JBoss 3.0 Step-by-Step Tutorial

Abstract
JBoss, an open source application and EJB server, is a complex tool. This article attempts to walk the new user through the use of Jboss 3.0 from installation through the deployment and use of a "Hello World" EJB and client application. (July 17, 2002)

Using JBoss can be a daunting task for the first time. I will attempt to describe every step of the process from installing JBoss to getting a client to calling a "Hello World" Enterprise Java Bean from a client located on a different machine. The second machine is not required but emphasizes the power of Enterprise Java Beans (EJB's) and the transparency of location they can have.

Getting and Installing JBoss

First make sure you have a recent JDK such as 1.3 or higher. The next step is obtaining JBoss from the JBoss website. You can find the JBoss binaries (executables) at http://www.jboss.org/downloads.jsp. At the time of writing the file to retrieve is JBoss-3.0.0.zip (jboss-3.0.0_tomcat-4.0.3.zip) with integrated Tomcat 4.0. Tomcat is the official reference implementation for servlet containers. More information about tomcat can be found at the Apache Jakarta project.

Once the file finishes downloading you need to extract the zip file to the directory of your choice. For Windows you can use WinZip and for Linux simply use `unzip` at the command line to unzip the file. I chose to extract my file to:

```
F:\jboss-3.0.0_tomcat-4.0.3
```

Starting JBoss

Starting JBoss is easy, simply change to the `bin` directory where JBoss is installed and execute run.bat on windows or run.sh on Linux. So the sequence of commands when starting at `C:\` was:

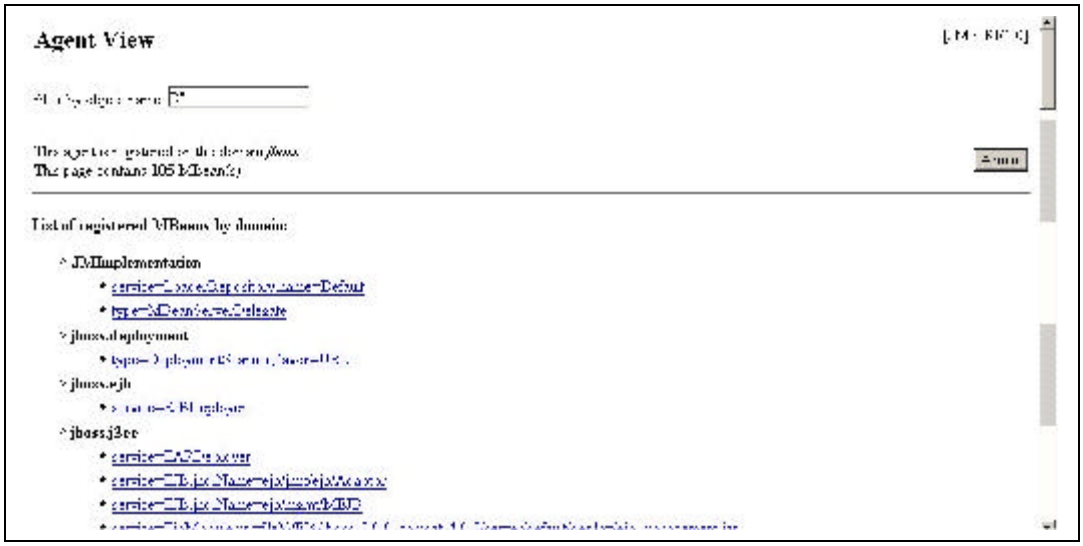
```
F:
cd \jboss-3.0.0_tomcat-4.0.3
cd bin
run
```

Once I typed "`run`" a lot of logging information was displayed until the last line of logging output stopped displaying:

```
23:01:20,309 INFO  [Server] JBoss (MX MicroKernel) [3.0.0
Date:200205311035] Started in 0m:43s:973ms
```

Now that JBoss has been started we can test that it is up and running using the web interface provided through the integrated version of Tomcat. Simply point your browser to http://localhost:8082/ (see figure 1) and you should see a page titled "Agent View".

Figure 1



Writing the HelloWorld EJB

Now we write a simple “hello world” EJB. I will presume you understand how the pieces of EJB’s fit together if not I would recommend getting Master Enterprise Java Beans, 2nd Edition or see the references at the end of this tutorial. We will begin by writing the server side which is the Enterprise Java Bean.

