

JBoss to Geronimo - EJB-CMP Migration

An entity bean is defined as a representation of persistent data that has the ability to read from database and populate its fields with data. It can be updated and stored back to the database. There are two types of entity beans: Bean-Managed Persistence(BMP) and Container-Managed Persistent (CMP). This article covers an example of a CMP, more specifically, a CMP application migration. For this type of entity bean, actual code must be written to handle persistent operations such as loading, saving and finding data. The developer must use persistence API such as JDBC to select, insert, update, delete from a database.

This article is organized in the following sections:

- [CMP implementation analysis](#)
- [Sample application](#)
- [The JBoss environment](#)
- [The Geronimo environment](#)
- [Step-by-step migration](#)
- [Summary](#)

CMP implementation analysis

CMP implementation may vary from one vendor to another. The purpose of this section is to provide a CMP specific feature-to-feature comparison between JBoss v4 and Apache Geronimo so you can clearly identify the differences and plan accordingly before migration.

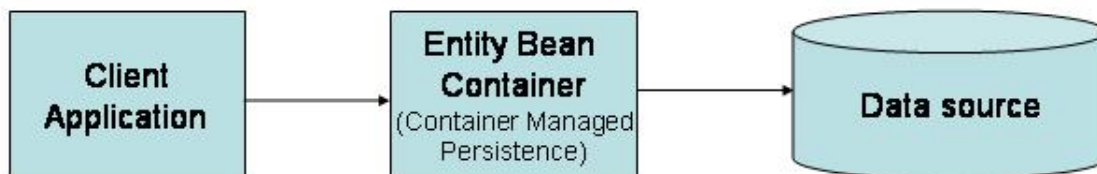
Feature	JBoss v4	Apache Geronimo
EJB Container	JBoss AS 4.0 comes with its own EJB Container implementation.	Geronimo uses OpenEJB as its EJB Container.

[Back to Top](#)

Sample application

The [Loan CMP Sample](#) application is very simple. When the command line client is run, an entry is made into the database. The `findByPrimaryKey()` method of the `CustomerHomeRemote` interface is called and the field values of the returned `CustomerRemote` object are printed to the console. This is followed by a call to the `findBySssNo()` method after which the field values of the returned `CustomerRemote` object are printed to the console.

The following figure illustrates the application flow:



The user runs the command line client which then either creates an entity bean (which then adds itself to the datasource) or asks for one, by primary key, which is created from information that is stored in the database.

[Back to Top](#)

Application Beans

The Loan CMP application consists of the following packages:

- `com.ibm.demo.entity.client`
 - `CMPCClient`
 - contains the main class that is called from the console.
- `com.ibm.demo.entity.bmp`
 - `CustomerBean`
 - implements `javax.ejb.EntityBean`
 - fields of the bean are defined here
 - contains business methods corresponding to the methods exposed by the `CustomerRemote` interface.
 - Contains callback methods that are called by the container to manage the bean. These methods include the create and find methods which use jdbc to make entries to the database and to search the database.
 - Has a helper method that looks up the datasource through jndi.
 - `CustomerRemote`

- interface that extends javax.ejb.EJBObject
- exposes the setter and getter methods of the EJB
- CustomerHomeRemote
 - interface that extends javax.ejb.EJBHome
 - exposes the create and find methods of the EJB

[Back to Top](#)

Tools used

The tools used for developing and building the Loan CMP sample 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 Maven

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM). Maven can manage a project's build, reporting and documentation from a central piece of information.

For this migration example Maven 1.0.2 was used. Maven can be downloaded from the following URL:

<http://maven.apache.org>

[Back to Top](#)

Sample database

The sample database for the Loan CMP application has only one table. This is an in-memory table. The **MEMORY** storage engine creates tables with contents that are stored in just in memory. These were formerly known as HEAP tables.

The following table describes the fields of the **CUSTOMER** table.

