JBoss to Geronimo - EJB-Session Beans Migration

{scrollbar}

top
A typical J2EE application may contain Enterprise JavaBeans or EJBs. These beans contain the application's business logic and live business data. Although it is possible to use standard Java objects to contain your business logic and business data, using EJBs addresses many of the issues of using simple Java objects, such as scalability, lifecycle management and state management.

This document discusses one type of EJBs, the Session EJB. This type of EJBs are useful for mapping business process flow (or equivalent application concepts). There are two types of Session EJBs, stateless and stateful. This article addresses both these Session Bean types and shows how to use them in a practical scenario.

EJBs hold conversations with clients. A conversation is basically an interaction between the EJB and the client and these interactions are composed of method calls by the clients to the EJBs. Stateful session beans retain state on behalf of a client. This means that if the state of the bean changes during a client's method call, this state is retained for subsequent calls by the same client. A stateless session bean on the other hand retains no conversational state from method to method. In other words, it is expected to hold its conversational state for only a single method call.

This article is organized in the following sections:

EJB implementation analysis #analysis

EJB implementation may vary from one vendor to another. The purpose of this section is to provide a session bean specific feature-to-feature comparison between JBoss and Apache Geronimo so you can clearly identify the differences and plan accordingly before migration.

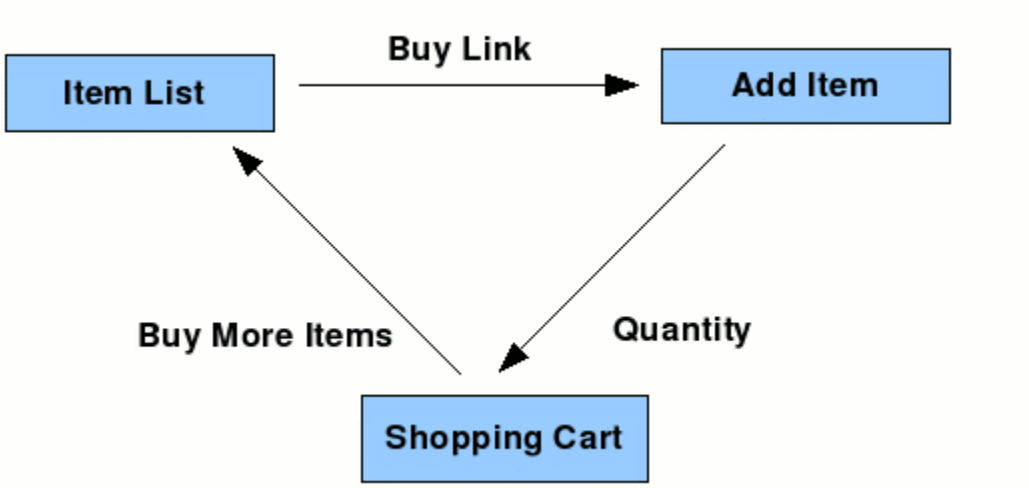
Feature	JBoss v4.0.5	Apache Geronimo (OpenEJB)
Stateful and stateless Session Beans	supported	supported
BMP (Bean Managed Persistence) Entity Beans	supported	supported
CMP (Container Managed Persistence) Entity Beans	supported	supported
Message driven beans (MDBs)	supported	supported
Interoperability using RMI-IIOP or JAXRPC	supported	supported
Ability to expose stateless session beans and MDBs as Web Services	supported	supported
Support for sending and receiving messages via Web Services	supported	supported
Easy provisioning and hot deployment of EJB and JMX-based Web Services	supported	supported
Access to EJBs from external CORBA objects	supported	supported

Back to Top

Sample application #sample

This is a simple eCommerce web site which sells computer related items in both retail and wholesale markets. To get the whole sale prices of items, customer has to buy at least given number of items. Otherwise he/she will get only retail prices of the item. End user of the application can add those items to their shopping cart. A stateless session bean has been used to manage the discount calculation business workflow activities while shopping cart of the application has been implemented with the help of a stateful session bean.

The following figure illustrates the application flow:



A customer of this application directly login to the item list page. It will give all the details of the items they have currently on their stock. To buy one of these items in the stock, use the Buy link which comes with each item. It will forward customer in to the Add Item to Cart web page. To proceed buying, customer has to enter the quantity of items he/she needs. If this quantity exceeds the minimum discount quantity, discounts will be granted and added items in to the customer's shopping cart. Shopping cart web page displays the list of items already added to the cart and it enables removing of items from the cart. This application will be not allow to add the same item shopping cart twise.