The Server Side Enterprise Java Bean

The first class to write (in no particular order) is the interface the EJB client will interact with.

Remote Interface

We will start by coding the remote interface. These are the business methods that the client will call to perform work.

HelloWorld.java

```
package com.mastertech.sample;

/**
 * This is the remote interface that the client calls to
 * have the EJB do the work.
 */
public interface HelloWorld extends javax.ejb.EJBObject
{
    public String hello() throws java.rmi.RemoteException;
}
```

Home Interface

The home interface acts as a factory for EJB's. The home interface is how the container creates and destroys the EJB's used by applications.

HelloWorldHome.java

```
package com.mastertech.sample;

/**
 * HelloWorldHome provides the container the means to
 * create and destroy EJB's.
 */
public interface HelloWorldHome extends javax.ejb.EJBHome
{
    HelloWorld create() throws java.rmi.RemoteException,
        javax.ejb.CreateException;
}
```

The Bean Implementation

Below is the actual bean class that implements the clients interface.

HelloWorldBean.java

```
package com.mastertech.sample;

import javax.ejb.SessionContext;

/**
 * This class is the actual implementation of the business
 * logic.  This is the EJB for simplicity's sake.
 */
public class HelloWorldBean implements javax.ejb.SessionBean
{
    private SessionContext ctx;

    public void setSessionContext(SessionContext ctx)
    {
        this.ctx = ctx;
    }

    public void ejbRemove()
    {
        System.out.println( "ejbRemove()" );
    }

    public void ejbActivate()
    {
        System.out.println( "ejbActivate()" );
    }

    public void ejbPassivate()
    {
        System.out.println( "ejbPassivate()" )
    }

      /**
       * The method called to display the string "Hello World!"
       * on the client.
       */
    public String hello()
    {
        System.out.println( "hello()" );
        return "Hello World!";
    }
}
```

Deployment Descriptor

The XML deployment descriptor that provides the plumbing between the container and the beans you write. Below is the deployment descriptor we will use.

ejb-jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar>
  <description>JBoss Hello World Application</description>
  <display-name>Hello World EJB</display-name>
  <enterprise-beans>
    <session>
      <ejb-name>Hello</ejb-name>
      <home>com.mastertech.sample.HelloHome</home>
      <remote>com.mastertech.sample.Hello</remote>
      <ejb-class>com.mastertech.sample.HelloBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Bean</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Once the deployment descriptor is complete it needs to be put into the META-INF directory. The META-INF directory should reside at the root of the package directory. For this example that is in the F:\projects\jboss-tutorial directory but yours may be where ever your project resides. My complete project directory structure is as follows:

```
F:\projects\jboss-tutorial\com
F:\projects\jboss-tutorial\com\mastertech
F:\projects\jboss-tutorial\com\mastertech\sample
F:\projects\jboss-
tutorial\com\mastertech\sample\HelloWorld.class
F:\projects\jboss-
tutorial\com\mastertech\sample\HelloWorldBean.class
F:\projects\jboss-
tutorial\com\mastertech\sample\HelloWorldHome.class
F:\projects\jboss-tutorial\META-INF
F:\projects\jboss-tutorial\META-INF\ejb-jar.xml
```

JBoss.xml

According to the JBoss documentation the Jboss.xml file is almost never required so it will be ignored for this tutorial.

Put That Bean in a Jar!

Now the bean classes and deployment descriptor need to be placed into a jar file before we can deploy the EJB to the JBoss application server. To create the deployment jar file first change directories to the root of the project directory with:

```
cd F:\projects\jboss-tutorial
```

Then execute the `jar` command to create the deployable jar file with:

```
jar cf HelloWorld.jar com META-INF
```

This will result in a file called “HelloWorld.jar” in the F:\projects\jboss-tutorial directory.

