openejb-jar.xml

{scrollbar}

top

This article provides a great deal of information for users to get an understanding of Geronimo deployment plans for EJBs. This article covers the structure, overview and elements of EJB deployment plans. This topic is a broad area to discuss and this page should be updated constantly with the future enhancement of Geronimo. Though this document is limited in scope, it addresses the deployment plans of openejb-jar.xml in depth which will help the user to get an understanding of what each element does in the openejb-jar.xml. The article covers how applications can refer to different resources and handling such scenarios.

This article is organized into the following sections :

1.0 Basic steps to Deploy EJB in Geronimo

Typically, there are several steps that need to be taken in order to deploy EJBs in Geronimo :

1. First, it needs to create the EJB classes and interfaces.
2. Create the standard ejb-jar.xml deployment descriptor.
3. Create a Geronimo-specific openejb-jar.xml deployment plan.
4. Package the EJBs and deployment descriptors into an EJB JAR, or a directory tree laid out like an EJB JAR.

Use the deployment tool described in deployment plan "The Deploy Tool" to deploy the EJB JAR (or an EAR containing the EJB JAR) into the server or else use the Geronimo web console for it.

2.0 Describing the Geronimo deployment plan for EJB

Geronimo deployment plan for an EJB JAR is an XML document called "openejb-jar.xml". This document is defined by the openejb-jar-2.0.xsd schema and can be found in the <geronimo-home>/schema/ subdirectory of the main Geronimo installation directory. The deployment plan for an EJB JAR may be included in the EJB JAR, in which case it should be named META-INF/openejb-jar.xml or included in an EAR (but outside of the EJB JAR) and referenced by an alt-dd element of the EAR deployment plan or else saved as a separate file and provided to the deploy tool when the EJB JAR module is deployed (though this does not work when the EJB JAR is in an EAR).

The deployment plan should always use the OpenEJB namespace, and it typically requires elements from the Geronimo Naming, Geronimo Security, and Geronimo System namespaces. Additionally, it has a required attribute to identify its configuration name, and an optional attribute to select a parent configuration. A typical deployment for openejb-jar.xml can be presented as follows:

xmlopenejb-jarsolid <openejb-jar xmlns:openejb="http://www.openejb.org/xml/ns/openejb-jar-2.1" targetNamespace="http://www.openejb.org/xml/ns /openejb-jar-2.1" xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:security="http://geronimo.apache.org/xml/ns/security-1.1" xmlns: sys="http://geronimo.apache.org/xml/ns/deployment-1.1" xmlns:pkgen="http://www.openejb.org/xml/ns/pkgen-2.0" version="1.0"> ... </openejb-jar>

xmlns

The main namespace for the deployment plan, which should always be http://www.openejb.org/xml/ns/openejb-jar-2.1

xmlns:naming

A secondary namespace, used to identify the common elements for resolving EJB references, resource references, and Web services references. If any of those need to be resolved in the deployment plan, this attribute should be present, and should be set to http://geronimo.apache.org/xml/ns/naming-1.1

xmlns:security

A secondary namespace, used to identify the common elements for security configuration. If there are any security settings in the deployment plan, this attribute should be present, and should be set to http://geronimo.apache.org/xml/ns/security-1.1

xmlns:sys

A secondary namespace, used to identify the common elements for common libraries and module-scoped services. If there are any of those present in the deployment plan, this attribute should be present, and should be set to http://geronimo.apache.org/xml/ns/deployment-1.1

xmlns:pkgen

A secondary namespace, used to identify the common elements for configuring automatic primary key generation for CMP entity beans (such as, using a sequence or auto-increment column). If there are any primary key generators present in the deployment plan, this attribute should be present, and should be set to http://www.openejb.org/xml/ns/pkgen-2.0

xmlsolid <?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-1.1"> <dep:moduleId> <dep:groupId>default</dep:groupId> <dep:artifactId>BankEJB</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> <cmp-connection-factory> <resource-link>BankPool</resource-link> </cmp-connection-factory> <enterprise-beans> <session> <ejb-name>BankManagerFacadeBean</ejb-name> <jndi-name>org.apache.geronimo.samples.bank.ejb.BankManagerFacadeBean</jndi-name> <ejb-ref> <ref-name>ejb/Customer</ref-name> <ejb-link>Customer</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/Account</ref-name> <ejb-link>Account</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/ExchangeRate</ref-name> <ejb-link>ExchangeRate</ejb-link> </ejb-ref> </session> <entity> <ejb-name>Account</ejb-name> <local-jndi-name>AccountLocalEntity</local-jndi-name> <table-name>Account</table-name> <cmp-field-mapping> <cmp-field-name>accountNumber</cmp-field-name> <table-column>ACC_NO</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>accountType</cmp-field-name> <table-column>ACC_TYPE</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customer</cmp-field-name> <table-column>CUSTID_FK</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>balance</cmp-field-name> <table-column>BALANCE</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>Customer</ejb-name> <local-jndi-name>CustomerLocalEntity</local-jndi-name> <table-name>Customer</table-name> <cmp-field-mapping> <cmp-field-name>customerId</cmp-field-name> <table-column>CUST_ID</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customerName</cmp-field-name> <table-column>CUST_NAME</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>ExchangeRate</ejb-name> <local-jndi-name>ExchangeRate</local-jndi-name> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> </enterprise-beans> <relationships> <ejb-relation> <ejb-relation-name>Customer-Account</ejb-relation-name> <ejb-relationship-role> <ejb-relationship-role-name>Account-to-Customer</ejb-relationship-role-name> <relationship-role-source> <ejb-name>Account</ejb-name> </relationship-role-source> <cmr-field> <cmr-field-name>customer</cmr-field-name> </cmr-field> <foreign-key-column-on-source/> <role-mapping> <cmr-field-mapping> <key-column>CUST_ID</key-column> <foreign-key-column>CUSTID_FK</foreign-key-column> </cmr-field-mapping> </role-mapping> </ejb-relationship-role> </ejb-relation> </relationships> </openejb-jar>

2.1 Geronimo EJB Deployment Plan overview

In a EJB deployment plan, the main attributes defined in the top.

configIdrequiting

A unique name identifying this module. If deployed as a standalone EJB JAR, this name is used to identify the module to the deployment tool (to start, stop, undeploy, or redeploy the EJB JAR).

parentId

Identifies the parent configuration for this EJB JAR (the value specified here should match the configId for that module). If deployed as a standalone EJB JAR, this can be used to make the EJB JAR depend on another module such as another standalone EJB JAR or a J2EE Connector (or it should otherwise be omitted or set to the usual parent for J2EE modules, geronimo/j2ee-server/1.0/car).

inverseClassloading

If set to true, the class loader for the EJB JAR tries to load a class before checking whether the class is available from its parent class loader. configId
A unique name identifying this module. If deployed as a standalone EJB JAR, this name is used to identify the module to the deployment tool (to start, stop, undeploy, or redeploy the EJB JAR).

