JAW Examples for JBoss AS 5

Bill Quirk, Qwest Communications, Inc.
(Bill.Quirk@qwest.com)

These are the examples that go with the 'JBoss at Work' Book published by O'Reilly (ISBN 0-596-00734-5). The examples have been updated for JBoss App Server 5 and include examples using Java EE5 EJB 3 and JPA.

Versioning and Preparation

- The runtime environment used is jre1.6.0_14
- The build environment used is
  - jdk1.6.0_14
  - apache-ant-1.7.1
  - xdoclet-1.2.3
- JWSDP 2.0 was used for chapter 10 and is available at:
  - http://java.sun.com/webservices/downloads/previous/webservicespack.jsp
- AXIS 1.4 was used for chapter 10 and is available at:
  - http://www.apache.org/dyn/closer.cgi/ws/axis/1_4
  - Define an AXIS_1_4_HOME environment variable and set it to the directory where you placed the distribution.
- All EE jars, except the included jsp, taglib, commons-logging and log4j jars, are from the JBoss distribution:
  - ${env.JBOSS_HOME}/common/lib/.
- The examples were tested with
  - RedHat server version EAP 5.0
    - All examples work as-is with this version
    - Release Notes:
      - http://www.redhat.com/docs/en-US/JBoss_Enterprise_Application_Platform/5.0.0/html-single/Release_Notes_GA/index.html
  - JBoss.org version jboss-5.1.0.GA
    - The examples in chapters 1-9 and Appendix B work as-is with this version.
    - The chapter 10 example works with the following Java 6 fix recommended in the JBoss.org 5.1.0 release notes:
      - When running under Java 6 you need to manually copy the following libraries from the JBOSS_HOME/client directory to the JBOSS_HOME/lib/endorsed directory, so that the JAX-WS 2.0 apis supported by JBossWS are used:
        - jbossws-native-saaj.jar; jbossws-native-jaxrpc.jar; jbossws-native-jaxws.jar; jbossws-native-jaxws-ext.jar
    - Release Notes:
      - http://anonsvn.jboss.org/repos/jbossas/tags/JBoss_5_1_0_GA/build/docs/readme.html

A new environment variable JBOSS_CONFIG_NAME is added to the build scripts. You must set this environment variable to the server configuration you are using:

- Set JBOSS_CONFIG_NAME to 'default', 'production', etc. for RedHat EAP.
- Set JBOSS_CONFIG_NAME to 'default' if you are using the JBoss.org 5.0.0 GA.

Chapter 1 (ch01)

- Stays the same.

Chapter 2 (ch02)

- compile-lib is supplanted, using jars from ${env.JBOSS_HOME}/common/lib/ and is no longer needed. Ant scripts updated to reflect this.
- Note: Tomcat is now deployed in the 'deploy/jbossweb.sar' directory.

Chapter 3 (ch03)

- webapp compile-lib is supplanted, using jars from ${env.JBOSS_HOME}/common/lib/ and is no longer needed. Ant scripts updated to reflect this.

Chapter 4 (ch04)

- webapp compile-lib is supplanted, using jars from ${env.JBOSS_HOME}/common/lib/ and is no longer needed. Ant scripts updated to reflect this.

- In 5.0 the hsqldb.jar (and other common jars formerly in: '${env.JBOSS_HOME}/server/${env.JBOSS_CONFIG_NAME}"/lib' were migrated by JBoss to: '${env.JBOSS_HOME}/common/lib/ Ant scripts updated to reflect this.

Chapter 5 (ch05)

- webapp compile-lib is supplanted, using jars from ${env.JBOSS_HOME}/common/lib/ and is no longer needed. Ant scripts updated to reflect this.

- In 5.0 the hsqldb.jar (and other common jars formerly in: '${env.JBOSS_HOME}/server/${env.JBOSS_CONFIG_NAME}"/lib' were migrated by JBoss to: '${env.JBOSS_HOME}/common/lib/ Ant scripts updated to reflect this.

