Getting started

The OpenEJB Eclipse plugin provides the org.apache.openejb.server plugin which can be run as an OSGi bundle in a container such as Equinox or as part of an Eclipse Rich Client Platform (RCP) application.

First of all, you will need to obtain the org.apache.openejb.server bundle. If you develop using Eclipse, you may wish to consider installing the OpenEJB Eclipse Plugin, as this contains the necessary bundle.

I've provided a sample Equinox directory with OpenEJB and a sample EJB installed here: http://people.apache.org/~jgallimore/openejb-osgi.zip - the Eclipse projects for the EJB and client are here: http://people.apache.org/~jgallimore/projects.zip

I've also produced a simple RCP application based on the OpenEJB injection of entity manager example, which is available here http://people.apache.org/~jgallimore/rcp-app-source.zip

If you wish to build the plugin, have a look at: http://cwiki.apache.org/confluence/display/OPENEJB/Building+from+source

Deploying the sample bundles in Equinox

You can either use the OSGi console to deploy your bundles, or alternatively you can build a directory structure with all the necessary bundles and a config.ini.

The org.apache.openejb.server bundle has the following dependencies:

- org.eclipse.core.contenttype
- org.eclipse.core.jobs
- org.eclipse.core.runtime
- org.eclipse.core.variables
- org.eclipse.equinox.app
- org.eclipse.equinox.common
- org.eclipse.equinox.launcher
- org.eclipse.equinox.preferences
- org.eclipse.equinox.registry
- org.eclipse.osgi

These are most easily picked up from an existing Eclipse installation.

To deploy the org.apache.openejb.server bundle at the console, do the following:

Start the console:

```
java -jar org.eclipse.osgi_3.4.0.jar -console
```

Use the install command to deploy the bundle (you'll need to do this for the dependencies as well:

```
osgi> install file:///path/to/org.apache.openejb.server-1.0.2.jar
```

You can view the installed bundles using the ss command:

osgi> ss Framework is launched. id State Bundle 0 ACTIVE org.eclipse.osgi_3.4.0.v20080605-1900 1 ACTIVE org.apache.openejb.server_1.0.2 2 <<LAZY>> org.eclipse.core.contenttype_3.3.0.v20080604-1400 3 ACTIVE org.eclipse.core.jobs_3.4.0.v20080512 4 ACTIVE org.eclipse.core.runtime_3.4.0.v20080512 5 ACTIVE org.eclipse.core.variables_3.2.100.v20080529-1300 6 ACTIVE org.eclipse.equinox.app_1.1.0.v20080421-2006 7 ACTIVE org.eclipse.equinox.common_3.4.0.v20080421-2006 8 ACTIVE org.eclipse.equinox.preferences_3.2.201.R34x_v20080709 9 ACTIVE org.eclipse.equinox.registry_3.4.0.v20080516-0950 10 ACTIVE test.ejb_1.0.0 11 <<LAZY>> test.ejb.client_1.0.0 12 RESOLVED org.eclipse.equinox.launcher_1.0.101.R34x_v20080819

To create a directory structure instead, add the bundles you want to deploy into a directory, and add a config.ini file to a configuration subdirectory:

```
|-configuration
| |-config.ini
|
|-org.eclipse.core.contenttype.jar
|-org.eclipse.core.jobs.jar
|-org.eclipse.core.runtime.jar
|-org.eclipse.core.variables.jar
|-org.eclipse.equinox.app.jar
|-org.eclipse.equinox.common.jar
|-org.eclipse.equinox.launcher.jar
|-org.eclipse.equinox.preferences.jar
|-org.eclipse.equinox.registry.jar
|-org.eclipse.osgi.jar
```

config.ini needs to specify a osgi.bundles property in Java properties format (bundles with @start specified after them will be autostarted):

osgi.bundles=org.apache.openejb.server,org.eclipse.core.contenttype,org.eclipse.core.jobs,org.eclipse.core.runtime@start,...

Developing a RCP / OSGi application with OpenEJB

Knowing how to structure a RCP / OSGi application can be quite difficult. For an application using OpenEJB, I'd generally recommend starting off by spliting your solution into two projects. One bundle will contain the beans and a small BundleActivator to register an OpenEJBApplication service. The other bundle will be the EJB client. Regardless of whether you're creating a RCP app or just using Equinox, the instructions for the bundle containing your EJBs is the same. The client bundle, however, will be different if you're using RCP as you'll no doubt want to take advantage of the UI features Eclipse provides, and create an application, perspective and the view etc. that you want to provide.

