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

Enterprise Product Group (EPG)

Dell White Paper

By Todd Muirhead and Dave Jaffe, Dell Inc. and Norman Richards and Shaun Connolly, JBoss Inc.

Dell | Enterprise Systems

August 2005
Contents

Executive Summary................................................................. 3
Introduction........................................................................... 4
JBoss Application Server Overview........................................ 5
   Ease of Development......................................................... 5
   Data Persistence ............................................................... 6
   Application Security.......................................................... 6
   Clustering and Failover ...................................................... 6
   Internationalization............................................................ 6
The JBoss DVD Store Application.......................................... 7
   Architecture........................................................................ 7
   Database Tier ..................................................................... 8
   EJB3 Tier ........................................................................... 8
   Web Tier .............................................................................. 9
   Security ............................................................................... 9
   Clustering .......................................................................... 9
The MySQL Database........................................................... 11
   The Database Schema......................................................... 11
Conclusions........................................................................... 13
Acknowledgements............................................................... 14
Appendix A. Products Table EJB3 Entity Bean......................... 15
Appendix B. Purchase Method of DvdStoreBean Session Bean .... 17
Appendix C. mysqlDs2 Build Script ....................................... 18

Figure 1: JBoss Application Architecture .................................. 7

Table 1: DVD Store Database Schema...................................... 11
Executive Summary

The JBoss® Application Server (JBoss AS), the most popular Java 2 Enterprise Edition (J2EE) application server available on the market\(^1\), provides a standards-based platform for scalable Enterprise applications at an unbeatable zero cost software license, enabling scale out of applications without prohibitive per CPU licensing costs. The Dell PowerEdge 1855 blade server, now certified for JBoss, provides a cost-effective platform for such scaled-out applications. The PowerEdge 1855 blade server delivers on the promise of high density computing by offering complete server class features combined with a density and price advantage over traditional rack servers. Together, Dell and JBoss offer open source customers reliable, stable server solutions via leadership in standards-based computing.

To demonstrate the ease of migrating applications to the PowerEdge 1855/JBoss platform, a web application using Java Server Pages representing an online DVD Store with a MySQL database backend was ported to JBoss. This paper provides details of how the application was implemented, showing how JBoss facilitates the addition of such Enterprise features as secure sign on, persistence, clustering and failover and internationalization.
Introduction

According to a recent survey, the JBoss® Application Server (JBoss AS) has emerged as the most popular Java 2 Enterprise Edition (J2EE) application server available on the market.1 The cornerstone of the JBoss Enterprise Middleware Systems (JEMS), JBoss AS is a proven platform for building and deploying new revenue-generating applications. JBoss AS provides a standards-based platform for scalable enterprise applications at an unbeatable zero cost software license, enabling radical scaling out of applications without prohibitive per CPU licensing costs. The market share leader in the United States and Asia/Pacific/Japan for x86 servers running Linux, Dell provides JBoss customers with affordable, supported open source solutions on award winning PowerEdge servers.

The cost-effective, easy-to-manage Dell PowerEdge 1855 blade server is an excellent platform for JBoss Application Server. The PowerEdge 1855, now a certified platform for JBoss AS, supports up to 10 blade servers in a 7-rack unit (12.25””) chassis. Each blade can be configured with up to two Xeon EM64T processors and thus supports the latest 64 bit operating systems.

The certification of JBoss Application Server on the Dell Power Edge 1855 is another milestone demonstrating Dell’s support for open source software. The combination of JBoss AS and MySQL database from MySQL AB, which come packaged with Novell’s SUSE™ Linux Enterprise Server 9 provides a very cost-effective 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 platform, a web application built with Java Server Pages (JSP) implementing the front end of an online DVD Store was ported to JBoss running on two PowerEdge 1855 blades in less than a week, in the process gaining such features as secure sign-on, shopping cart persistence, clustering and failover and internationalization. The MySQL database backend required no changing. (The original JSP-based application is described at http://conferences.oreillynet.com/cs/mysqluc2005/view/e_sess/6226).