Field	data type
id	INTEGER
name	VARCHAR(45)
birthdate	DATE
sss_no	VARCHAR(25)
address	VARCHAR(60)
annual_salary	DOUBLE
loan_amount	DOUBLE

[Back to Top](#)

The JBoss environment

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 v4.0.2 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 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 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.

[Back to Top](#)

Install and configure prerequisite software

In order to build and run the Loan CMP application included in this article, you need to install and configure the build tool and the database that is used by the application.

Modify database settings

This application is using the HSQL database that comes as part of the JBoss bundle. You need to modify the script for creating the database. Edit the **localDB.script** file located in the following directory:

```
<jboss_home>\server\<your_server_name>\data\hypersonic
```

Add at the top of the **localDB.script** file the content of the following example in order to create the sample HSQL database.



Make sure JBoss is not running at the time of modifying this file.

```
CREATE MEMORY TABLE CUSTOMER(ID INTEGER NOT NULL PRIMARY KEY,NAME VARCHAR(45),BIRTHDATE DATE,SSS_NO VARCHAR(25),
ADDRESS VARCHAR(60),ANNUAL_SALARY DOUBLE,LOAN_AMOUNT DOUBLE)
```

Configure Maven

As mentioned before, Apache Maven is used to build the binaries for the Loan CMP application. If you do not have Maven installed this is a good time for doing it.

Apache Maven can be downloaded from the following URL:

<http://maven.apache.org>

[Back to Top](#)

Build the sample application

In order to build the loan application a Maven script has been provided. Download the Loan application from the following link:

[Loan CMP Sample](#)

After extracting the zip file, a **loan-cmp** directory will be created. From now on, this directory will be referred as **<cmp_home>**. In that directory open the project.properties file. Edit the maven.jboss.home property to match your environment. It is important that you use **"//"** on the windows platform as is done below.

maven.jboss.home=Z://JBoss-4.0.2

From a command prompt or shell go to the **<cmp_home>** directory and run the following command:

```
maven
```

This will build a jar and a war file and put them in the **<cmp_home>/target** folder. The jar created by the Maven build contains a JBoss specific deployment descriptor, the **jboss.xml** file in located the META-INF directory of the JAR is shown in the following example:

JBoss deployment descriptor - jboss.xml

```
<?xml version="1.0"?>

<jboss>
  <enterprise-beans>
    <entity>
      <ejb-name>CustomerEJB</ejb-name>
      <jndi-name>CustomerHomeRemote</jndi-name>
    </entity>
  </enterprise-beans>
</jboss>
```

The jndi-name element is used to bind the CustomerEJB to the name CustomerHomeRemote in JNDI.

Running **maven** will also build the a war file and put it in the <cmp_home>/target folder. The war created by the maven build contains a JBoss specific Web application deployment descriptor, the jboss-web.xml file in the WEB-INF directory of the WAR is shown in the following example:

JBoss deployment descriptor - jboss.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
  <ejb-ref>
    <ejb-ref-name>ejb/CustomerHome</ejb-ref-name>
    <jndi-name>CustomerHomeRemote</jndi-name>
  </ejb-ref>
</jboss-web>
```

[Back to Top](#)

Deploy the sample application

To deploy the Loan CMP application in JBoss, copy the **entity-ejb-cmp.jar** and **entity-ejb.war** files you just built with Maven to the following directory:

```
<jboss_home>\server\<your_server_name>\deploy
```

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.

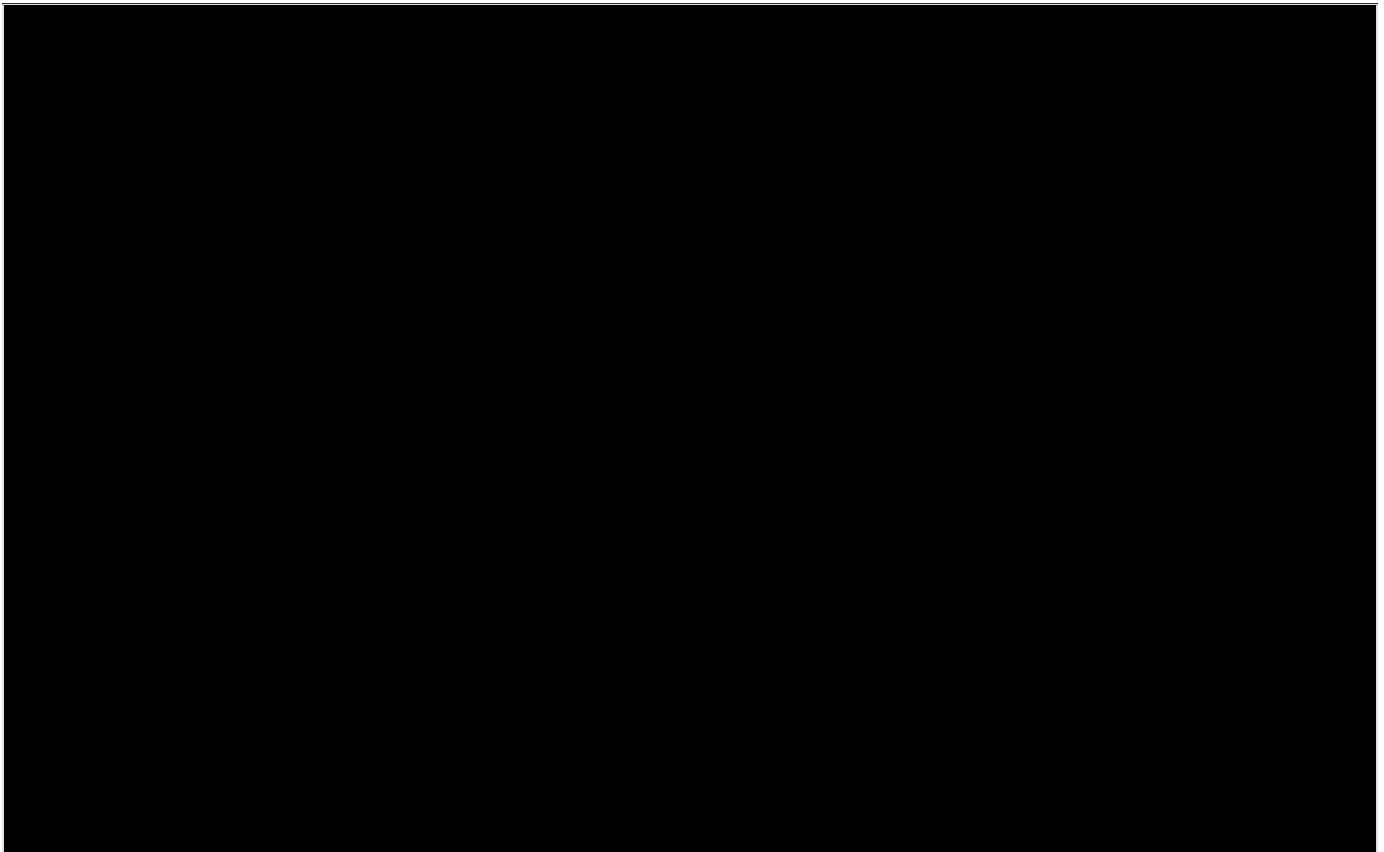
[Back to Top](#)

Test the sample application

To test the sample client application type the following command from the <cmp_home> directory:

```
maven run:client
```

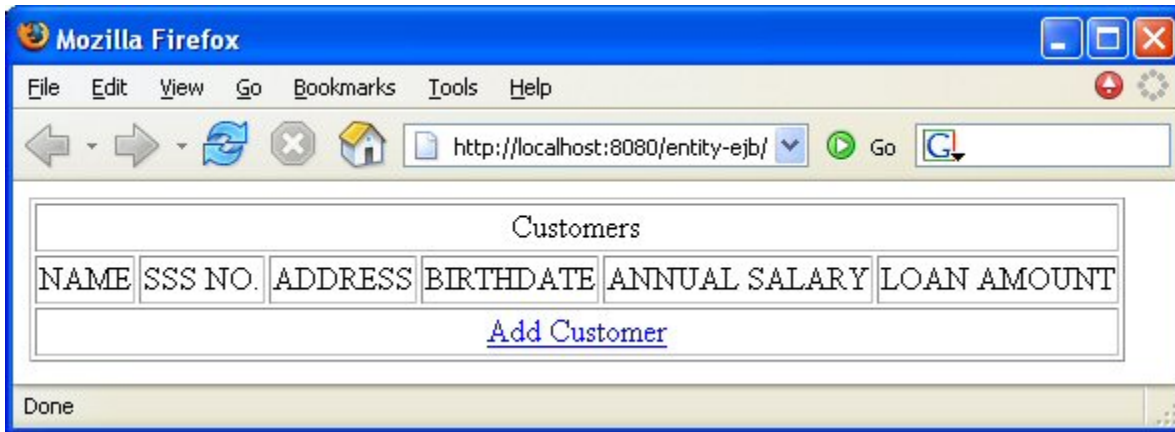
When you run this command, you will receive a list of all the loans that were retrieved from the database, you should see a screen similar to the one shown in the following example:



To test the sample Web application point your browser to:

<http://localhost:8080/entity-ejb>

You should see the following screen:



Click on **Add Customer**. Enter the new customer information then click **Create**, this will take you to the first page showing the updated list of customers.



When you are entering the **BIRTHDATE** make sure you specify the date with the following format (MM/DD/YYYY)

[Back to Top](#)

The Geronimo environment

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](#)

Configure resources

For this scenario the Loan CMP will use directly the SystemDatabase from Geronimo. In this case there is no need to set up a new connector for the SystemDatabase since it is already configured as the DefaultDatasource.

Start the Geronimo server

Ensure that Geronimo is up and running. If the server has not been started yet, do so by typing the following command:

```
<geronimo_home>/bin/startup.sh
```

Once the server is started you should see a screen similar as the one illustrated in the following example:



Configure database via Geronimo Console

Access the Geronimo Console by pointing your Web browser to the following URL:

<http://localhost:8080/console>

Enter the following **system** as the username and **manager** as the password, click **Login**.

Once logged in, on the bottom left corner from the left navigation panel click on **DB Manager**. In the text area labeled **SQL Command/s** enter the following SQL statement and click **Run SQL**; this will create the table used by the Entity Bean.

```
CREATE TABLE CUSTOMER(ID INTEGER NOT NULL PRIMARY KEY,NAME VARCHAR(45),BIRTHDATE DATE,SSS_NO VARCHAR(25),ADDRESS
VARCHAR(60),ANNUAL_SALARY DOUBLE,LOAN_AMOUNT DOUBLE)
```

Done

[Back to Top](#)

Configure Maven

You should set the `maven.geronimo.home` property in `project.properties` to point to your `<geronimo_home>` directory.

[Back to Top](#)

Step-by-step migration

The same EJB jar file that was created and deployed in JBoss may be deployed in Geronimo with no changes to its contents but you still need to edit the `jndi` properties of sample application. Edit the `jndi.properties` file located in the `<cmp_home>/jndi` directory as shown in the following example:

jndi.properties

```
#####  
### JBoss Settings  
#####  
#java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory  
#java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces  
#java.naming.provider.url=localhost  
  