parentId

Identifies the parent configuration for this EJB JAR (the value specified here should match the configId for that module). If deployed as a standalone EJB JAR, this can be used to make the EJB JAR depend on another module such as another standalone EJB JAR or a J2EE Connector (or it should otherwise be omitted or set to the usual parent for J2EE modules, geronimo/j2ee-server/1.0/car). If deployed as part of an EAR this is usually not necessary, as EAR configuration will be the parent of this module.

inverseClassloading

If set to true, the class loader for the EJB JAR tries to load a class before checking whether the class is available from its parent class loader. If omitted or set to false, the normal (check parent first) class loader delegation behavior is used

2.1.1 Typical contents of Geronimo EJB Deployment Plan

There are some typical contents of the Geronimo EJB Deployment Plan that have been identified and are described below

1. Security settings indicating which users and roles should be able to access secure EJBs or secure EJB methods.
2. The details to resolving EJB references, resource references, and Web services references declared by EJBs in the ejb-jar.xml deployment descriptor. This isn't always necessary for EJB references (which may be resolved using an ejb-link in the ejb-jar.xml) but generally is for the other types of references.
3. JNDI names for each EJB, used by remote clients attempting to connect to the EJB.
4. Web Services settings, for session beans exposed as web services.
5. Database pool, table, column, query, and relationship information for CMP Entity Beans.
6. JMS mapping information for Message-Driven Beans.

For the simplest EJB JARs these settings may not be required or the defaults may be sufficient, but in most cases the Geronimo EJB deployment plan will need a substantial amount of information.
As usual in other Geronimo deployment plans open-ejb.jar has specific structure and element followed in an order.

2.1.2 Geronimo EJB Deployment Plan overview

The elements in the classloader-infoGroup are used to customize the EJB JAR's class. There are four elements that need to be discussed under class path settings.

import

Refers to another configuration deployed in the server. That configuration will be added as a parent of this one (a configuration may have more than one parent). The main effect is that the class loader for the EJB JAR will add the class loader for that configuration as a parent. Additionally, the parent configuration will be started before the EJB JAR.

dependency

Adds a third-party library to the class path for the EJB JAR. Any common libraries used in this manner should be located in a subdirectory of the repository/ directory of the main Geronimo installation.

hidden-classes

Lists packages or classes that may be in a parent class loader, but should not be exposed from there to the EJB JAR. This is typically used when the EJB JAR wants to use a different version of a library that one of its parent configurations (or Geronimo itself) uses.

non-overridable-classes

Lists packages or classes that the EJB JAR should always load from a parent class loader, and never load from its own class loader. This might be used to force an EJB to share the same instance of a common library with other applications or modules, even if they each include it in their own class path.

filter
Used to list classes or packages. The format is a comma-separated list of packages or fully-qualified class names (for example: javax.servlet,javax.ejb).

When considering the JMS and MDB Sample given(link the sample application) and it does not have any dependencies and imports.

xmlsolid <openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1" xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:security="http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:groupId>samples</sys:groupId> <sys:artifactId>OrderEjb</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> <sys:dependencies> <sys:dependency> <sys:groupId>geronimo</sys:groupId> <sys:artifactId>activemq-broker</sys:artifactId> <sys:version>1.1</sys:version> <sys:type>car</sys:type> </sys:dependency> <sys:dependency> <sys:groupId>samples</sys:groupId> <sys:artifactId>jms-resources</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>rar</sys:type> </sys:dependency> </sys:dependencies> <sys:hidden-classes/> <sys:non-overridable-classes/> </sys:environment>

Here the dependencies are defined here references the sample JMS resource since the application and activemq-broker which is in /repository/activemq-broker/1.1/rar

2.2 Working out deployment plans with samples

2.2.1 Deploying JAR containing Simple Helloworld EJB

This section will cover Geronimo v1.1 deployment plan for the simple helloworld EJB.
Here is the folder structure for the above simple application.Sample Application is available to download here
HelloWorld ejb Sample

java HelloWorld |_ejbModule | |_org.geronimo.ejbsample | | |_HelloBean | | |_HelloHome | | |_HelloObject | | | |_org.geronimo.ejbsample.client | |_HelloWorld |_META-INF | |_ejb-jar.xml | |_openejb-jar.xml | |_dist |_build |_build.xml

Here is the ejb-jar.xml for the above sample :

solidejb-jar.xml

Geronimo v1.1 deployment plan for the above sample

solidopenejb.jar

Steps to deploying the sample

There are two ways in which the sample can be deployed :

1. The sample can be built from scratch using the build.xml and editing the <geronimo_home> as your directory. Obtain the helloworld-ejbs.jar from the **dist** directory and use the Geronimo web console. Browse for the helloworld-ejbs.jar file and click on the **Install** button.

2.Use the following command line code to deploy the application.

Running the Client to test the HelloWorld EJB service

org.geronimo.ejbsample.client.HelloWorld.java is the client code for the above application. Set the necessary class path to run client in your environment.

You will see the "Hello world" print on your command line

2.2.2 Working with different EJB types

Common settings for CMP Entity beans

There are few settings for CMP Entity beans in a Geronimo EJB deployment plan and following are element we consider to set values.

cmp-connection-factory

This is specifies a JDBC connection pool that should be used by CMP entity beans to connect to the database. There are several styles available to refer to the connection pool.The cmp-connection-factory element points to a database pool using the same syntax a resource reference uses. There are several ways to identify the database pool. One is to specify the component by a simple name (resource-link), while the other is to use a more complex ObjectName (target-name, or the individual components in the objectNameGroup). The resource-link handles most common resource situations (JDBC pools deployed as J2EE connectors in the same application or deployed standalone in the same server) while the target-name or objectNameGroup can be used for any

<cmp-connection-factory> <resource-link>BankPool</resource-link> </cmp-connection-factory>

ejb-ql-compiler-factory

The name of a class that knows how to translate EJB-QL queries into SQL statements for a particular database product. This must be the fully-qualified class name of a class that implements org.tranql.sql.EJBQLCompilerFactory. The default is for the Derby database, though this may work for other database products too.

db-syntax-factory

The name of a class that knows how to customize CMP SQL statements for a particular database product. This must be the fully-qualified class name of a class that implements org.tranql.sql.DBSyntaxFactory. The default is for the Derby database, though this may work for other database products too.

enforce-foreign-key-constraints

This is effectively a true/false element – if it's present that means true, and if it's not present, that means false. If true, then Geronimo will make a special effort to execute insert, update, and delete statements in an order consistent with the foreign keys between tables. If false, then Geronimo will execute statements in any order, though still within the same transaction. This element should be present if the underlying database enforces foreign keys at the moment a statement is executed instead of at the end of the transaction.

