Application discovery via the classpath

This document details the various ways to get OpenEJB to pickup applications you'd like deployed while in an embedded mode.

Empty ejb-jar.xml approach (recommended)

Simplify the issue of searching for annotated applications by adding an ejb-jar.xml like this to your app:

"META-INF/ejb-jar.xml

```
<ejb-jar/>
```

OpenEJB will find the app in the classpath and deploy it along with any annotated beans it may contain.

The ejb-jar.xml can contain more than just "<ejb-jar/>" as usual.

This is the recommended approach for people using OpenEJB for unit testing as it allows OpenEJB to find your application in the classpath without the need for you to specify any path information which tends to complicate builds.

Including/Excluding paths (advanced)

If you do not like the idea of having the ejb-jar.xml in your app or an openejb.xml, we can search the classpath for annotated beans (@Stateless, @Stateful, @MessageDriven) and load them automatically just as if they contained an ejb-jar.xml.

This form of searching, however, is very expensive as it involves iterating over every path in the classpath and reading in each class definition contained thereunder and checking it for annotations.

This approach can only be made faster by helping us trim down or pinpoint the paths we should search via the **openejb.deployments.classpath.include** property which can be specified as a *system property* or a property passed into the *InitialContext*.

The value of this property is a regular expression and therefore can be absolute or relative. For example the path "/Users/dblevins/work/swizzle/swizzle-stream/target/classes" which contains the class files of an application you wish to test could be included in any of the following values to the "openejb. deployments.classpath.include" property:

- "file:///Users/dblevins/work/swizzle/swizzle-stream/target/classes/" *(an absolute path)*
- "file:///Users/dblevins/work/swizzle/.*" *(relative)*
- ".*swizzle-stream.*" *(very relative)*
- ".*(swizzle-stream|swizzle-jira|acme-rocket-app).*" *(including several paths)*

Note the filtering is done on URLs in the classpath, so forward slashes should always be used even on OSs using backslash ("\").

There is an **openejb.deployments.classpath.exclude** property if you wish to work in the opposite direction. The default values for both properties are as follows:

openejb.deployments.classpath.include="" *//include nothing*
openejb.deployments.classpath.exclude=".*" *//exclude everything*

The exclude and the include is applied separately and the results of each are combined together to create the list of paths OpenEJB will scrape for annotations.

Note by default these settings will only affect which jars OpenEJB will scan for annotated components when no descriptor is found. If you would like to use these settings to also filter out jars that do contain descriptors, set the **openejb.deployments.classpath.filter.descriptors** property to *true*. The default is *f alse*.

Troubleshooting

If you're having trouble determining if the META-INF/ejb-jar.xml file for your ejb module is in the classpath, a little debug code like this in your test setup will help you see what OpenEJB sees (which may be nothing):

```
Enumeration<URL> ejbJars = this.getClass().getClassLoader().getResources("META-INF/ejb-jar.xml");
while (ejbJars.hasMoreElements()) {
    URL url = ejbJars.nextElement();
    System.out.println("app = " + url);
}
```