An overview of the features provided by JBoss AS, with a focus on the new features added to the DVD Store application, is provided in Section 3, followed by details of the JBoss DVD Store Implementation in Section 4. MySQL database details are provided in Section 5.
Section 3

JBoss Application Server Overview

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

Companies rely on JBoss AS for deploying scalable and secure Web-based applications comprised of complex business logic that serves thousands of concurrent requests. Since JBoss AS is founded on a service oriented microkernel architecture, it is designed to provide all J2EE 1.4 services – including Enterprise Java Beans (EJB), Java Server Pages(JSP)/ Servlets, and Web Services – along with extended enterprise services – for clustering, caching, fail-over, persistence, and distributed deployment – in a plug and play fashion. This means that small, medium, and large enterprises can feel comfortable standardizing on JBoss AS since it is designed to scale from single low-end server configurations to clustered environments that contain hundreds of high-end servers.

While JBoss AS provides a complete platform for building a Web-based application’s user interface, business logic, and data access logic, this paper and corresponding DVD Store demo application, described in more detail below, are focused on the following highlights:

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

Ease of Development

JBoss AS is the favorite application server among developers for good reason. JBoss AS makes it easy for developers to leverage enterprise features without undue complexity as reflected in the design of features such as Clustering, which requires no changes or design time code modifications. Moreover, the latest release of JBoss AS continues to simplify development by including support for two very important new technologies; Enterprise Java Beans 3.0 (EJB3) and Java Server Faces (JSF).
JSF provides a standards-based framework for handling a Web application’s Presentation Layer, and EJB3 provides a much simpler programming model for the Business Logic Layer and Data Access Layer. Both JSF and EJB3 are critical components of the Java Enterprise Edition 5 platform standard which directly addresses 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 greater value in less time.

**Data Persistence**

Since JBoss AS supports the new EJB3 specification, it inherently provides a highly flexible and productive mechanism for storing Java objects and EJBs in relational database tables. JBoss AS not only provides 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 while the DVD Store application uses MySQL as its database today, other relational databases such as Oracle or SQL Server can easily be used tomorrow without any changes to the application.

**Application Security**

Since application servers are platforms for enterprise applications, they are expected to provide a wide range of mission-critical services and security features. JBoss AS leverages the Java Authentication and Authorization Service (JAAS) standard, which provides a seamless security architecture across J2EE applications. Using simple declarative security statements, access to J2EE applications can be restricted to authenticated users. Moreover, security can be entirely 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. This 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 your Java beans.

**Internationalization**

Organizations creating globally deployed applications with complex business logic that serves thousands of users require a platform like JBoss AS that is designed for internationalization. By utilizing Java Server Faces (JSF), developers can create applications that target a global audience. JSF isolates the user interface’s text in a way that makes localization of the application to a specific language easy.
The JBoss DVD Store Application

The DVD Store application was implemented using the standard Java Enterprise Edition 5 technologies available in JBoss AS 4.0.3. This section describes both the implementation and configuration of the application running on JBoss AS.

The application uses a standard multi-tiered architecture consisting of a database tier, an EJB3 tier and a web tier. Although the web and EJB3 tiers are logically separate layers, they are collocated in order to maximize performance. This is the recommended deployment strategy for applications built on JBoss AS.

Architecture

Figure 1: JBoss Application Architecture shows the overall system architecture used in this test. JBoss 4.0.3 was deployed on two Dell PowerEdge1855 blade servers. The DVD Store application was deployed on each JBoss instance, with JBoss Cache providing the clustering link between them. Additional PE1855 nodes could be added to the cluster with no additional JBoss configuration required. Both JBoss instances communicate to a MySQL server running on a Dell PE2800.
A front-end load balancer is required to provide HTTP failover between nodes. A hardware load balancer is generally preferred. However, JBoss also supports the use of Apache with the mod_jk connector to provide software load balancing.

Database Tier

MySQL provides the database tier (see Section 5 for database details). The connection between JBoss and MySQL is managed in JBoss using a standard JCA (Java Connector Architecture) connection pool. The connection pool allows for faster database access and removes the connection management code from the application.

EJB3 Tier

The EJB3 tier provides the core business logic and data management features of the application. This is done through EJB3 entity beans, which provide the mapping between data in the database and Java objects, and stateless session beans, which provide the business logic that the application will use.