Common Settings for EJB Session beans

The following elements mainly need to be set up :

ejb-name

Identifies the EJB that these settings apply to. This should match the ejb-name for the EJB in ejb-jar.xml.

jndi-name

The Home interface for the EJB is registered in JNDI at the address specified here. This global JNDI name is used by application clients to connect to this EJB, as well as by CORBA clients if this EJB is exposed via CORBA. It is only meaningful if the EJB has a (remote) Home interface.

local-jndi-name

The LocalHome interface for the EJB is registered in JNDI at the address specified here. It is only meaningful if the EJB has a LocalHome interface. Note that the official Geronimo distributions do not allow access to local EJBs via JNDI (making this setting irrelevant), but a custom Geronimo build may.

tssGroup

This is a set of elements that contains CORBA security settings, for EJBs exposed as CORBA objects. It is not necessary if the EJB will not be accessed via CORBA.

jndiEnvironmentRefsGroup

A set of elements that handle resolving references declared by the current Session bean.For an example EJB references, Resource references, and Web Service references can be handled.

The following openejb-jar.xml has taken from the http://cwiki.apache.org/GMOxDOC11/ejb-sample-application.html
This example is using a EJB session bean and how it has implemented inside the openejb-jar.xml

xmlopenejb-jar.xmlsolid <?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-1.1"> <dep:moduleId> <dep:groupId>default</dep:groupId> <dep:artifactId>BankEJB</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> <cmp-connection-factory> <resource-link>BankPool</resource-link> </cmp-connection-factory> <enterprise-beans> <session> <ejb-name>BankManagerFacadeBean</ejb-name> <jndi-name>org.apache.geronimo.samples.bank.ejb.BankManagerFacadeBean</jndi-name> <ejb-ref> <ref-name>ejb/Customer</ref-name> <ejb-link>Customer</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/Account</ref-name> <ejb-link>Account</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/ExchangeRate</ref-name> <ejb-link>ExchangeRate</ejb-link> </ejb-

ref> </session> <entity> <ejb-name>Account</ejb-name> <local-jndi-name>AccountLocalEntity</local-jndi-name> <table-name>Account</table-name> <cmp-field-mapping> <cmp-field-name>accountNumber</cmp-field-name> <table-column>ACC_NO</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>accountType</cmp-field-name> <table-column>ACC_TYPE</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customer</cmp-field-name> <table-column>CUSTID_FK</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>balance</cmp-field-name> <table-column>BALANCE</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>Customer</ejb-name> <local-jndi-name>CustomerLocalEntity</local-jndi-name> <table-name>Customer</table-name> <cmp-field-mapping> <cmp-field-name>customerId</cmp-field-name> <table-column>CUST_ID</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customerName</cmp-field-name> <table-column>CUST_NAME</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>ExchangeRate</ejb-name> <local-jndi-name>ExchangeRate</local-jndi-name> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> </enterprise-beans> <relationships> <ejb-relation> <ejb-relation-name>Customer-Account</ejb-relation-name> <ejb-relationship-role> <ejb-relationship-role-name>Account-to-Customer</ejb-relationship-role-name> <relationship-role-source> <ejb-name>Account</ejb-name> </relationship-role-source> <cmr-field> <cmr-field-name>customer</cmr-field-name> </cmr-field> <foreign-key-column-on-source/> <role-mapping> <cmr-field-mapping> <key-column>CUST_ID</key-column> <foreign-key-column>CUSTID_FK</foreign-key-column> </cmr-field-mapping> </role-mapping> </ejb-relationship-role> </ejb-relation> </relationships> </openejb-jar>

The following piece of code has excerpt from above openejb-jar.xml and shows how the session bean is defined. BankManagerFacadeBean is Stateless Session Bean which is acting as a service class for different application clients.It's **jndi-name** attribute is assigned to **org.apache.geronimo.samples.bank.ejb.BankManagerFacadeBean** Then the **<ejb-ref>** tag is used to refer the other dependent ejbs.In this case **<ejb-ref>** tag has two sub elements which are **<ref-name>** and the **<ejb-name>**

<session> <ejb-name>BankManagerFacadeBean</ejb-name> <jndi-name>org.apache.geronimo.samples.bank.ejb.BankManagerFacadeBean</jndi-name> <ejb-ref> <ref-name>ejb/Customer</ref-name> <ejb-link>Customer</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/Account</ref-name> <ejb-link>Account</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/ExchangeRate</ref-name> <ejb-link>ExchangeRate</ejb-link> </ejb-ref> </session>

Common Settings for EJB Entity beans

Entity Beans settings can include JNDI names used by remote clients,CMP settings, and haven't tested COBRA configurations and resolving references. There are set of elements to be configured in the openejb-jar.xml.The following code piece is excerpt from the Sample Application and which shows how entity bean has been implemented in the deployment plans.

ejb-name

This ejb-name element identifies the EJB that these settings apply to. This should match the ejb-name for the EJB in ejb-jar.xml

jndi-name

The Home interface for the EJB is registered in JNDI at the address specified here. This global JNDI name is used by application clients to connect to this EJB.It is only meaningful if the EJB has a (remote) Home interface.

local-jndi-name

The LocalHome interface for the EJB is registered in JNDI at the address specified here. It is only meaningful if the EJB has a LocalHome interface. Note that the official Geronimo distributions do not allow access to local EJBs via JNDI (making this setting irrelevant), but a custom Geronimo build may.

tssGroup

This is a set of elements that contains CORBA security settings, for EJBs exposed as CORBA objects. (It has not Tested this elements in practically)

jndiEnvironmentRefsGroup

This is a set of elements that handle resolving references declared by the current Session bean (including EJB references, Resource references, and Web Service references)and this group is common to all EJB types.

... <entity> <ejb-name>ExchangeRate</ejb-name> <local-jndi-name>ExchangeRate</local-jndi-name> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> ....

Considering the above piece of code under the **<entity>** tag **<ejb name>** is assigned for the relevant EJB entity bean.Since this **ExchangeRate** has a LocalHome interface **<local-jndi-name>** the EJB is registered in JNDI at the address specified as **ExchangeRate** .Under the **<resource-ref>** tags

Common Settings for EJB Message Beans

Message-driven bean settings include mapping the MDB to a specific message destination (usually a JMS Topic or Queue), as well as resolving references to other EJBs, resources, or Web services.The Geronimo user guide provides a sample applicationLink the sample which elaborate how to deploy a MDB in Geronimo.There are common settings to be done in the deployment plan in order to achieve this.

ejb-name

Identifies the EJB that these settings apply to. This should match the ejb-name for the EJB in ejb-jar.xml.

resource-adapter

The resource-adapter element identifies the resource adapter that this message-driven bean connects to. This is typically a JMS server, meaning ActiveMQ for the default Geronimo JMS provider.It identifies the resource adapter instance that this MDB should use to connect to its destination. For example, a specific ActiveMQ broker may have several resource adapter instances set up, with different authentication settings, and this identifies the specific instance to use.
There are several ways to identify the resource adapter. One is to specify the component by a simple name (resource-link), while the other is to use a more complex GBean Name (target-name, or the individual components in the objectNameGroup). The resource-link handles most common resource situations (a JMS resource adapter deployed as part of the same EAR or in the top level of the server) while the target-name or objectNameGroup can be used for any resource. This might be important if, for example, two resource adapter deployments use the same name, so the resource-link does not uniquely identify one and it must be fully-qualified.This can be used to identify any resource adapter in the same EAR or at the top level in the server. The value specified here should match the resourceadapter-name specified for the resource adapter instance in its Geronimo deployment plan.

target-name

A way to specify any resource adapter running in the server. This should be a GBean Name identifying the resource, such as geronimo.server: J2EEServer=geronimo,J2EEApplication=null, JCAResource=MyJMS,j2eeType=JCAResourceAdapter,name=JMS_RA

activation-config

Holds any configuration data (in the form of name/value pairs) required by the resource adapter in order to supply messages to the MDB.

jndiEnvironmentRefsGroup
A set of elements that handle resolving references declared for the current message-driven bean (including EJB references, Resource references, and Web Service references).