- In HibernateCarDAO.java added a call to 'session.isOpen()' in the finally block test to close the Hibernate session to silence an 'already closed' exception. Commits and rollbacks cause the Hibernate session to be closed automatically.

- In 5.0 the file hibernate/hibernate-service.xml has changed to XXX-hibernate.xml with new content. All files in the common/hibernate directory changed to reflect this.

    See: http://community.jboss.org/wiki/JBossHibernate3?decorator=print Section 'Hibernate (M)Bean in JBossAS5' for more information.

Chapters 6-9, Appendix B (ch06,ch07/ch07,ch08/ch08,ch09 & appb)

- In 5.0 JBoss ejb deployer wants local reference EJB class names fully qualified. Changed ejb class names to fully qualified class names in ControllerServlet XDoclet ejb comments:

  @web.ejb-local-ref

  * name="ejb/InventoryFacadeLocal"
  * type="Session"
  * home="com.jbossatwork.ejb.InventoryFacadeLocalHome"
  * local="com.jbossatwork.ejb.InventoryFacadeLocal"

- Common, ejb, and webapp compile-lib are supplanted, using jars from ${env.JBOSS_HOME}/common/lib/ and is no longer needed. Ant scripts updated to reflect this.

- In 5.0 the hsqldb.jar (and other common jars formerly in: '${env.JBOSS_HOME}/server/${env.JBOSS_CONFIG_NAME}"/lib' were migrated by JBoss to: '${env.JBOSS_HOME}/common/lib/ Ant scripts updated to reflect this.

- In HibernateCarDAO.java added a call to 'session.isOpen()' in the finally block test to close the Hibernate session to silence an 'already closed' exception. Commits and rollbacks cause the Hibernate session to be closed automatically.

- In 5.0 the file hibernate/hibernate-service.xml has changed to XXX-hibernate.xml with new content. All files in the common/hibernate directory changed to reflect this.

  See: http://community.jboss.org/wiki/JBossHibernate3?decorator=print Section 'Hibernate (M)Bean in JBossAS5' for more information.

- Chapter 6 contains new example content, based upon the ch06-c final example of chapter 6. The new content is in directory ch06-c-EE5 and features EE5, EJB3 and JPA Annotations. See appendix A for a complete description.

- Chapter 7 contains new example content, based upon the ch07 example. The new content is in directory ch07/ch07-EE5 and features EE5, EJB3 and JPA Annotations. See appendix B for a complete description.

- Chapter 8 contains new example content, based upon the ch08 example. The new content is in directory ch08/ch08-EE5 and features EE5, EJB3 and JPA Annotations. See appendix C for a complete description.

- Chapter 9 contains new example content, based upon the ch09-c example. The new content is in directory ch09/ch09-c-EE5 and features EE5, EJB3 and JPA Annotations. See appendix D for a complete description.

Chapter 10 (ch10/ch10)

- **Note for Jboss.org server version:** The example works with the following Java 6 fix recommended in the JBoss.org 5.1.0 release notes:
    - When running under Java 6 you need to manually copy the following libraries from the JBOSS_HOME/client directory to the JBOSS_HOME/lib/endorsed directory, so that the JAX-WS 2.0 apis supported by JBossWS are used:
        - jbossws-native-saaj.jar
        - jbossws-native-jaxrpc.jar
        - jbossws-native-jaxws.jar
        - jbossws-native-jaxws-ext.jar

- In 5.0 JBoss ejb deployer wants local reference EJB class names fully qualified. Changed ejb class names to fully qualified class names in ControllerServlet XDoclet ejb comments:

    @web.ejb-local-ref

    * name="ejb/InventoryFacadeLocal"

    * type="Session"

    * home="com.jbossatwork.ejb.InventoryFacadeLocalHome"

    * local="com.jbossatwork.ejb.InventoryFacadeLocal"

- Common, ejb, and webapp compile-lib are supplanted, using jars from ${env.JBOSS_HOME}/common/lib/ and is no longer needed. Ant scripts updated to reflect this.