Deploy HelloWorld EJB

To deploy the HelloWorld.jar file simply copy the jar file to the JBoss deployment directory which is (on my computer):

```
F:\jboss-3.0.0_tomcat-4.0.3\server\default\deploy
```

Upon deployment the servers console will change. This is what my JBoss 3.0 server displayed:

```
15:09:21,184 INFO  [MainDeployer] Starting deployment of
package: file:/F:/jboss
-3.0.0_tomcat-4.0.3/server/default/deploy/HelloWorld.jar
15:09:21,324 INFO  [EjbModule] Creating
15:09:21,354 INFO  [EjbModule] Deploying HelloWorld
15:09:21,464 INFO  [EjbModule] Created
15:09:21,484 INFO  [EjbModule] Starting
15:09:21,555 INFO  [EjbModule] Started
15:09:21,555 INFO  [MainDeployer] Successfully completed
deployment of package: file:/F:/jboss-3.0.0_tomcat-
4.0.3/server/default/deploy/HelloWorld.jar
```

When I had some errors in my ejb-jar.xml I got the following message on my JBoss console:

```
16:13:17,007 INFO  [MainDeployer] Starting deployment of
package: file:/F:/jboss
-3.0.0_tomcat-4.0.3/server/default/deploy/HelloWorld.jar
16:13:17,027 INFO  [MainDeployer] Deployment of package:
file:/F:/jboss-3.0.0_tomcat-
4.0.3/server/default/deploy/HelloWorld.jar is waiting for
an appropriate deployer.
```

The problem was resolved by fixing finding and fixing the problem with the `ejb-jar.xml`.

The Client Side

There is no point in having an EJB if we don't have a client application to you use it. Now we need to create the client for the HelloWorld EJB we wrote. Below is the source code for a command line client that can be executed on the same machine.

HelloWorldClient.java

```
package com.mastertech.sample;

import javax.naming.Context;
import javax.naming.InitialContext;
import java.util.Hashtable;

public class HelloWorldClient
{
    public static void main( String [] args )
    {
        Hashtable env = new Hashtable();

        env.put(Context.INITIAL_CONTEXT_FACTORY,
            "org.jnp.interfaces.NamingContextFactory");
        env.put(Context.PROVIDER_URL, "localhost:1099");
        env.put("java.naming.factory.url.pkgs",
            "org.jboss.naming:org.jnp.interfaces");

        try
        {
            Context ctx = new InitialContext(env);

            Object obj = ctx.lookup( "HelloWorld" );
            HelloWorldHome home =
               (HelloWorldHome)javax.rmi.PortableRemoteObject.narrow(
               obj, HelloWorldHome.class );
            HelloWorld helloWorld = home.create();

            System.out.println( helloWorld.hello());
            helloWorld.remove();
        }
        catch ( Exception e )
        {
            e.printStackTrace();
            System.out.println( "Exception: " + e.getMessage() );
        }
    }
}
```

Once HelloWorldCLient.java is compiled you can execute it on the same machine with no changes. To execute HelloWorldClient.java on another machine you need to change the line:

```
env.put(Context.PROVIDER_URL, "localhost:1099");
```

To point to the machine that has the lookup services. On my network I simply changed the above line to point to:

```
env.put(Context.PROVIDER_URL,
"server.mastertech.net:1099");
```

For the client code to execute you need some JBoss client jar files in your class path in addition to a jar containing the remote interfaces and stubs to the HelloWorld EJB we wrote. On the machine where I ran HelloWorldClient.class I had the following in my classpath:

```
C:\j2ee131\lib\j2ee.jar
```

```
F:\jboss-3.0.0_tomcat-4.0.3\client\log4j.jar
F:\jboss-3.0.0_tomcat-4.0.3\client\jboss-common-client.jar
F:\jboss-3.0.0_tomcat-4.0.3\client\jboss-system-client.jar
F:\jboss-3.0.0_tomcat-4.0.3\client\jnp-client.jar
F:\jboss-3.0.0_tomcat-4.0.3\client\jboss-client.jar
F:\jboss-3.0.0_tomcat-4.0.3\client\jbosssx-client.jar
F:\projects\HelloWorld.jar
```

Hello World!

When you execute the HelloWorldClient you will see "Hello World!" on your console.

As you can see getting JBoss up and running the first time can be quite challenging. I would strongly recommend aquiring the JBoss documentation at a nominal charge from the JBoss web site. They will also have a free Getting Started guide soon.

Author Bio

Scott Pumer is a consultant with Masterpiece Technology, Inc. in Raleigh, NC. He has been programming for over eight years with much of his recent work being java-based web applications.

Recommended Reading

Ed Roman, Scott Ambler, and Tyler Jewel. *Mastering Enterprise Java Beans*, 2nd Edition, Wiley Computer Publishing, New York, 2002

TheServerSide.com, A J2EE Community, http://www.theserverside.com

Java 2 Platform, Enterprise Edition (J2EE), http://java.sun.com/j2ee/