The sample application given in the user guide (link it to the JMS and MDB sample) demonstrates how to implement the MDB in geronimo.Basically the Business overview of this application is the following. Order processing application has two defined message queues to receive orders and consignments. Order requests can be generated and sent via the company's web application. When order requests are received to the order queue, a MDB will be triggered. It will carry out the next level of order request processing by saving those requests in to a server repository. Those saved order requests will be processed by a company employee later.

In order to deploy a JMS/MDB with Geronimo it's required to create a JMS resource plan.The purpose of creating the JMS plan is that from the geronimo server end you will be accessing JMS/MDB resource using this plan which will define all the definitions for the connection factories and the Queues/Topics

xmlsolidJMS_resource_plan.xml <?xml version="1.0" encoding="UTF-8"?> <connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleId> <dep:groupId>samples</dep:groupId> <dep:artifactId>jms-resources</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>rar</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>geronimo</dep:groupId> <dep:artifactId>activemq-broker</dep:artifactId> <dep:type>car</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> <resourceadapter> <resourceadapter-instance> <resourceadapter-name>CommonConnectionFactory</resourceadapter-name> <config-property-setting name="Password">geronimo</config-property-setting> <config-property-setting name="UserName">geronimo</config-property-setting> <nam:workmanager xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1"> <nam:gbean-link>DefaultWorkManager</nam:gbean-link> </nam:workmanager> </resourceadapter-instance> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>javax.jms.QueueConnectionFactory</connectionfactory-interface> <connectiondefinition-instance> <name>CommonConnectionFactory</name> <connectionmanager> <xa-transaction> <transaction-caching/> </xa-transaction> <single-pool> <match-one/> </single-pool> </connectionmanager> </connectiondefinition-instance> </connection-definition> </outbound-resourceadapter> </resourceadapter> <adminobject> <adminobject-interface>javax.jms.Queue</adminobject-interface> <adminobject-class>org.activemq.message.ActiveMQQueue</adminobject-class> <adminobject-instance> <message-destination-name>OrderQueue</message-destination-name> <config-property-setting name="PhysicalName">OrderQueue</config-property-setting> </adminobject-instance> <adminobject-instance> <message-destination-name>ConsignmentQueue</message-destination-name> <config-property-setting name="PhysicalName">ConsignmentQueue</config-property-setting> </adminobject-instance> </adminobject> <adminobject> <adminobject-interface>javax.jms.Topic</adminobject-interface> <adminobject-class>org.activemq.message.ActiveMQTopic</adminobject-class> </adminobject> </connector>

Geronimo web console provides the needful to create a JMS resource plan structure and it's required to fill your bunisess logic to map to fill your requirements.Once you log in to the web console in your left hand side navigation you will find a JMS resource link under services.In this application there is a MDB that will listen on OrderQueue. openejb-jar.xml defines Geronimo specific JMS resource services available of that MDB. It links OrderRecv MDB with OrderQueue via CommonConnectionFactory.This Article will be discussing further details of the relationship of the JMS resource plan and the openejb-jar.xml in Resolving references session(link this to the topic)

xmlsolidopenejb-jar.xml <openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1" xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:security="http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:groupId>samples</sys:groupId> <sys:artifactId>OrderEjb</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> <sys:dependencies> <sys:dependency> <sys:groupId>geronimo</sys:groupId> <sys:artifactId>activemq-broker</sys:artifactId> <sys:type>car</sys:type> </sys:dependency> <sys:dependency> <sys:groupId>samples</sys:groupId> <sys:artifactId>jms-resources</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>rar</sys:type> </sys:dependency> </sys:dependencies> <sys:hidden-classes/> <sys:non-overridable-classes/> </sys:environment> <enterprise-beans> <message-driven> <ejb-name>OrderRecvMDB</ejb-name> <resource-adapter> <resource-link>CommonConnectionFactory</resource-link> </resource-adapter> <activation-config> <activation-config-property> <activation-config-property-name>destination</activation-config-property-name> <activation-config-property-value>OrderQueue</activation-config-property-value> </activation-config-property> <activation-config-property> <activation-config-property-name>destinationType</activation-config-property-name> <activation-config-property-value>javax.jms.Queue</activation-config-property-value> </activation-config-property> </activation-config> </message-driven> </enterprise-beans> </openejb-jar>

According to openejb-jar.xml, it shows how the Geronimo resources are set as dependencies in order to access in the EJB applications.Basically "activemq-broker" and "jms-resources" are the two dependencies comes with Geronimo server to use in MDB and JMS applications.

<sys:dependencies> <sys:dependency> <sys:groupId>geronimo</sys:groupId> <sys:artifactId>activemq-broker</sys:artifactId> <sys:type>car</sys:type> </sys:dependency> <sys:dependency> <sys:groupId>samples</sys:groupId> <sys:artifactId>jms-resources</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>rar</sys:type> </sys:dependency> </sys:dependencies>

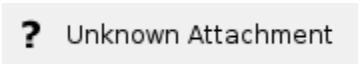
2.3 Resolving References from EJB

EJBs can declare references to other EJBs either locally or as an interfaces remotely.These resources can be various types (database pools or JMS connection factories) to message destinations, and to web services. As in most cases, the references are declared in the ejb-jar.xml deployment descriptor, and those references are resolved to specific targets in the Geronimo openejb-jar.xml deployment plan.

This article will be discussing how the various types of references are resolved in EJB Applications with Geronimo server. These are applicable to all EJB types (Session, Entity, and Message-Driven) All of EJB types can declare references of any of these types.

Common Resource Mapping Elements

As it's mentioned earlier all the EJB types can declare references of any type using common resource Mapping elements.Therefore these elements have defined as common.Following diagram shows how these elements are organized in the xml graphical representation.



domain

The domain name portion of the GBean Name identifying the EJB or resource to map to. This should usually be geronimo.server

server

The J2EE Server portion of the GBean Name identifying the EJB or resource to map to. This should usually be geronimo, and if not specified, it defaults to the name of the JSR-77 J2EEServer that was configured for the deployer GBean that's deploying the current module
application

The name of the application that the target EJB or resource is defined by the **application**. If the target module was deployed standalone, this would be null. If it was deployed as part of an EAR, this should be the application-name specified for the EAR, or the moduleId if no application-name was specified.

module

The modulesID of the module that the target EJB or resource is in.

type

Should identify the type of the EJB or resource that this is pointing to. For EJBs, this should be one of StatefulSessionBean, StatelessSessionBean, EntityBean, or MessageDrivenBean, depending on the type of the target EJB.

name
The JSR-77 name of the target object. This would be the ejb-name for an EJB

EJB References

EJB can be referred locally and remotely as we discussed in the previous sections. EJB reference points to the remote interface of an EJB,usually deployed elsewhere in the local Geronimo server.It can also refer to EJBs running ellse where,using CORBA to connect to the remote server.But this has not been discussed in this article .This section will brief how to resolve references with EJB

When EJB references are configured, the client EJB looks in its local java:comp/env/... JNDI space to access the referenced EJB.