- In 5.0 the hsqldb.jar (and other common jars formerly in: '${env.JBOSS_HOME}/server/${env.JBOSS_CONFIG_NAME}"/lib' were migrated by JBoss to: '${env.JBOSS_HOME}/common/lib/ Ant scripts updated to reflect this.

- In 5.0 the file hibernate/hibernate-service.xml has changed to XXX-hibernate.xml with new content. All files in the common/hibernate directory changed to reflect this.

    See: http://community.jboss.org/wiki/JBossHibernate3?decorator=print Section 'Hibernate (M)Bean in JBossAS5' for more information.

- 5.0 is strict on ordering of jboss.xml <service> elements. Because of this I had to fix the fix in the ejb directory build.xml to place the port-component element after the method-attributes:

- **Note:** This replace only works correctly because we have one and only one session bean. If we had multiple beans this replace would have added a port-component to each session bean after each method-attributes element. If you have more than one session bean you will have to hand-edit the jboss.xml or more probably move to EE5 annotations.

```
<replace file="${gen.source.dir}/jboss.xml">
<replacetoken><![CDATA[</method-attributes>]]></replacetoken>
 <replacevalue><![CDATA[
            </method-attributes>
            <port-component>
              <port-component-name>Inventory</port-component-name>
                                                  <port-component-uri>jbossatwork-
                        ws/InventoryService</port-component-uri>
            </port-component>
          ]]></replacevalue>
        </replace>
```

- In JBoss5 the Axis-based Web Service stack has been replaced by the JBossWS stacks:
  - Documentation for the JBossWS stack can be found here:
    - http://community.jboss.org/wiki/JBossWS
  - To see the deployed InventoryService you now navigate to:
    - http://localhost:<yourport>/jbossws
  - I recommend you review the new JBossWS documentation and get familiar with using JAX-WS based Web Services. JAX-WS is widely regarded as being superior to the previous JAX-RPC based services. As well, many other useful Web Services related standards covering security, transactions, etc. have matured and are supported by JBossWS.
- Now that we are at J2SE 1.5+ and we need to use AXIS 1.2+. The build.xml script in the client directory is changed to reflect this.
  - Download AXIS 1.4 from Apache.org.
  - Define an AXIS_1_4_HOME environment variable and set it to the directory where you placed the distribution.
- I changed MyAxisClient.java by un-commenting the Axis 1.2+ lines of code and commenting-out the Axis 1.1 lines.
- Chapter 10 contains new example content, based upon the ch10 example. The new content is in directory ch10/ch10-EE5 and features EE5, EJB3 and JPA Annotations. See appendix E for a complete description.

Appendix A – Chapter 6 example for JBOSS AS 5, EE5, EJB3 and JPA with Annotations.

Chapter 6 contains new example content, based upon the ch06-c final example of chapter 6. The new content is in directory ch06-c-EE5 and features EE5, EJB3 and JPA Annotations.

All of the XDoclet tasks and comments (useful as I still agree they are) have been removed from the new example as EE5 annotations make them superfluous.

For more information please consult the JBoss and Java documentation:

- http://www.jboss.org/file-access/default/members/jbossejb3/freezone/docs/tutorial/1.0.7/html/index.html
- http://java.sun.com/javaee/5/docs/tutorial/doc/

Note: When invoking this example using the jboss.org 5.1.0 GA server you will see a warning, EJBTHREE-1246. Per jBoss.org (https://jira.jboss.org/jira/browse/EJBTHREE-1246) this can be safely ignored.