EJB3 entity beans provide a greatly simplified method of mapping relational data to the database. One EJB3 entity bean was created for each table in the database. EJB3 entity beans are implemented as lightweight POJOs (plain old Java objects) with minimal use of annotations to provide persistence details. Unlike older J2EE versions, no special interfaces, magic methods or XML configuration is needed. Appendix A shows the EJB3 entity bean that corresponds to the PRODUCTS table. The application provides table and column name mapping details but is otherwise relieved of the burden of managing the object-relational mapping.

EJB3 persistence eliminates the need for an application to manage SQL queries. The following listing illustrates the code required to load the orders for a customer from the database.

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

Although it is possible to provide native SQL queries to optimize performance, the application as it is written is completely portable to any database that JBoss supports. No application code would need to be rewritten to make the changes.

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. The following listing shows the external interface to the DVD store session bean:

```java
public interface DvdStore {
```
Like EJB3 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. Appendix B shows the implementation of the purchase method. The purchase method processes the order and creates all of the objects that need to be written to the database. It also handles the application-specific requirements of not completing the transaction when there is insufficient quantity of a DVD available.

Web Tier

Although the business logic is provided entirely by the EJB3 tier, the web tier assembles the logic and presents a coherent application to the user. The DVD store uses a combination of JSP (JavaServer Pages) and JSF (JavaServer Faces) to manage the user experience.

Using a JSF component-based architecture has many advantages. The MVC (model view controller) architecture allows for a very clean separation of the web application responsibilities. The JSF backing beans provide a rich model for the interface and map user requests onto the EJB3 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.

Security

Security is entirely externalized in the DVD Store, and there are no explicit security checks in the application. We have defined a JAAS (Java Authentication and Authorization Service) domain that understands the username and passwords in the relational database. Using simple declarative security statements, access to the application can be restricted to authenticated users. Using JAAS, the security domain may be replaced with one that consulted an LDAP server or made use of another authentication technology without changing any application code.

Clustering

Clustering support was added to both the web tier and the EJB3 tier. The application uses HTTP session replication and clustered single sign on the web tier. If any JBoss node were to fail, a front-end HTTP load balancer could fail the
request over to another JBoss box. 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 maintained primarily read-only data such as product categories. Once this data is loaded into the cache, the EJB3 tier no longer needs to consult the database to load that data. The application did not have to be changed in any way to support the cache. All clustering options in JBoss make use of the JBossCache/JGroups stack and require multicasting communication between the nodes.
The MySQL Database

The backend database employed in this test is 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 are employed.

The Database Schema

The MySQL DVD store database was comprised of seven main tables and one other small table (see Table 1).

<table>
<thead>
<tr>
<th>Table</th>
<th>Columns</th>
<th>Number of Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERS</td>
<td>CUSTOMERID, FIRSTNAME, LASTNAME, ADDRESS1, ADDRESS2, CITY, STATE, ZIP, COUNTRY, REGION, EMAIL, PHONE, CREDITCARDTYPE, CREDITCARD, CREDITCARDEXPIRATION, USERNAME, PASSWORD, AGE, INCOME, GENDER, PROD_ID_IDX, PROD_ID1, PROD_ID2 … PROD_ID10</td>
<td>200 million</td>
</tr>
<tr>
<td>ORDERS</td>
<td>ORDERID, ORDERDATE, CUSTOMERID, NETAMOUNT, TAX, TOTALAMOUNT</td>
<td>120 million</td>
</tr>
<tr>
<td>ORDERLINES</td>
<td>ORDERLINEID, ORDERID, PROD_ID, QUANTITY, ORDERDATE</td>
<td>600 million</td>
</tr>
<tr>
<td>CUST_HIST</td>
<td>CUSTOMERID, ORDERID, PROD_ID</td>
<td>600 million</td>
</tr>
<tr>
<td>PRODUCTS</td>
<td>PROD_ID, CATEGORY, TITLE, ACTOR, PRICE, SPECIAL, COMMON_PROD_ID1, COMMON_RATING1, COMMON_PROD_ID2, COMMON_RATING2, COMMON_PROD_ID3, COMMON_RATING3</td>
<td>1 million</td>
</tr>
<tr>
<td>INVENTORY</td>
<td>PROD_ID, QUAN_IN_STOCK, SALES</td>
<td>1 million</td>
</tr>
<tr>
<td>REORDER</td>
<td>PROD_ID, DATE_LOW, QUAN_LOW, DATE_REORDERED, QUAN_REORDERED, DATE_EXPECTED</td>
<td>variable</td>
</tr>
<tr>
<td>CATEGORIES</td>
<td>CATEGORY, CATEGORYNAME</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 1: DVD Store Database Schema