Basically there are four ways of resolving EJB.The simplest is the ejb-link, which is just the name of an EJB elsewhere in the same application, or deployed in a standalone EJB JAR in Geronimo. The target-name or **objectNameGroup** can be used to specify the EJB by its full GBean Name, either as a single String or as separate GBean Name components. Finally, the **corbaNameGroup** is used to configure references to EJBs via CORBA.This section has not been experimented yet and this article won't give a proper introduction for it.

ejb-ref

Groups all the settings for a reference to another EJB via remote interface. </para>

ref-name
Each reference must have a ref-name,which is used to match the definition from the openejb-jar.xml to the EJB reference declared in ejb-jar.xml The value in the openejb-jar.xml must match the ejb-ref-name in the ejb-ref in the ejb-jar.xml

objectNameGroup

This has been described previously in section 2.1 When type should be one of StatefulSessionBean, StatelessSessionBean, EntityBean or MessageDrivenBean, depending on the type of he target EJB.

ejb-link

An ejb-link can be specified in openejb-jar.xml to identify an EJB in the same application EAR by name.It is a must to match the ejb-name for the EJB in its ejb-jar.xml. A value specified openeb-jar.xml for this element which overrides any ejb-link specified for the same EJB reference in ejb-jar.xml.

target-name

objectNameGroup is basically spliting out all the components But this element holds a single GBean Name containing all that information required to identify the EJB. This may be used to map an EJB in a different application. It would typically look like geronimo.server:J2EEApplication=ear-name, EJBModule=ejb-jar-name.jar, J2EEServer=geronimo,j2eeType=StatelessSessionBean,name=EJBName
Sample Code

When considering a typical admin object(such as a JMS topic or queue), the type should be JCAAdminObject and the name should match the message-destination-name for the admin object in the geronimo deployment plan for the JMS resource adapter.

Local EJB Reference

A local EJB reference points to the local interface of an EJB which is deployed elsewhere in the local Geronimo server. When local EJB references are configured in the geronimo deployment plan(openejb-jar.xml)the client EJB looks in its local java:comp/env/... JNDI space to access the referenced EJB.It 's required to configure few elements in the openejb-jar.xml and following are the elements.This has been discussed in the previous topic too and somebody might think that the information has been repeated.But it's worth to identify the different use of elements in Local EJB reference and Remote EJB reference.

ejb-local-ref

Groups all the settings for a reference to another EJB via its local interface
ref-name

Each reference must have a ref-name, which is used to match the definition here to the EJB reference declared in ejb-jar.xml. The value here must match the ejb-ref-name in the ejb-local-ref in ejb-jar.xml.

objectNameGroup

Refers to the EJB deployed in Geronimo using the syntax described in Section 2.1, "Common Resource Mapping Elements". The type should be one of StatefulSessionBean, StatelessSessionBean, EntityBean, or MessageDrivenBean, depending on the type of the target EJB.

ejb-link

An ejb-link can be specified here to identify an EJB in the same application EAR by name (must match the ejb-name for the EJB in its ejb-jar.xml). A value specified here overrides any ejb-link specified for the same EJB reference in ejb-jar.xml.

target-name
Instead of splitting out all the components using the objectNameGroup, this element holds a single GBean Name containing all that information required to identify the EJB. This may be used to map an EJB in a different application. It would typically look like: geronimo.server:J2EEApplication=ear-name, EJBModule=ejb-jar-name.jar, J2EEServer=geronimo,j2eeType=StatelessSessionBean,name=EJBName

Geronimo user guide has already given you a great insight about playing with EJB with geronimo in the sections of **Sample Applications** and **JBOSS to geronimo migration**.There plenty of samples which demonstrate how to refer a EJB Locally or Remotely.

One of the simplest sample is for referring EJB Locally is "very simple EJB session bean sample" in ("http://cwiki.apache.org/GMOxDOC11/very-simple-session-ejb-example.html)

The Scenario of the Sample is a Web Client(JSP) which locate and access a EJB by referring it locally.Simply it has used home interface to locate the EJB (service) and the remote interface has used to invoke methods on the EJB. The remote interface is just like a ordinary Remote Method Invocation interface (RMI). This EJB service is used only by the JSP web client which is run in the same Geronimo server (same JVM). Let's discuss how the EJB locally in deployment plans

xmlsolidweb.xml <?xml version="1.0" encoding="UTF-8"?> <web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"> <display-name> MyTImeWeb</display-name> <welcome-file-list> <welcome-file>index.jsp</welcome-file> </welcome-file-list> <!-- To refer local EJB's --> <ejb-local-ref> <ejb-ref-name>ejb/MyTimeBean</ejb-ref-name> <ejb-ref-type>Session</ejb-ref-type> <local-home>mytimepak.MyTimeLocalHome</local-home> <local>mytimepak.MyTimeLocal</local> <ejb-link>MyTimeBean</ejb-link> </ejb-local-ref> </web-app>

As above shown in the web.xml file **<ejb-local-ref>** tag has used to define and set the values for locally referring EJB interfaces **MyTimeLocalHome** and **MyTimeLocal**.As you can see in the following **geronimo-web.xml** has not defined any entry for it that means for there is not server specific entry need to be define for a EJB reference.

xmlsolidgeronimo-web.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1" xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:sec="http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:groupId>default</sys:groupId> <sys:artifactId>MyTimeWeb</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> </sys:environment> <context-root>/mytime</context-root> </web-app> xmlsolidejb-jar.xml <?xml version="1.0" encoding="UTF-8"?> <ejb-jar id="ejb-jar_1" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd" version="2.1"> <description>Example of a session bean</description> <display-name>MyTimeBeanEJBName</display-name> <enterprise-beans> <session id="Session_MyTime"> <description>An EJB named MyTimeBean</description> <display-name>MyTimeBeanName</display-name> <ejb-name>MyTimeBean</ejb-name> <local-home>mytimepak.MyTimeLocalHome</local-home> <local>mytimepak.MyTimeLocal</local> <ejb-class>mytimepak.MyTimeBean</ejb-class> <session-type>Stateless</session-type> <transaction-type>Container</transaction-type> </session> </enterprise-beans> </ejb-jar>

In the ejb-jar.xml shows the normal deployment descriptor entries as usual for any server deployment.