The changes needed to migrate this chapter example are:

- common directory
    - In CarDTO and AccountingDTO:
        - Add: import javax.persistence.*;
        - Add: @Entity, @Table, @Id, @GeneratedClass and @Column annotations, removing all corresponding Hibernate XDoclet comments.
    - In HibernateCarDAO and HibernateAccountingDAO
        - Add: import javax.persistence.*;
        - Add: private EntityManager manager;
        - Add: EntityManager manager to constructor
        - Change Hibernate calls to JPA calls:
            - delete() -> remove()
            - update() -> merge()
            - create() -> persist()
            - find() -> find()

- Criterion() methods are not supported until EE6 JPA so we use the createQuery() methods instead. The ‘SQL’ you see is not the native SQL. It is the JPA EJB3-QL that operates on the objects, not the tables.

  - ServiceLocator.java:
    - Removed this class as notations simplify this.
  - Removed all build script usage of XDoclet.
  - Replaced the hibernate directory with a new persistence directory.
    - Added persistence.xml that specifies the named persistence unit used by the DAO classes. This links up the data source with the annotated user of the JPA interface.
  - Reference the AS 5.0 /common/lib for compile time library usage to get current EE5 jars used by JBoss 5

- ejb directory
  - Create interface InventoryFacadeLocal with InventoryFacade methods.
    - import javax.ejb3.*;
    - Add @Local annotation before Interface definition
  - Create interface InventoryFacadeRemote with InventoryFacade methods
    - import javax.ejb3.*;
    - Add @Remote annotation before Interface definition
  - In InventoryFacade.java:
    - Add: import javax.ejb3.*;
    - Add: @Stateless annotation before class definition
    - Modify class definition: public class InventoryFacade implements InventoryFacadeLocal, InventoryFacadeRemote
    - Remove superfluous XDoclet ejb descriptor comments.
    - Remove superfluous ejb lifecycle methods.
    - In the body of the class add
      - @PersistenceContext(unitName="jawEntityManager")

        - Remember that ‘jawEntityManager ‘ is defined in the persistence.xml.

    - EntityManager em;

    - **Note:** This cannot be in the DAO POJO, it must be in a container managed class. The em, however is passed into the POJO and still takes advantage of the transactional persistence context.

  - Supplant the XDoclet comments with the transaction annotation before each method:

    - @TransactionAttribute(TransactionAttributeType.REQUIRED)

  - Reference the AS 5.0 /common/lib for compile time library usage to get current EE5 jars used by JBoss 5

- build.xml

  - Remove J2EE 2.1 local jar and add JBoss AS lib in classpath for EJB3 annotations and imports.

  - Remove the superfluous gen_src_dir and XDoclet EJB Descriptor artifact generation

  - build now only packages the class files in the ejb jar.

- src directory

  - Removed jboss-app.xml – jaw.har no longer created therefore no need to specify it.

- webapp directory

  - Remove XDoclet tags and build file XDoclet generation and create a simple web.xml.

    - **Note:** You must update the version of the web.xml to 2.5 and the xml namespace to http://java.sun.com/xml/ns/javaee for the annotations to work within a web application.

  - Added test for inventory ejb reference equal null as this can happen with annotations.

  - Modify controller servlet by removing call to ServiceLocator and all the Home object references and replace with either:

    - **Java 2 style JNDI lookup of the local object**

```
private static final String COMP_NAME_LOCAL= "/jaw/InventoryFacade/local";

InventoryFacadeLocal inventory = null;

try {

        InitialContext ctx = new InitialContext();

        inventory = (InventoryFacadeLocal) ctx.lookup(COMP_NAME_LOCAL);

} catch (Exception se) {

        throw new RuntimeException(se.getMessage());

}
```

- **Java 2 style JNDI lookup of the remote object**

```
private static final String COMP_NAME_REMOTE=
"/jaw/InventoryFacade/remote";

InventoryFacadeRemote inventory = null;

try {

        InitialContext ctx = new InitialContext();

        inventory = (InventoryFacadeRemote)
        ctx.lookup(COMP_NAME_REMOTE);

} catch (Exception se) {
        throw new RuntimeException(se.getMessage());
}
```

- **EJB3 style annotation declaration for the local object**

```
@EJB InventoryFacadeLocal inventory;
```

  - Note: You must update the version of the web.xml to 2.5 and the xml namespace to http://java.sun.com/xml/ns/javaee for the annotations to work within a webapp.