This is the structure I've used in the both examples I've created, and is detailed below.

Creating an EJB bundle

You will need to create an OSGi bundle to hold your EJBs. This could be an existing EJB jar, which you can add a manifest file to. This bundle will need to declare a dependency on org.apache.openejb.server in the manifest (META-INF/MANIFEST.MF):

```
... Require-Bundle: org.apache.openejb.server;bundle-version="1.0.2"
Eclipse-RegisterBuddy: org.apache.openejb.server ...
```

The Eclipse-RegisterBuddy appears to be necessary to avoid a ClassCastException error when running in Equinox outside Eclipse.

Finally in the bundle Activator class, you will need to register an OSGi OpenEJBApplication service:

```
Bundle bundle = context.getBundle();
OpenEjbApplication application = new OpenEjbApplication(bundle);
context.registerService(OpenEjbApplication.class.getName(), application, null);
```

When your new bundle is deployed, you should see OpenEJB deploy your EJBs:

```
osgi>
Apache OpenEJB 3.1.1-SNAPSHOT build: 20090424-12:35
http://openejb.apache.org/
INFO - openejb.home = C:\Documents and Settings\Jon\openejb
INFO - openejb.base = D:\Temp\openejb-
osgi\configuration\org.eclipse.osgi\bundles\1\1\.cp
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager,
type=TransactionManager, provider-id=Default Transaction Manager)
INFO - Configuring enterprise application: D:\Temp\openejb-
osgi\configuration\org.eclipse.osgi\bundles\10\1\.cp
INFO - Configuring Service(id=Default Stateless Container, type=Container,
provider-id=Default Stateless Container)
```

```
INFO - Auto-creating a container for bean EchoImpl: Container(type=STATELESS,
id=Default Stateless Container)
INFO - Enterprise application "D:\Temp\openejb-
osgi\configuration\org.eclipse.osgi\bundles\10\1\.cp" loaded.
INFO - Assembling app: D:\Temp\openejb-
osgi\configuration\org.eclipse.osgi\bundles\10\1\.cp
INFO - Jndi(name=EchoImplRemote) --> Ejb(deployment-id=EchoImpl)
INFO - Jndi(name=EchoImplRemote) --> Ejb(deployment-id=EchoImpl)
INFO - Created Ejb(deployment-id=EchoImpl, ejb-name=EchoImpl, container=Default
Stateless Container)
INFO - Deployed Application(path=D:\Temp\openejb-
osgi\configuration\org.eclipse.osgi\bundles\10\1\.cp)
```

Creating a client bundle (simple OSGi)

You can create a bundle to access your beans. This bundle will need to declare dependencies on both the bundle with your EJBs (which will need to export your bean interfaces) and also on org.apache.openejb.server in META-INF/MANIFEST.MF:

```
... Require-Bundle: test.ejb;bundle-version="1.0.0",
org.apache.openejb.server;bundle-version="1.0.2" ...
```

You can then access your beans in the usual way:

```
Properties p = new Properties();
p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.client.LocalInitialContextFactory");
InitialContext initialContext = new InitialContext(p);
Echo server = (Echo) initialContext.lookup("EchoImplRemote");
server.echo("Hello world!");
```

Creating a client bundle (Eclipse plugin)

This is a similar process to creating a simpe OSGi bundle, but is a bit more involved. To get started, using the RCPapplication wizard in Eclipse is recommended.

- Create a new project
- Select Plugin project
- Give your project a name, and make sure the plugin is set to target Eclipse instead of Equinox.
- Enter data to create the plugin. Select yes, to create a RCP application.
- Select one of the RCP application templates
- Fillin any final details

Again, you will need to add a dependency to the EJB bundle you created earlier and to org.apache.openejb.server, by adding something similar to this to META-INF/MANIFEST.MF:

```
... Require-Bundle: test.ejb;bundle-version="1.0.0",
org.apache.openejb.server;bundle-version="1.0.2" ...
```

You can access your beans by performing a lookup wherever you need to use them, in the usual way:

```
Properties p = new Properties();
p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.client.LocalInitialContextFactory");
InitialContext initialContext = new InitialContext(p);
Echo server = (Echo) initialContext.lookup("EchoImplRemote");
server.echo("Hello world!");
```

You can run your application by right click on the plugin, and selecting 'Run as' → 'Eclipse Application'.