xmlopenejb-jar.xmlsolid <?xml version="1.0" encoding="UTF-8"?> <openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1" xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:pkgen="http://www.openejb.org/xml/ns/pkgen-2.0" xmlns:sec="http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:groupId>default</sys:groupId> <sys:artifactId>TimeBean_artifact_in_openejb</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> </sys:environment> <enterprise-beans> <session> <ejb-name>MyTimeBean</ejb-name> <ejb-ref> <ref-name>ejb/MyTimeBean</ref-name> <ejb-link>MyTimeBean</ejb-link> </ejb-ref> </session> </enterprise-beans> </openejb-jar> <?xml version="1.0" encoding="UTF-8"?> <openejb-jar xmlns="

http://www.openejb.org/xml/ns/openejb-jar-2.1" xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:pkgen="http://www.openejb.org/xml/ns/pkgen-2.0" xmlns:sec="http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:groupId>default</sys:groupId> <sys:artifactId>TimeBean_artifact_in_openejb</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> </sys:environment> <enterprise-beans> <session> <ejb-name>MyTimeBean</ejb-name> <ejb-ref> <ref-name>ejb/MyTimeBean</ref-name> <ejb-link>MyTimeBean</ejb-link> </ejb-ref> </session> </enterprise-beans> </openejb-jar>

As you already know openejb-jar.xml is the Geronimo specific deployment plan to deploy an EJB on Geronimo server.The following peice of code has been excerpt from the above to discuss further.The session bean is defined by giving the information for the client access.Note that **ref-name=ejb/MyTimeBean** is the same that clients use to lookup for the EJB. **ejb-link=MyTimeBean** in the openejb-jar.xml is the same for the **ejb-name=MyTimeBean** in ejb-jar.xml. The ejb-link and the ejb-name must be the same and then only a client will locate the EJB.

xmlsolid <session> <ejb-name>MyTimeBean</ejb-name> <ejb-ref> <ref-name>ejb/MyTimeBean</ref-name> <ejb-link>MyTimeBean</ejb-link> </ejb-ref> </session>

Resource References

In EJB applications require to refer many external resources typically JDBC resources,JMS resources or URLS.The geronimo deployment plan for EJB (openejb-jar.xml) handles such situations using the **resource-ref** .This element is used to map resource references to specific resources available in the geronimo server itself. The elements available for mapping resource references are defined as follows and it has used to some excerpt codes from the sample application to demonstrate these elements.Therefore consider the following openejb-jar.xml in the http://cwiki.apache.org/GMOxDOC11/ejb-sample-application.html

xmlsolid <?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-1.1"> <dep:moduleId> <dep:groupId>default</dep:groupId> <dep:artifactId>BankEJB</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> <cmp-connection-factory> <resource-link>BankPool</resource-link> </cmp-connection-factory> <enterprise-beans> <session> <ejb-name>BankManagerFacadeBean</ejb-name> <jndi-name>org.apache.geronimo.samples.bank.ejb.BankManagerFacadeBean</jndi-name> <ejb-ref> <ref-name>ejb/Customer</ref-name> <ejb-link>Customer</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/Account</ref-name> <ejb-link>Account</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/ExchangeRate</ref-name> <ejb-link>ExchangeRate</ejb-link> </ejb-ref> </session> <entity> <ejb-name>Account</ejb-name> <local-jndi-name>AccountLocalEntity</local-jndi-name> <table-name>Account</table-name> <cmp-field-mapping> <cmp-field-name>accountNumber</cmp-field-name> <table-column>ACC_NO</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>accountType</cmp-field-name> <table-column>ACC_TYPE</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customer</cmp-field-name> <table-column>CUSTID_FK</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>balance</cmp-field-name> <table-column>BALANCE</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>Customer</ejb-name> <local-jndi-name>CustomerLocalEntity</local-jndi-name> <table-name>Customer</table-name> <cmp-field-mapping> <cmp-field-name>customerId</cmp-field-name> <table-column>CUST_ID</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customerName</cmp-field-name> <table-column>CUST_NAME</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>ExchangeRate</ejb-name> <local-jndi-name>ExchangeRate</local-jndi-name> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> </enterprise-beans> <relationships> <ejb-relation> <ejb-relation-name>Customer-Account</ejb-relation-name> <ejb-relationship-role> <ejb-relationship-role-name>Account-to-Customer</ejb-relationship-role-name> <relationship-role-source> <ejb-name>Account</ejb-name> </relationship-role-source> <cmr-field> <cmr-field-name>customer</cmr-field-name> </cmr-field> <foreign-key-column-on-source/> <role-mapping> <cmr-field-mapping> <key-column>CUST_ID</key-column> <foreign-key-column>CUSTID_FK</foreign-key-column> </cmr-field-mapping> </role-mapping> </ejb-relationship-role> </ejb-relation> </relationships> </openejb-jar>

ref-name

Each resource reference must have a ref-name, which is used to match the definition in the openejb-jar.xml to the resource reference in ejb-jar.xml. Specifically, the value here must match the res-ref-name in the resource-ref in ejb-jar.xml.

xmlsolidExcerpt from openejb-jar.xml <entity> .... <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity>

objectNameGroup

This element can refer to the resource deployed in Geronimo using the syntax described in "Common Resource Mapping Elements". For a typical connection factory (such as a JDBC pool or JMS connection factory), the type should be JCAManagedConnectionFactory and the name should match the value of the connectiondefinition-instance/name element in the Geronimo deployment plan for the resource adapter (or the name selected for the resource when deploying it through the console). Note: other resource types may use different types and names.

resource-link

This element can be used to identify any resource deployed as a J2EE connector (including JDBC pools and JMS connection factories). The value specified here should match the connectiondefinition-instance/name element in the Geronimo deployment plan for the connector.

xmlsolidExcerpt from openejb-jar.xml <entity> <ejb-name>ExchangeRate</ejb-name> <local-jndi-name>ExchangeRate</local-jndi-name> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity>

This BankPool is an XML file which is the Geronimo data base plan created using geronimo(Debery has integrated with Geronimo to use as a Data Base resource) data base manager. Therefore, this has been deployed in the Geronimo server itself and **resource-link** tag will provide the access of it to the application.

target-name

A way to specify any type of resource running in the server. This should be a GBean Name identifying the resource, such as geronimo.server:J2EEServer=geronimo,J2EEApplication=null, JCAModule=my-db-pool,j2eeType=JCAManagedConnectionFactory,name=MyDatasource

url

If the resource type was java.net.URL, this element can be used to provide a value for the URL reference (such as http://www.amazon.com/). This element should not be used for other resource types.

Therefore, as discussed above there are several ways to identify a targeted resource. One is to specify the component by a simple name (resource-link), while the other is to use a more complex GBean Name (target-name, or the individual components in the objectNameGroup and there is not sample provided for this). The resource-link handles most common resource situations (J2EE connectors in the same application or deployed standalone, including JDBC pools and JMS connection factories) while the target-name or objectNameGroup can be used for any resource. This might be important if, for example, two resource adapter deployments use the same name for their connection factory, so the resource-link does not uniquely identify one and it must be fully-qualified. Finally, in the special case of URL resources, the url element provides a value for the resource (instead of looking something else up in the server environment).

