprise JavaBeans (EJB), JSP, servlets, and Web services, along with extended enterprise services for clustering, caching, failover, persistence, and distributed deployment. This service orientation allows small, medium, and large enterprises to standardize on JBoss AS because it is designed to scale from single, low-end server configurations to clustered environments that contain hundreds of high-end servers.

JBoss AS is designed to provide a complete platform for building a Web application’s user interface, business logic, and data access logic. This article and the DVD store demonstration application are focused on the following features:

- Ease of development
- Data persistence
- Application security
- Clustering and failover
- Internationalization

**Ease of development**

JBoss AS enables developers to leverage enterprise features without undue complexity—as reflected in the design of features such as clustering, which requires no application changes or design-time code modifications. Moreover, JBoss AS 4.0 continues to simplify development by including support for two important technologies: JavaServer Faces (JSF) and ESB 3.0.

JSF provides a standards-based framework for handling a Web application’s presentation layer, and EJB 3.0 provides a simple programming model for the business-logic layer and data-access layer. Both JSF and EJB 3.0 are critical components of the Java Platform, Enterprise Edition 5 (Java EE 5) standard, which is designed to directly address the needs of developing enterprise-class applications (including user interface, business logic, data access, and persistence) in a dramatically simplified manner. By enabling developers to isolate their business logic from the user interface and data-access logic, JBoss AS helps simplify development, improve application maintenance, and enable IT organizations to deliver high value in a relatively short period of time.

**Data persistence**

Because JBoss AS supports the EJB 3.0 specification, it inherently provides a highly flexible and productive mechanism for storing Java objects and EJB components in relational database tables. JBoss AS not only supplies high-performance access to data, it also enables the application to support any relational database for its back-end data. This support for transparent data persistence means that although the DVD store application uses MySQL as its database, other relational databases such as an Oracle* or Microsoft* SQL Server* database can be used without any changes to the application.

**Application security**

Because application servers are platforms for enterprise applications, they are expected to provide a wide range of mission-critical services and security features. JBoss AS supports the Java Authentication and Authorization Service (JAAS) standard application programming interface (API), which is designed to provide a seamless security architecture across J2EE applications. Using simple
declarative security statements, developers can restrict J2EE application access to authenticated users. Moreover, security can be externally configurable so there are no explicit security checks within an application.

**Clustering and failover**

JBoss AS achieves scalability and fault tolerance through its clustering technology, which makes it suitable for deployment across large numbers of servers. The clustering technology is designed to be transparent to the application, so cluster nodes automatically discover one another on boot-up—with no additional configuration. Any application can be made to run on a JBoss AS cluster, and clustering can be activated by changing a JBoss AS configuration setting. Doing so is enough to enable load balancing, state replication, and failover for an application’s Java beans.

**Internationalization**

Organizations creating globally deployed applications require a platform that is designed for internationalization, such as JBoss AS. By using JSF, developers can create applications that target a global audience. JSF isolates the user interface’s text to help simplify the localization of the application to a specific language.

**Implementing the DVD store application**

The Dell and JBoss team implemented the DVD store application using the standard Java EE 5 technologies available in JBoss AS 4.0.3. This section describes the implementation and configuration of the application running on JBoss AS.

**A multi-tiered architecture**

The DVD store application uses a standard multi-tiered architecture consisting of a database tier, an EJB 3.0 tier, and a Web tier. Although the Web and EJB 3.0 tiers are logically separate layers, they are colocated to maximize performance. This colocation is the recommended deployment strategy for applications built on JBoss AS.

Figure 1 shows the overall system architecture used in the demonstration implementation. JBoss AS 4.0.3 was deployed on two Dell PowerEdge1855 blade servers. The DVD store application was deployed on each JBoss AS instance, with JBoss Cache providing the clustering link between them. Additional PowerEdge 1855 nodes could have been added to the cluster with no additional JBoss AS configuration required. Both JBoss AS instances communicated with a MySQL database running on a Dell PowerEdge 2800 server.

A front-end load balancer is required to provide HTTP failover between nodes. A hardware load balancer is generally preferred; however, JBoss AS also supports the use of the Apache HTTP server

[Figure 1. System architecture for the DVD store Web application]

with the mod_jk connector to provide software load balancing.

**Database tier.** MySQL provided the database tier in the implementation described in this article (see the “The DVD store MySQL database” sidebar in this article for database details). The connection between JBoss AS and MySQL is managed in JBoss AS using a standard J2EE Connector Architecture (JCA) connection pool. The connection pool allows for fast database access and removes the connection management code from the application.

**EJB 3.0 tier.** The EJB 3.0 tier provides the data management and core business logic features of the application. It does so through EJB 3.0 entity beans, which provide the mapping between Java objects and data in the database, and stateless session beans, which provide the business logic that the application will use.

Compared to earlier EJB versions, EJB 3.0 entity beans provide a greatly simplified method of mapping relational data to the database. The entity beans are implemented as lightweight plain old Java objects (POJOs) with minimal use of annotations to provide persistence details. Unlike EJBs under earlier J2EE versions, no special interfaces, magic methods, or XML configurations are needed.

The Dell and JBoss team created one EJB 3.0 entity bean for each table in the database.

The application provides table and column mapping details but is otherwise relieved of the burden of managing the object-relational mapping.

EJB 3.0 persistence eliminates the need for an application to manage SQL queries. Figure 2 lists the code required to load the orders for a customer from the database.