Application classes and JSP pages

- org.apache.geronimo.samples.computer.dto
  - ItemDTO - Data transfer object for item related information between web and ejb tiers.
  - TransactionDTO - Data transfer object for shopping cart transaction related information between web and ejb tiers.
- org.apache.geronimo.samples.computer.ejb
  - ItemServiceBean - Stateless session bean to handle item realated workflow activities.
  - ShoppingCartBean - Stateful session bean to handle shopping cart related activities fro shopping cart activities.
- org.apache.geronimo.samples.computer.web
  - ItemServiceDispatchServlet - A servlet to dispatch item service related activities from front end to web tier.
  - ShoppingCartDispatchServlet - A servlet to dispatch shopping cart related activities from fron end to web tier.

This eCommerce sample application also includes the following JSP pages:

- buy_item.jsp - Accepts quantity of items customer buys.
- error.jsp - Display error conditions of the application.
- index.jsp - Forward in to the item list of the application.
- list_items.jsp - Display list of items in the stock.
- shopping_cart.jsp - List of transactions added to the shopping cart will be displayed.

Tools used

The tools used for developing and building the Computer acessories seller application are:

Eclipse

The Eclipse IDE was used for development of the sample application. This is a very powerful and popular open source development tool. Integration plug-ins are available for both JBoss and Geronimo. Eclipse can be downloaded from the following URL:
http://www.eclipse.org

Apache Ant

Ant is a pure Java build tool. It is used for building the war files and populating the database for the Online Brokerage application. Ant can be downloaded from the following URL:
http://ant.apache.org

Back to Top

The JBoss enviroment #jboss

This section shows you how and where the sample JBoss reference environment was installed so you can map this scenario to your own implementation. Note that for this migration example JBoss v4.0.5 was used.

Detailed instructions for installing, configuring, and managing JBoss are provided in the product documentation. Check the product Web site for the most updated documents.

The following list highlights the general tasks you will need to complete to install and configure the initial environment as the starting point for deploying the sample application.

1. Download and install JBoss v4.0.5 as explained in the product documentation guides. From now on the installation directory will be referred as **<jboss_home>**
2. Create a copy of the default JBoss v4.0.5 application server. Copy recursively **<jboss_home>\server\default** to **<jboss_home>\server\<your_server_name>**
3. Start the new server by running the **run.sh -c <your_server_name>** command from the **<jboss_home>\bin** directory.
4. Once the server is started, you can verify that it is running by opening a Web browser and pointing it to this URL: http://localhost:8080. You should see the JBoss Welcome window and be able to access the JBoss console.
5. Once the application server is up and running, the next step is to install and configure all the remaining prerequisite software required by the sample application. This step is described in the following section.

Install and configure prerequisite software

In order to build and run this sample application included in this article, you need to install and configure the Ant build tool.

Configure Ant

As mentioned before, Apache Ant is used to build the binaries for the Online Brokerage application. If you do not have Ant installed this is a good time for doing it and make sure that **<ant_home>/bin** directory is added to the system's path variable.

Apache Ant can be downloaded from the following URL:

http://ant.apache.org

Configure XDoclet

XDoclet is going to be used as build tool for the configuration file generation. It is an open source code generation engine. It enables Attribute-Oriented Programming for java. In short, this means that you can add more significance to your code by adding meta data (attributes) to your java sources. This is done in special JavaDoc tags.
Although XDoclet originated as a tool for creating EJBs, it has evolved into a general-purpose code generation engine. XDoclet consists of a core and a constantly growing number of modules. It is fairly straight forward to write new modules if there is a need for a new kind of component.
http://xdoclet.sourceforge.net/xdoclet/index.html

Just extract the latest version of the XDoclet and set the **xdoclet.home** parameter in to the **build.properties** file.

Build the sample application

The computer acessories selling application included with this article provides an Ant script that you will use in order to build the application. Download the computer acessories selling application from the following link:

Computer

After extracting the zip file, a **computer** directory is created. In that directory open the build.properties file and edit the properties to match your environment as shown in the following example:

xmlsolidbuild.properties ## Set the Geronimo 1.1 home here geronimo.home=<geronimo_home> ## Set XDoclet 1.2.3 Home xdoclet. home=<xdoclet_home>

Before starting the build process just set the correct paths for the **geronimo.home** and **xdoclet.home** entries in the **build.properties** file in the **config** directory.

From a command prompt or shell go to the **computer** directory and run **ant jboss**. This will build the ear file and place it directly in the **releases/jboss** directory.