The CUSTOMERS table was pre-populated with two hundred million customers, one hundred million US customers and one hundred million customers from the rest of the world. The ORDERS table was pre-populated with ten million orders per month for a full year. The ORDERLINES table was pre-populated with an average of 5 items per order. The PRODUCTS table contained one million DVD titles, each with a principal actor listed for search purposes. For realism titles
and actor names are generated by taking combinations of real movie titles and actor names. Additionally, the CATEGORIES table contained the 16 DVD categories.

The schema is fully documented in the database build script in Appendix C.
Section 6

Conclusions

The DVD Store application was ported in less than a week, with no changes required to the database layer. Through the use of EJB3 entity beans to map the relational database data to application objects, the application now will work without changes with any database supported by JBoss, including Microsoft SQL Server and Oracle. The use of stateless session beans simplifies the implementation of the application’s business logic. The web front end, implemented with Java Server Faces, may be easily modified in appearance or language. Built-in JBoss features such as Java Authentication and Authorization Service and the JBoss Cache provide secure sign-on and clustering.

All of these features were built and successfully tested on two Power Edge 1855 blade servers, with the MySQL database running on a PowerEdge 2800 server.
Acknowledgements

The Dell authors would like to acknowledge the contributions of Norman Richards to this project in quickly porting the DVD Store code to the JBoss platform, working with us to implement it on the Dell PowerEdge 1855 blade servers, and documenting it in a clear manner.
Appendix A. Products Table EJB3 Entity Bean

@javax的企业
@Table(name="PRODUCTS")
public class Product
    implements Serializable
{
    long productId;
    String title;
    String actor;
    float price;
    Category category;
    Product relatedProduct;

    @Id(generate=GeneratorType.AUTO)
    @Column(name="PROD_ID")
    public long getProductId() {
        return productId;
    }
    public void setProductId(long id) {
        this.productId = id;
    }

    @ManyToOne
    @JoinColumn(name="CATEGORY")
    public Category getCategory() {
        return category;
    }
    public void setCategory(Category category) {
        this.category = category;
    }

    @Column(name="TITLE")
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }

    @Column(name="ACTOR")
    public String getActor() {
        return actor;
    }
    public void setActor(String actor) {
        this.actor = actor;
    }

    @Column(name="PRICE")
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }

    @ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="COMMON_PROD_ID")
    public Product getRelatedProduct() {
return relatedProduct;
}
public void setRelatedProduct(Product related) {
    this.relatedProduct=related;
}
Appendix B. Purchase Method of DvdStoreBean Session Bean

```java
public Order purchase(Customer customer, List<OrderLine> lines)
    throws InsufficientQuantityException {
    Order order = new Order();
    order.setCustomer(customer);
    order.setOrderDate(new Date());
    em.persist(order);

    List<Product> errorProducts = new ArrayList<Product>();
    float total = 0;
    for (OrderLine line : lines) {
        total += line.getQuantity() * 
            line.getProduct().getPrice();
        line.setOrderDate(order.getOrderDate());
        line.getPk().setOrderId(order.getOrderId());
        em.persist(line);

        Inventory inv = getInventory(line.getProduct());
        if (!inv.order(line.getQuantity())) {
            errorProducts.add(line.getProduct());
        }

        History history = 
            new History(customer, order, line.getProduct());
        em.persist(history);
    }

    if (errorProducts.size()>0) {
        ctx.setRollbackOnly();
        throw new InsufficientQuantityException(errorProducts);
    }

    order.setNetAmount(total);
    order.setTax((float) (order.getNetAmount() * .0825));
    order.setTotalAmount(order.getNetAmount() + 
        order.getTax());

    em.persist(order);

    return order;
}
```
Appendix C. mysqlds2 Build Script