Although native SQL queries can be used to optimize performance, the application as written is portable to any database that

---

1 For code of the EJB 3.0 entity bean that corresponds to the PRODUCTS table, see the supplemental online section of this article at www.dell.com/powersolutions.
public List<Order> getOrders(Customer customer) {
    return em.createQuery("from Order o where o.customer = :customer")
        .setParameter("customer", customer)
        .getResultList();
}

Figure 2. Java code for loading orders for a customer from the database

| public List<Order> getOrders(Customer customer) { |
|    return |
|    em.createQuery("from Order o where o.customer = :customer") |
|    .setParameter("customer", customer) |
|    .getResultList(); |
| } |

JBoss AS supports. No application code must be rewritten to switch to a different database.

The application provides a single stateless session bean that provides the core business logic of the application. The DvdStoreBean provides the logic for data lookup, such as the getOrders() method, inventory management, and purchase processing. Figure 3 shows the external interface to the DVD store session bean.

Like EJB 3.0 entity beans, session beans are similarly lightweight and require no heavy interfaces or external XML configuration files to take advantage of enterprise aspects such as security and transactions. For example, the purchase method of the DvdStoreBean processes the order and creates all of the objects that must be written to the database. It also handles the application-specific requirements when a transaction cannot be completed because of insufficient quantities of a DVD.

**Web tier.** Although the business logic is provided by the EJB 3.0 tier, the Web tier assembles the logic and presents a coherent application to the user. The DVD store uses a combination of JSF and JSP to manage the user experience.

Using a JSF component-based architecture has many advantages. The model-view-controller (MVC) architecture allows for a clean separation of the Web application responsibilities. The JSF backing beans provide a rich model for the interface, and they map user requests onto the EJB 3.0 business logic. The JSP pages provide a simple view with no embedded business logic and minimal HTML and HTTP details exposed. JSF also provides powerful internationalization features that externalize text for easy localization of the application in a specific language.

**Externalized security**

Security is externalized in the DVD store application—it has no explicit security checks. The Dell and JBoss team defined a JAAS domain that could understand the usernames and passwords in the relational database. Using simple declarative security statements, developers can restrict application access to authenticated users. JAAS enables developers to replace the security domain with one that consults a Lightweight Directory Access Protocol (LDAP) server or makes use of another authentication technology without changing any application code.

**Clustering support**

The Dell and JBoss team added clustering support to both the Web tier and the EJB 3.0 tier of the DVD store application. The application uses HTTP session replication and clustered single sign-on in the Web tier. If any JBoss AS node were to fail, a front-end HTTP load balancer would fail the request over to another JBoss AS server. This application server would be in the right state to process the request, and the user would continue using the application with no noticeable interruption of service.

Additionally, a second-level clustered entity cache in the Dell-JBoss implementation maintained primarily read-only data such as product categories. Once this data was loaded into the cache, the EJB 3.0 tier no longer needed to consult the database to load that data. The DVD store application did not need to be changed in any way to support the cache. All clustering options in JBoss AS make use of the JBoss Cache/JGroups stack and require multicasting communication between the nodes.

**Porting applications to the JBoss AS platform**

The DVD store application was ported in less than a week, with no changes required to the database layer. Through the use of

```
public interface DvdStore |
| { |
|     public Customer getCustomer(String user); |
|     public List<Order> getOrders(Customer customer); |
|     public List<Product> getRecentHistory( |
|         Customer customer, int howmany); |
|     public List<Product> getProducts(); |
|     public List<Product> searchProducts(String title, |
|         String actor, Category category, int howmany); |
|     public Order purchase(Customer customer, |
|         List<OrderLine> lines) |
|         throws InsufficientQuantityException; |
|     public List<Category> getCategories(); |
| } |
```

Figure 3. Java code for the external interface to the DVD store session bean

---

*For code of the purchase method of the DvdStoreBean session bean, see the supplemental online section of this article at www.dell.com/powersolutions.*

---

58 DELL POWER SOLUTIONS

Reprinted from Dell Power Solutions, February 2006. Copyright © 2006 Dell Inc. All rights reserved.

February 2006
The DVD STORE MYSQL DATABASE

The back-end database employed in the Dell-JBoss implementation was a large MySQL database (100 GB total size), representing an online DVD store with 1 million DVD titles, 200 million customers, and 120 million orders. Advanced database features such as transactions and referential integrity constraints were employed.

The MySQL DVD store database comprised seven main tables and one small table (see Figure A).

The CUSTOMERS table was prepopulated with 200 million customers: 100 million U.S. customers and 100 million customers from the rest of the world. The ORDERS table was prepopulated with 10 million orders per month for a full year. The ORDERLINES table was prepopulated with an average of five items per order. The PRODUCTS table contained 1 million DVD titles, each with a principal actor listed for search purposes. For realism, titles and actor names were generated by taking combinations of actual movie titles and actor names. Additionally, the CATEGORIES table contained the 16 DVD categories.

The schema is fully documented in the database build script, which can be found in the supplemental online section of this article at www.dell.com/powersolutions. The complete MySQL benchmarks and the Dell Technology Showcase. He has a B.S. in Chemistry from Yale University and a Ph.D. in Chemistry from the University of California, San Diego.

Norman Richards is a developer at JBoss, Inc. He is the co-author of JBoss: A Developer’s Notebook and XDoclet in Action.

Shaun Connolly is the vice president of product management at JBoss, Inc. Shaun is responsible for the products that make up JEMS. Prior to joining JBoss, Shaun served in vice president– and director-level product management and product development positions at Princeton Softech, HP Middleware, Bluestone Software, and Primavera Systems. Shaun has a B.S. in Electrical Engineering from Drexel University and has been a panelist, speaker, and contributor of articles on a wide range of IT topics.