- **EJB3 style annotation declaration for the remote object**

```
@EJB InventoryFacadeRemote inventory;
```

  - Note: You must update the version of the web.xml to 2.5 and the xml namespace to http://java.sun.com/xml/ns/javaee for the annotations to work within a webapp.

Appendix B – Chapter 7 example for JBOSS AS 5, EE5, EJB3 and JPA with Annotations.

Chapter 7 contains new example content, based upon the ch07 example. The new content is in directory ch07/ch07-EE5 and features EE5, EJB3 and JPA Annotations.

The changes needed to migrate this chapter example are:

- All updates listed in appendix 'A'
- common directory
    - JmsProducer
        - Add: import javax.annotation.Resource;
        - Two ways to setup for sending the JMS message:
            - **Java 2 Style using JNDI lookup**
                - **Note:** This preserves the original signature call, where the ControllerServlet passes in the string JNDI names to use. Notice here the strings are used directly as a transitional step to using notations. Notice also the use of the JBoss standard "java:/JmsXA" object that in turn uses the XAConnectionFactory to ensure proper container managed JMS transaction usage.

```
InitialContext ctx = new InitialContext();

ConnectionFactory connectionFactory = (ConnectionFactory)
ctx.lookup("java:/JmsXA");

Queue queue = (Queue) ctx.lookup("queue/CreditCheckQueue");

connection = connectionFactory.createConnection();

session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

messageProducer = session.createProducer(queue);
```

            - **EE5 Style using resource annotations**
                - **Note:** Resource annotations are supported in either stand-alone J2EE application classes or container managed classes (Servlets, EJBs, etc.). The common/utils factored

utility classes are neither an application nor a container managed class. So, for annotations to work here, we must:

- Add the Resource annotations in the ControllerServlet:
- Alter the signature of the JmsProducer.sendMessage() method to pass in the ConnectionFactory and Queue that are now injected in the container managed servlet.

- Replace the destination call with a messageproducer call using the injected queue resource

```
messageProducer = session.createProducer(queue);
```

- ejb directory
  - CreditCheckProcessorBean
    - Remove XDoclet comments.
    - Add: import javax.annotation.Resource;
    - Add Message Bean Annotations:

```
@MessageDriven(name="CreditCheckProcessorMDB",
activationConfig=
{ @ActivationConfigProperty(
   propertyName="destinationType",
   propertyValue="javax.jms.Queue"),
  @ActivationConfigProperty(
   propertyName="destination",
   propertyValue="queue/CreditCheckQueue"),
  @ActivationConfigProperty(
   propertyName="subscriptionDurability",
   propertyValue="NonDurable"),
  @ActivationConfigProperty(
   propertyName="acknowledgeMode",
```

propertyValue="Auto-acknowledge")

})

- Remove superfluous ejbCreate() method and ejbRemove() method.
  - **Note:** There are PostConstruct and PreDestroy annotations that do fulfill this functionality should it be needed.

- webapp directory
  - ControllerServlet
    - Add resource annotations above the processRequest method:

      @Resource(mappedName="java:/JmsXA")

      ConnectionFactory connectionFactory;

      @Resource(mappedName=”queue/CreditCheckQueue”)

      Queue queue;

    - Remove JNDI names (Now in resource annotation mappedName attribute)
    - Remove JNDI names from JmsProducer.sendMessage() formal parameters.
    - Add javax.jms.ConnectionFactory and javax.jms.Queue to JmsProducer.sendMessage() formal parameters.

Appendix C – Chapter 8 example for JBOSS AS 5, EE5, EJB3 and JPA with Annotations.