-- mysqlds2_create_db.sql: DVD Store Database Version 2.1 Build Script - MySQL version
-- Copyright (C) 2005 Dell, Inc. <dave_jaffe@dell.com> and <todd_muirhead@dell.com>
-- Last updated 5/13/05

-- Database
DROP DATABASE IF EXISTS DS2;
CREATE DATABASE DS2;
USE DS2;

-- Tables
CREATE TABLE CUSTOMERS
  ( CUSTOMERID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    FIRSTNAME VARCHAR(50) NOT NULL,
    LASTNAME VARCHAR(50) NOT NULL,
    ADDRESS1 VARCHAR(50) NOT NULL,
    ADDRESS2 VARCHAR(50),
    CITY VARCHAR(50) NOT NULL,
    STATE VARCHAR(50),
    ZIP INT,
    COUNTRY VARCHAR(50) NOT NULL,
    REGION TINYINT NOT NULL,
    EMAIL VARCHAR(50),
    PHONE VARCHAR(50),
    CREDITCARDTYPE INT NOT NULL,
    CREDITCARD VARCHAR(50) NOT NULL,
    CREDITCARDEXPIRATION VARCHAR(50) NOT NULL,
    USERNAME VARCHAR(50) NOT NULL,
    PASSWORD VARCHAR(50) NOT NULL,
    AGE TINYINT,
    INCOME INT,
    GENDER VARCHAR(1)
  )
  TYPE=InnoDB;

CREATE TABLE CUST_HIST
  ( CUSTOMERID INT NOT NULL,
    ORDERID INT NOT NULL,
    PROD_ID INT NOT NULL
  )
  TYPE=InnoDB;

CREATE TABLE ORDERS
  ( ORDERID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ORDERDATE DATE NOT NULL,
    CUSTOMERID INT,
    NETAMOUNT NUMERIC(12,2) NOT NULL,
    TAX NUMERIC(12,2) NOT NULL,
    TOTALAMOUNT NUMERIC(12,2) NOT NULL
  )
  TYPE=InnoDB;
CREATE TABLE ORDERLINES
(
ORDERLINEID SMALLINT NOT NULL,
ORDERID INT NOT NULL,
PROD_ID INT NOT NULL,
QUANTITY SMALLINT NOT NULL,
ORDERDATE DATE NOT NULL
)
TYPE=InnoDB;

CREATE TABLE PRODUCTS
(
PROD_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
CATEGORY TINYINT NOT NULL,
TITLE VARCHAR(50) NOT NULL,
ACTOR VARCHAR(50) NOT NULL,
PRICE NUMERIC(12,2) NOT NULL,
SPECIAL TINYINT,
COMMON_PROD_ID INT NOT NULL
)
;

CREATE TABLE INVENTORY
(
PROD_ID INT NOT NULL PRIMARY KEY,
QUAN_IN_STOCK INT NOT NULL,
SALES INT NOT NULL
)
TYPE=InnoDB;

CREATE TABLE CATEGORIES
(
CATEGORY TINYINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
CATEGORYNAME VARCHAR(50) NOT NULL
)
TYPE=InnoDB;

INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (1,'Action');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (2,'Animation');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (3,'Children');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (4,'Classics');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (5,'Comedy');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (6,'Documentary');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (7,'Drama');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (8,'Family');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (9,'Foreign');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (10,'Games');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (11,'Horror');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (12,'Music');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (13,'New');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (14,'Sci-Fi');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (15,'Sports');
INSERT INTO CATEGORIES (CATEGORY, CATEGORYNAME) VALUES (16,'Travel');

CREATE TABLE REORDER
(
PROD_ID INT NOT NULL,
DATE_LOW DATE NOT NULL,
QUAN_LOW INT NOT NULL,
...
DATE_REORDERED DATE,
QUAN_REORDERED INT,
DATE_EXPECTED DATE
)
TYPE=InnoDB;