#####  
### Geronimo Settings  
#####  
java.naming.factory.initial=org.openejb.client.RemoteInitialContextFactory  
java.naming.provider.url=localhost:4201  
java.naming.security.principal=username  
java.naming.security.credentials=passwd
```

The following example shows the **customer-ejb.xml** deployment plan used for deploying the EJB application, this deployment plan is located in the <cmp_home>/dd directory.

customer-ejb.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<openejb-jar  
  xmlns="http://www.openejb.org/xml/ns/openejb-jar"  
  xmlns:naming="http://geronimo.apache.org/xml/ns/naming"  
  xmlns:security="http://geronimo.apache.org/xml/ns/security"  
  xmlns:sys="http://geronimo.apache.org/xml/ns/deployment"  
  configId="CustomerEJB"  
  parentId="geronimo/system-database/1.0/car">  
<enterprise-beans>  
  <entity>  
    <ejb-name>CustomerEJB</ejb-name>  
    <jndi-name>CustomerHomeRemote</jndi-name>  
    <local-jndi-name>CustomerRemote</local-jndi-name>  
    <resource-ref>  
      <ref-name>jdbc/ibm-demo</ref-name>  
      <resource-link>SystemDataSource</resource-link>  
    </resource-ref>  
  </entity>  
</enterprise-beans>  
</openejb-jar>
```

This plan sets **geronimo/system-database/1.0/car** as the parent. What follows is the definition of the entity bean. The **jndi-name** element indicates the jndi name of the entity bean's home interface **CustomerHomeRemote**. This is the name that the Loan CMP sample application will lookup in the jndi context. The element **local-jndi-name** indicates the jndi name of the local interface, which in this case happens to be a remote interface, **CustomerRemote**. Next, a reference to the **SystemDataSource** is defined giving the application access to the database.

The Web Application client can be directly deployed in Geronimo. This is because the build step packages both the JBoss **jboss-web.xml** and Geronimo **geronimo-web.xml** specific deployment plans in the war file. You can see both of these files in the <cmp_home>/src/webapp/WEB-INF directory.

The **geronimo-web.xml** deployment plan should look like the following example.

Geronimo deployment plan geronimo-web.xml

```
<web-app xmlns="http://geronimo.apache.org/xml/ns/web"
  xmlns:naming="http://geronimo.apache.org/xml/ns/naming"
  configId="EntityDemoWebApp"
  parentId="CustomerEJB">

  <context-root>entity-ejb</context-root>

  <ejb-ref>
    <ref-name>ejb/CustomerHome</ref-name>
    <target-name>geronimo.server:EJBModule=CustomerEJB,J2EEApplication=null,J2EEServer=geronimo,
j2eeType=EntityBean,name=CustomerEJB</target-name>
  </ejb-ref>

</web-app>
```

Build the Loan CMP application by typing **maven** from the <cmp_home> directory. This will create the **entity-ejb-cmp.jar** and **entity-ejb.war** in the <cmp_home>/target directory.

[Back to Top](#)

Deploy the migrated application

To deploy the migrated Loan CMP application, make sure the Geronimo server is up and running.

From a command line, change directory to <geronimo_home> and type the following command:

```
java -jar bin/deployer.jar --user system --password manager deploy <cmp_home>/target/entity-ejb-cmp.jar <cmp_home>/dd/customer-ejb.xml
```

With this command you first tell the deployer tool where is the module to deploy, then you tell the deployer tool how to deploy the application by specifying the deployment plan.