Resolving JMS Resources

This section will be covered how to resolve JBM references from EJB in geronimo.It's used the JMS sample application in the Uder Guide to demonstrate the deployment plans configurations in order to use JMS in applications(http://cwiki.apache.org/GMOxDOC11/jms-and-mdb-sample-application.html) JMS resource can be a topic or queue which need to be configured specifically in the ejb-jar.xml file, as shown below. In this example it has been generated using Xdoclet at the build time.

xmlsolid <?xml version="1.0" encoding="UTF-8"?> <ejb-jar xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd" version="2.1"> ..... <enterprise-beans> <!-- Message Driven Beans --> <message-driven > <description> <![CDATA[Order Recieve Message Driven Bean]]> </description> <display-name>OrderRecvMDB</display-name> <ejb-name>OrderRecvMDB</ejb-name> <ejb-class>org.apache.geronimo.samples.order.ejb.OrderRecvMDB</ejb-class> <messaging-type>javax.jms.MessageListener</messaging-type> <transaction-type>Container</transaction-type> <message-destination-type>javax.jms.Queue</message-destination-type> <activation-config> <activation-config-property> <activation-config-property-name>destinationType</activation-config-property-name> <activation-config-property-value>javax.jms.Queue</activation-config-property-value> </activation-config-property> </activation-config> </message-driven> </enterprise-beans> <assembly-descriptor > </assembly-descriptor> </ejb-jar>

This sample application refers a JMS resource from the geronimo server and as you can see in the above ejb-jar.xml,following entried are used to configure it in ejb-jar.xml.This ejb-jar.xml was generated uisng xdoclet.And seems it has been limited to gernerate the primary needs of configuring JMS entries in the ejb-jar.xml

messaging-type=javax.jms.MessageListener
transaction-type=Container
message-destination-type=javax.jms.Queue

Message Listener allows a listener to be notified when a message arrives. Contrary to the pull-style of channels, some building blocks (e.g., PullPushAdapter) provide an event-like, push-style message delivery model. In this case, the entity to be notified of message reception needs to provide a callback to be invoked whenever a message has been received. The MessageListener interface provides a method to do so

xmlsolidopenejb-jar.xml <openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1" xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:security="http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:groupId>samples</sys:groupId> <sys:artifactId>OrderEjb</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>car</sys:type> </sys:moduleId> <sys:dependencies> <sys:dependency> <sys:groupId>geronimo</sys:groupId> <sys:artifactId>activemq-broker</sys:artifactId> <sys:type>car</sys:type> </sys:dependency> <sys:dependency> <sys:groupId>samples</sys:groupId> <sys:artifactId>jms-resources</sys:artifactId> <sys:version>1.0</sys:version> <sys:type>rar</sys:type> </sys:dependency> </sys:dependencies> <sys:hidden-classes/> <sys:non-overridable-classes/> </sys:environment> <enterprise-beans> <message-driven> <ejb-name>OrderRecvMDB</ejb-name> <resource-adapter> <resource-link>CommonConnectionFactory</resource-link> </resource-adapter> <activation-config> <activation-config-property> <activation-config-property-name>destination</activation-config-property-name> <activation-config-property-value>OrderQueue</activation-config-property-value> </activation-config-property> <activation-config-property> <activation-config-property-name>destinationType</activation-config-property-name> <activation-config-property-value>javax.jms.Queue</activation-config-property-value> </activation-config-property> </activation-config> </message-driven> </enterprise-beans> </openejb-jar>

Now the turn to see how this JMS resource has been configured on Geronimo specific deployment plan(openejb-jar.xml).Note that the segment of **dependencies** and **message-driven** has done it.First considering the setting for dependencies segment of the openejb-jar.xml,the JMS resources of the geronimo server and the deployed JMS resource plan have refered in it.**artifactId=activemq-broker** is the entry to make available(to the application)the JMS resource in the geronimo server itself.The second dependency setting is **artifactId=jms-resources** and **jms-resources** is the name of the JMS plan which refered by the application.This is a XML file which can be generated using Geronimo web console and should be already deployed in the server.

Basically Message driven bean is the EJB type which has the use of JMS.Therefore name of the EJB(**OrderRecvMDB**) has given to use the JMS resource.

message -destination

This element declares the Topic and the message-destination-ref in a particular EJB declares a reference to it and places the destination in to

message-destination-type

Declares the type of Message resource which will be refered by the application.As it's already discussed there are two types of message resources. Those are topic and Queue.In this case it's Queue.

For reference, here's a snippet from a JMS connector deployment plan that matches the EJB JAR configuration above and this can be generated using the geronimo web console.One can identify the <moduleId> ,<dependency> and <adminobject> are mapped the ejb-jar.xml.

xmlsolidjms-resource-plan.xml <?xml version="1.0" encoding="UTF-8"?> <connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleId> <dep:groupId>samples</dep:groupId> <dep:artifactId>jms-resources</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>rar</dep:type> </dep:moduleId> <dep:dependencies> <dep:dependency> <dep:groupId>geronimo</dep:groupId> <dep:artifactId>activemq-broker</dep:artifactId> <dep:type>car</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> <resourceadapter> <resourceadapter-instance> <resourceadapter-name>CommonConnectionFactory</resourceadapter-name> <config-property-setting name="Password">geronimo</config-property-setting> <config-property-setting name="UserName">geronimo</config-property-setting> <nam:workmanager xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1"> <nam:gbean-link>DefaultWorkManager</nam:gbean-link> <

/nam:workmanager> </resourceadapter-instance> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>javax.jms.QueueConnectionFactory</connectionfactory-interface> <connectiondefinition-instance> <name>CommonConnectionFactory</name> <connectionmanager> <xa-transaction> <transaction-caching/> </xa-transaction> <single-pool> <match-one/> </single-pool> </connectionmanager> </connectiondefinition-instance> </connection-definition> </outbound-resourceadapter> </resourceadapter> <adminobject> <adminobject-interface>javax.jms.Queue</adminobject-interface> <adminobject-class>org.activemq.message.ActiveMQQueue</adminobject-class> <adminobject-instance> <message-destination-name>OrderQueue</message-destination-name> <config-property-setting name="PhysicalName">OrderQueue</config-property-setting> </adminobject-instance> <adminobject-instance> <message-destination-name>ConsignmentQueue</message-destination-name> <config-property-setting name="PhysicalName">ConsignmentQueue</config-property-setting> </adminobject-instance> </adminobject> <adminobject> <adminobject-interface>javax.jms.Topic</adminobject-interface> <adminobject-class>org.activemq.message.ActiveMQTopic</adminobject-class> </adminobject> </connector>

If the **<adminoject-interface>** is used to identify the type of JMS resource interface which has been used(Queue)and **<adminobject-instance>** segment is configured the JMS resource which is already deployed in the server(OrderQueue) and it's configuration property name setting (Phisical name).If you are generating using the geronimo web console,the wizard will ask for the values for all of these and the plan will be generated for at the end.