Chapter 8 contains new example content, based upon the ch08 example. The new content is in directory ch08/ch08-EE5 and features EE5, EJB3 and JPA Annotations.

The changes needed to migrate this chapter example are:

- All updates listed in appendix 'A' and appendix 'B'
- Ejb directory
    - CreditCheckProcessor.java
        - Remove ServiceLocator call in sendNotificationEmail method
        - Add resource annotation above sendNotificationEmail method:

```
@Resource(mappedName="java:/Mail")
javax.mail.Session javaMailSession;
```

            - Note: We use the JBoss java namespace name.

Appendix D – Chapter 9 example for JBOSS AS 5, EE5, EJB3 and JPA with Annotations.

Chapter 9 contains new example content, based upon the ch09-c example. The new content is in directory ch09/ch09-c-EE5 and features EE5, EJB3 and JPA Annotations.

The changes needed to migrate this chapter example are:

- All updates listed in appendix 'A', appendix 'B' and appendix 'C'
- ejb directory
    - InventoryFacade.java
        - Replace the security-role-ref XDoclet Annotation with the DeclareRoles annotation:
            - @DeclareRoles({"guest","Manager"})
        - Add the appropriate RolesAllowed annotations to each method stating the roles that can access the method:

          @RolesAllowed("Manager")

          @RolesAllowed({"guest","Manager"})
        - **Note:** There are many ways to combine the @RolesAllowed, @PermitAll, and @DenyAll annotations to accomplish the same thing. For instance, we could have annotated the class with @RolesAllowed("Manager") and then annotated the publicly accessable methods with @PermitAll or we could have annotated the class with @PermitAll and then annotated the privately accessable methods with @RolesAllowed("Manager") . We could also have annotated only the locked down methods with @RolesAllowed("Manager") The @DenyAll annotation is useful for turning off a method or methods without removing the code or commenting the code out. The more explicit you are the better your chances of getting the security right the first time.
        - **Note:** @PostConstruct and @PreDestroy annotated methods do not obey security annotations.
    - Add META-INF directory
        - Add jboss.xml that includes only the security-domain element.
        - **Note:** Even though we no longer use XDoclet here due to EJB3 annotations, we retain the declarative security that associates the EJB with the JawJaasDbRealm. Keeping the declarative security will also

allow you to override the annotated security in the EJB classes should you need to (For example, a new role, ‘Sales’, is defined that assumes the ‘Manager’ role)

- Build.xml
  - Remove XDoclet and set META-INF directory to pick up jboss.xml:

    <property name="meta.inf.dir" value="META-INF"/>

    <metainf dir="${meta.inf.dir}" includes="*.xml"/>

- webapp directory
  - WEB-INF
    - Add descriptor content that was created dynamically by XDoclet.
    - **Note:** The preferred method for security is still declarative security using xml descriptors and retaining the XDoclet as much as possible may be favorable for your application. Annotating the ejb code with EJB3 annotations or XDoclet annotations maintains the rules local to the methods and can be useful to the programmer’s view of security when enhancing or maintaining the application code. Declarative security always overrides annotations so you can if needed alter the method security model without touching the code.

Appendix E – Chapter 10 example for JBOSS AS 5, EE5, EJB3 and JPA with Annotations.

Chapter 10 contains new example content, based upon the ch10 example. The new content is in directory ch10/ch10-EE5 and features EE5, EJB3 and JPA Annotations.