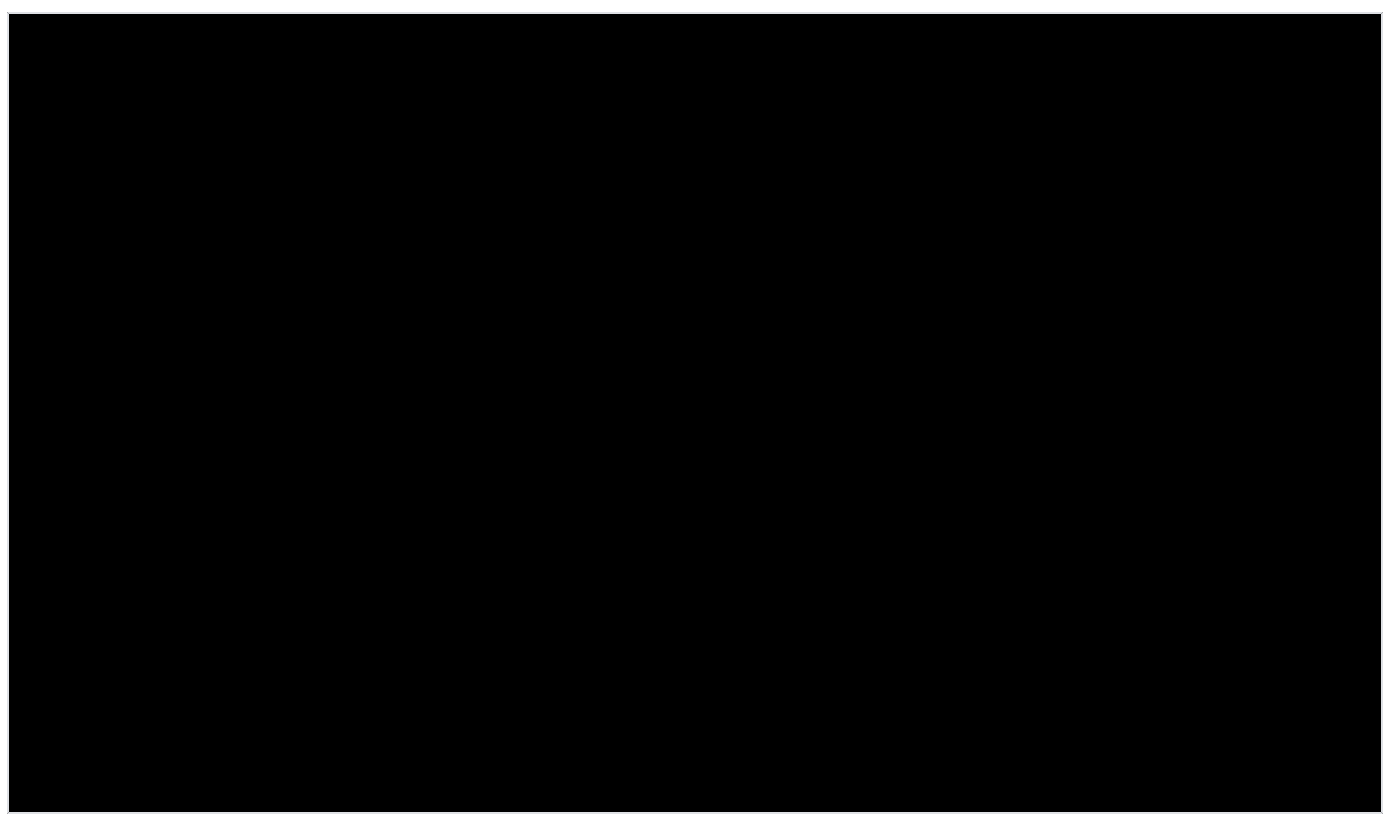
Deploy the Web Application by typing the following command:

```
java -jar bin/deployer.jar --user system --password manager deploy <cmp_home>/target/entity-ejb.war
```

From the command line change the the <cmp_home> directory and type the following command:

```
maven run:client
```

You should see something similar to the following example:



Test the applications the same way you tested on JBoss.

[Back to Top](#)

Summary

This article has shown you how to migrate a sample application, from JBoss to the Apache Geronimo application server. You followed 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 the Geronimo specific deployment descriptor the ejbreference name is mapped to the gbean name of the ejb unlike in the JBoss specific deployment descriptor where the resource name is mapped to the JNDI name of the ejb.
- In order to deploy a datasource in JBoss you need to just copy the configuration file to the deploy directory but in Geronimo you need to use the deployer tool or the Web console.

[Back to Top](#)