Deploy the sample application

To deploy the sample application just copy the **computer.ear** will be created under the **computer/releases/jboss** folder to the **<jboss_home>/server/<your_server_name>/deploy** folder.

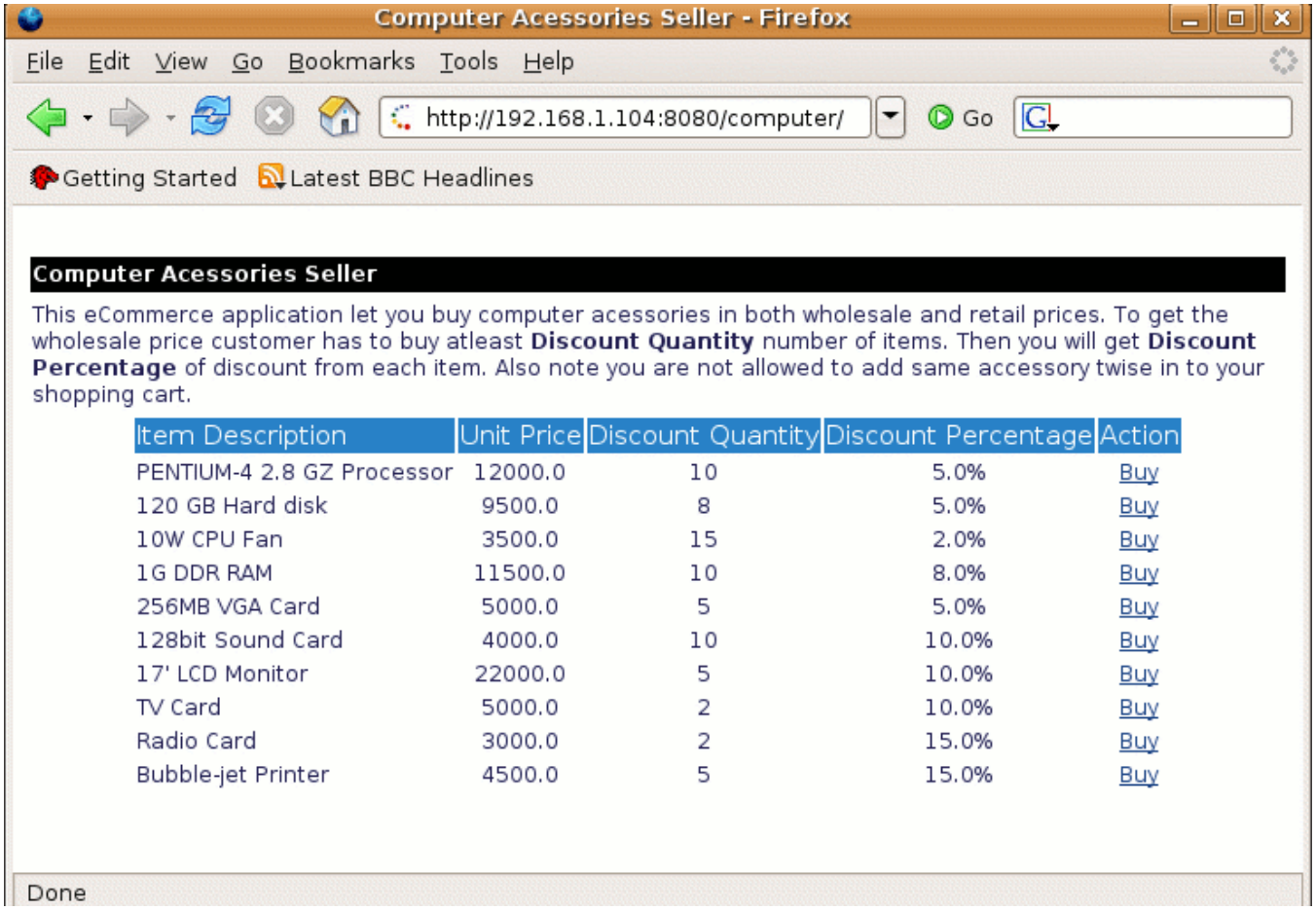
If JBoss is already started, it will automatically deploy and start the application; otherwise, the application will be deployed and started at the next startup.