Resolving Container-Managed Relationships

Container-managed relationships are initially defined in the ejb-jar.xml deployment descriptor, but the mappings to specific database elements are defined here. The mapping strategy is different depending on the type of relationship one-to-one, one-to-many, or many-to-many. The high-level elements used to map container-managed relationships are defined bellow.

ejb-relationship-role-name

Can be used as a convenience to indicate which relationship role this block applies to, in which case it ought to match the ejb-relationship-role/ejb-relationship-role-name in ejb-jar.xml. However, Geronimo does not actually pay attention to this element, so it's more for documentation purposes. The relationship in question will be uniquely identified by the combination of relationship-role-source and cmr-field instead (or by the relationship-role-source if only the other ejb-relationship-role in the relationship has a cmr-field).

relationship-role-source
Identifies the EJB that is associated with this relationship role. With cmr-field, (or if this ejb-relationship-role is paired with another ejb-relationship-role that has a cmr-field), this field uniquely identifies the relationship role which these settings apply to.

ejb-name

Holds the name of the EJB that forms this relationship role. This must match the relationship-role-source/ejb-name for this ejb-relationship-role in ejb-jar.xml.

cmr-field

If the ejb-relationship-role in ejb-jar.xml includes a cmr-field, this element can be used with the relationship-role-source to uniquely identify the relationship role these settings apply to. Otherwise, this must be one of two ejb-relationship-role blocks for this ejb-relation, and the other ejb-relationship-role must include a cmr-field.

cmr-field-name

The name of the cmr-field on the EJB. This must match the cmr-field/cmr-field-name for this relationship role in ejb-jar.xml.

foreign-key-column-on-source

For one-to-one and one-to-many relationships, the server normally expects that if an EJB (say, Person) declares a CMR field (say, getAddress()), then the underlying ADDRESS table has a column like PERSON_ID on it (referring to the primary key of the PERSON table, say PERSON.ID). So the EJB declaring the CMR field has the table with the primary key, and the other EJB has the foreign key. If that is true, then this element should not be present.

However, if the CMR field is still Person.getAddress() but it's instead the PERSON table that has a column like ADDRESS_ID on it (referring to say, ADDRESS.ID), then this element should be present to indicate that it's actually the case that the EJB with the CMR field has the foreign key, and the other EJB has the primary key.

This element never has any content; the data is conveyed by whether it's present or not. It should never be used for many-to-many mappings.

role-mapping

This element holds the data that maps the relationship to specific database columns (typically based on foreign keys, though Geronimo does not actually require that the database include actual foreign key constraints).
cmr-field-mapping

Holds a single primary-key to foreign-key mapping. There should be one cmr-field-mapping element for each column in the primary key (and therefore the foreign key that refers to it).

key-column

The primary key column in the database. If foreign-key-column-on-source is not present, then this should be a column on the table for the EJB named in the relationship-role-source element. Otherwise, this should be a column on the table for the other EJB in the relationship.

foreign-key-column

The foreign key column in the database. If foreign-key-column-on-source is present, then this should be a column on the table for the EJB named in the relationship-role-source element. Otherwise, this should be a column on the table for the other EJB in the relationship.

Following openejb-jar.xml has taken from the http://cwiki.apache.org/GMOxDOC11/ejb-sample-application.html.It has defined the dependencies in the **<environment>** segment and **<cmp-connection-factory>** is set to the Data Base plan.

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-1.1"> <dep:moduleId> <dep:groupId>default</dep:groupId> <dep:artifactId>BankEJB</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> <cmp-connection-factory> <resource-link>BankPool</resource-link> </cmp-connection-factory> <enterprise-beans> <session> <ejb-name>BankManagerFacadeBean</ejb-name> <jndi-name>org.apache.geronimo.samples.bank.ejb.BankManagerFacadeBean</jndi-name> <ejb-ref> <ref-name>ejb/Customer</ref-name> <ejb-link>Customer</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/Account</ref-name> <ejb-link>Account</ejb-link> </ejb-ref> <ejb-ref> <ref-name>ejb/ExchangeRate</ref-name> <ejb-link>ExchangeRate</ejb-link> </ejb-ref> </session> <entity> <ejb-name>Account</ejb-name> <local-jndi-name>AccountLocalEntity</local-jndi-name> <table-name>Account</table-name> <cmp-field-mapping> <cmp-field-name>accountNumber</cmp-field-name> <table-column>ACC_NO</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>accountType</cmp-field-name> <table-column>ACC_TYPE</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customer</cmp-field-name> <table-column>CUSTID_FK</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>balance</cmp-field-name> <table-column>BALANCE</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>Customer</ejb-name> <local-jndi-name>CustomerLocalEntity</local-jndi-name> <table-name>Customer</table-name> <cmp-field-mapping> <cmp-field-name>customerId</cmp-field-name> <table-column>CUST_ID</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customerName</cmp-field-name> <table-column>CUST_NAME</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> <entity> <ejb-name>ExchangeRate</ejb-name> <local-jndi-name>ExchangeRate</local-jndi-name> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity> </enterprise-beans> <relationships> <ejb-relation> <ejb-relation-name>Customer-Account</ejb-relation-name> <ejb-relationship-role> <ejb-relationship-role-name>Account-to-Customer</ejb-relationship-role-name> <relationship-role-source> <ejb-name>Account</ejb-name> </relationship-role-source> <cmr-field> <cmr-field-name>customer</cmr-field-name> </cmr-field> <foreign-key-column-on-source/> <role-mapping> <cmr-field-mapping> <key-column>CUST_ID</key-column> <foreign-key-column>CUSTID_FK</foreign-key-column> </cmr-field-mapping> </role-mapping> </ejb-relationship-role> </ejb-relation> </relationships> </openejb-jar>

Let's consider the CMP mapping implementation of the following code which is excerpt from the above openejb-jar.xml

xmlsolidExcerpt from openejb-jar.xml <entity> <ejb-name>Customer</ejb-name> <local-jndi-name>CustomerLocalEntity</local-jndi-name> <table-name>Customer</table-name> <cmp-field-mapping> <cmp-field-name>customerId</cmp-field-name> <table-column>CUST_ID</table-column> </cmp-field-mapping> <cmp-field-mapping> <cmp-field-name>customerName</cmp-field-name> <table-column>CUST_NAME</table-column> </cmp-field-mapping> <resource-ref> <ref-name>jdbc/BankDataSource</ref-name> <resource-link>BankPool</resource-link> </resource-ref> </entity>

Each DataBase mapping are configured with in the **<cmp-field-mapping>** and **<cmp-field-mapping>** is set to the **customerId** which is deployment plan specific reference for the name of the data field."CUS_ID" is the actual name of the data base field in the DataBase.

xmlsolidBankDB.sql CREATE TABLE customer(CUST_ID VARCHAR(15) PRIMARY KEY, CUST_NAME VARCHAR(40));