The changes needed to migrate this chapter example are:

- All updates listed in appendix 'A', appendix 'B', appendix 'C' and 'D'
- ejb directory
    - Add imports for Web Services annotations:
        - **Note:** The @WebContext is specific to JBoss and is not portable. This can be removed and once removed the context root changes to the jar file (ejb-jar) and the Service changes to InventoryFacade. Please study the JBossWS documentation to discover all annotations possible.
        - import javax.jws.*;
        - import org.jboss.wsf.spi.annotation.WebContext;
    - Add Class-level annotations:
        - @WebService
        - @WebContext(contextRoot = "/jbossatwork-ws" , urlPattern="/InventoryService")
    - Add new findAvailableCars method and annotate as a web method.

```
@WebMethod()
  public CarDTO[] findAvailableCars() throws EJBException {
   CarDTO[] cars = (CarDTO[]) listAvailableCars().toArray(new CarDTO[0]);
      return cars;
  }
```

- client directory
    - MyAxisClient.java
        - Added package import for the dynamically created classes.
            - import com.jbossatwork.ejb.*;

- Update the names of the classes dynamically created by the Axis ant task due to the slight changes in WSDL nomenclature. **Note:** Please review the JBossWS documentation for the remaining annotations that can be used to tailor the nomenclature of the services, methods, etc. used when the WSDL is created and the service is deployed.

```
InventoryFacadeService service = new
InventoryFacadeServiceLocator();

System.out.println("Getting InventoryEndpoint ...\n");

InventoryFacade endpoint = service.getInventoryFacadePort();
```

Note on using EE5 Annotations

One of the drawbacks of annotations can be the requirement that hard-coded string constants must be used when defining the annotation. This has one advantage in that the configuration is somewhat local to the usage, though a change in the configuration can lead to a rebuild and redeploy of application. Having converted these application examples to EE5, I have found that:

1. Using annotations in the definition of EJBs seems to be desirable considering they seldom change.
2. Using annotations for ejb invocation seems ok as long as you are reasonably sure you will always invoke either local or remote.
3. Using annotations for resource injection seems ok as long as you are reasonably sure that the resource definition (target, location, URI, etc.) will remain unchanged.
4. If you are reasonably sure that locations, targets, configurations, etc. will change from environment to environment, or if you expect usage to change from local to remote invocations, utilize client-side J2EE style methods and XML configurations to take advantage of the more dynamic configuration capabilities.

Many test and production environments require a packaged, non-exploded deployment unit be supplied for any type of deployment (initial, bug-fix, patch, etc.). In this case a repackaging is required for each version of the deployment unit and may make the consideration of hard-coded annotations moot in the cases where the deployment unit is always rebuilt.

EJB references (from JBoss.org)

Rules for the @EJB annotation

The @EJB annotation also has a mappedName() attribute. The specification leaves this a vendor specific metadata, but JBoss recognizes mappedName() as the global JNDI name of the EJB you are referencing. If you have specified a mappedName(), then all other attributes are ignored and this global JNDI name is used for binding.

If you specify @EJB with no attributes defined:

```
@EJB ProcessPayment myEjbref;
```

Then the following rules apply:

The EJB jar of the referencing bean is contained in is search for another EJB with the same interface. If there are more than one EJB that publishes same business interface, throw an exception, if there is one, use that one.

Search the EAR for EJBs that publish that interface. If there are duplicates, throw an exception, otherwise return that one.

Search globally in JBoss for an EJB of that interface. Again, if duplicates, throw an exception

@EJB.beanName() corresponds to <ejb-link>. If the beanName() is defined, then use the same algorithm as @EJB with no attributes defined except use the beanName() as a key in the search. An exception to this rule is if you use the ejb-link '#' syntax. The '#' syntax allows you to put a relative path to a jar in the EAR where the EJB you are referencing lives. See spec for more details

For XML the same rules apply as annotations exception <mapped-name> is the ejb-jar.xml equivalent to @EJB.mappedName().

Avoiding port binding problems

You may upon startup see JBoss log port binding problems, that is, ports already in use on your PC/server. One way to avoid this and subsequently use this to create a clustered environment with multiple servers running is to create a script with the jboss.service.binding.set option:

run -c default -Djboss.service.binding.set=ports-01

This adds 100 to each port created by JBoss so HTTP is now at port 8180, etc. To create multiple servers create multiple scripts for ports-01,ports-02,ports-03,ports-04,etc. (ports-04 HTTP is at 8480…)