Test the sample application

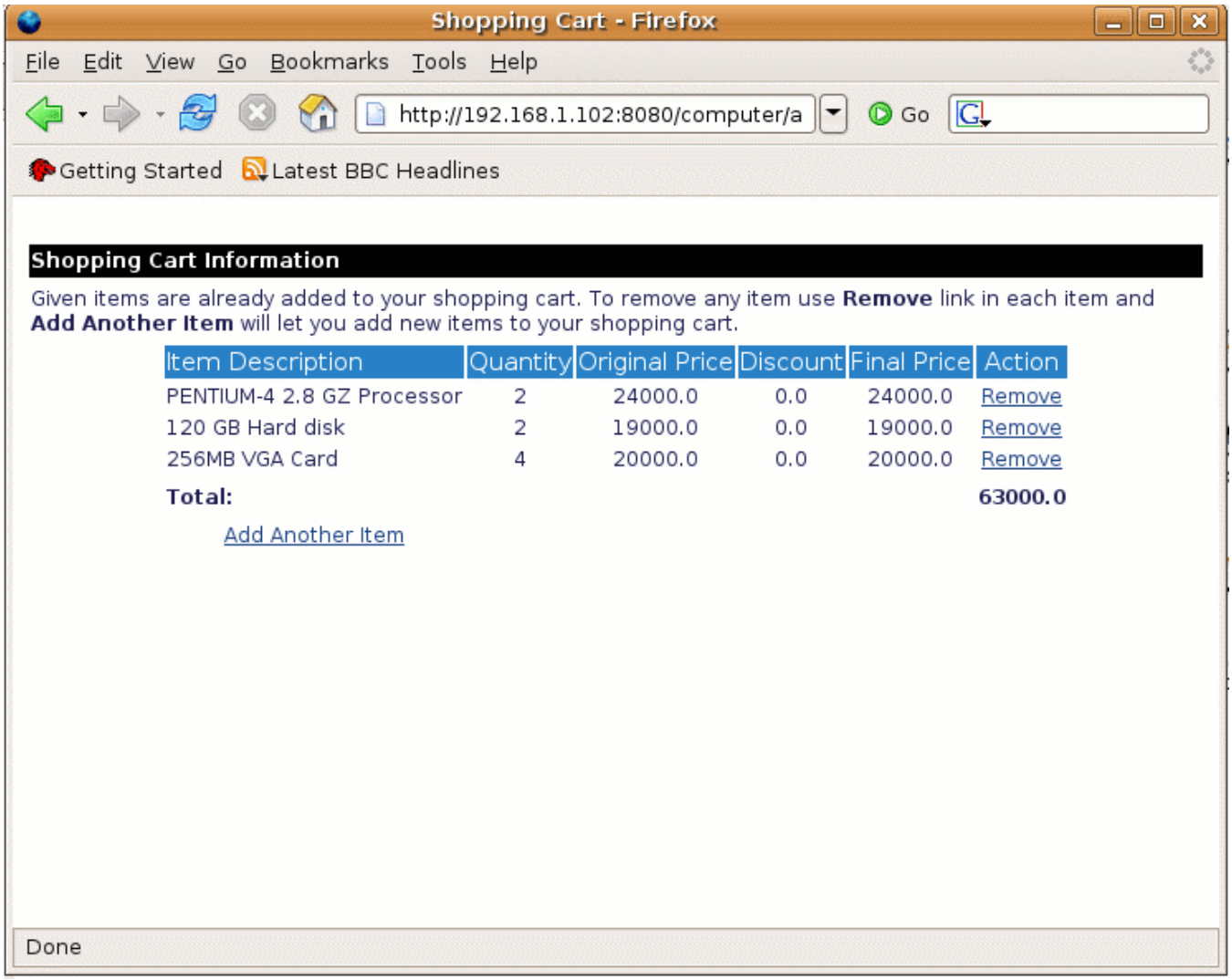
To test the application, open a Web browser and access the following URL:

http://localhost:8080/computer

This brings up the item list page of the computer acessories selling application. You can buy computer acessories using this application. The application is now configured and running.



Following is the demonstration of a shopping cart of the sample application.



Back to Top

The Geronimo enviroment #geronimo

Download and install Geronimo from the following URL:

http://geronimo.apache.org/downloads.html

The release notes available there provide clear instructions on system requirements and how to install and start Geronimo. Throughout the rest of this article we will refer to the Geronimo installation directory as **<geronimo_home>**.

TCP/IP ports conflict
If you are planning to run JBoss and Geronimo on the same machine consider to change the default service ports on, at least, one of these servers.

Back to Top

Step-by-step migration #migration

When you built the computer acessories selling sample application, Ant packaged the deployment descriptors for both JBoss **jboss.xml** and Geronimo **openejb-jar.xml** as they were already provided by the sample application. These files are located in the **computer/config** directory.

The following example shows the JBoss deployment descriptor.

xmlsolidjboss.xml <?xml version="1.0" encoding="UTF-8"?> <jboss> <enterprise-beans> <session> <ejb-name>ShoppingCart</ejb-name> <local-jndi-name>ShoppingCart</local-jndi-name> <method-attributes> </method-attributes> </session> <session> <ejb-name>ItemService</ejb-name> <local-jndi-name>ItemService</local-jndi-name> <method-attributes> </method-attributes> </session> </enterprise-beans> <resource-managers> </resource-managers> </jboss>

Compare it with the contents of the Geronimo deployment plan shown in the following example.

xmlsolidopenejb-jar.xml <?xml version="1.0" encoding="UTF-8"?> <openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"> <dep:moduleId> <dep:groupId>org.apache.geronimo.samples</dep:groupId> <dep:artifactId>ComputerEJB</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>car</dep:type> </dep:moduleId> <dep:dependencies /> <dep:hidden-classes/> <dep:non-overridable-classes/> </dep:environment> <enterprise-beans> <session> <ejb-name>ShoppingCart</ejb-name> <ejb-ref> <ref-name>ejb/ItemServiceLocal</ref-name> <ejb-link>ItemService</ejb-link> </ejb-ref> </session> <session> <ejb-name>ItemService</ejb-name> </session> </enterprise-beans> </openejb-jar>

First difference can be clearly noted is Geronimo specific configuration has more additional information than JBoss specific one. That part of the Geronimo configuration file is quite similar to a Maven 2 build script. Both of these given configuration files has EJB information. JBoss uses local JNDI names to link their EJBs while Geronimo directly use EJB's name. In addition to above differences also **openejb-jar.xml** file clearly gives the ejb reference information than **jboss.xml** file.

Looking at the web archive related configuration files give you few more differnces.

xmlsolidjboss-web.xml <?xml version="1.0" encoding="UTF-8"?> <jboss-web> <!-- EJB Local References --> <ejb-local-ref> <ejb-ref-name>ejb/ItemServiceLocal</ejb-ref-name> <local-jndi-name>ItemService</local-jndi-name> </ejb-local-ref> <ejb-local-ref> <ejb-ref-name>ejb/ShoppingCartLocal</ejb-ref-name> <local-jndi-name>ShoppingCart</local-jndi-name> </ejb-local-ref> </jboss-web> xmlsolidgeronimo-web.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1" xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleId> <dep:groupId>org.apache.geronimo.samples</dep:groupId> <dep:artifactId>ComputerWeb</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>car</dep:type> </dep:moduleId> <dep:dependencies /> <dep:hidden-classes/> <dep:non-overridable-classes/> </dep:environment> <naming:ejb-local-ref> <naming:ref-name>ejb/ItemServiceLocal</naming:ref-name> <naming:ejb-link>ItemService</naming:ejb-link> </naming:ejb-local-ref> <naming:ejb-local-ref> <naming:ref-name>ejb/ShoppingCartLocal</naming:ref-name> <naming:ejb-link>ShoppingCart</naming:ejb-link> </naming:ejb-local-ref> </web-app>

jboss-web.xml map the EJBs from their JNDI names as given the above while **geronimo-web.xml** uses directly EJB's name. The reference names given in each mapping will be used refer EJBs from the Servlets. **web.xml** file of the WAR file contains more information about each EJB reference name, which will be common to the both Geronimo and JBoss flavours of this application.

Build the sample application

Build the migrated Geronimo version of the sample application by running following command from the **computer** directory.
ant geronimo
It will create **computer.ear** file in the **computer/releases/geronimo** folder.

Deploy the migrated application

To deploy the migrated Computer acessories selling application, make sure the Geronimo server is up and running.

Open Geronimo console in your browser and follow the given steps:

1. Travel **Deploy New** from the **Console Navigation**.
2. Load **computer.ear** from **computer/releases/geronimo** folder in to the **Archive** input box.
3. Press **Install** button to deploy application in the server.

Once the application is deployed, open a Web browser and access the following URL:
http://localhost:8080/computer

Back to Top

Summary #summary

This article has shown how to migrate a sample application that uses Session Beans, from JBoss v4.0.5 to Apache Geronimo. This article provided step-by-step instructions to build the application, deploy and run it, and then migrate it to the Geronimo environment.

The following list summarizes the major differences found during this sample application migration.

- In order to deploy an EJB jar file in JBoss you need to just copy the configuration file to the deploy directory but in Geronimo you can use either deployer tool, console or hot deployment directory.
- The contents of the deployment plans for EJB jar files in JBoss and in Geronimo are almost similar except for the starting part of the Geronimo which is more similar to a Maven 2 build file.