Embedded and Remotable

Overview

This example shows how to use OpenEJB3's remoting capabilities in an embedded scenario. By remoting we mean that you wish to allow **clients in other vms** access your ejbs. *Note, you do not need to go to this extreme to unit test ejbs with remote interfaces.*

The basic recipe is the same for a standard embedded scenario but with these added ingreditents:

- **openejb.embedded.remotable** property
- **openejb-ejbd** jar

While creating the InitialContext, pass in the openejb.embedded.remotable property with the value of "true". When this is seen by the LocalInitialContextFactory, it will boot up the Server ServiceManager in the VM which will in turn look for ServerServices in the classpath.

Provided you have the openejb-ejbd jar in your classpath along with it's dependencies (openejb-server, openejb-client, openejb-core), then those services will be brought online and remote clients will be able to connect into your vm and invoke beans.

If you want to add more ServerServices such as the http version of the ejbd protocol you'd simply add the openejb-httpejbd jar to your classpath. A number of ServerServices are available currently:

- openejb-ejbd
- openejb-http
- openejb-telnet
- openejb-derbynet
- openejb-hsql
- openejb-activemq

The source for this example is in the "telephone-stateful" directory located in the openejb-examples.zip available on the download page.

If your goal is simply to unit test beans with remote interfaces, this is **not** the right example for you. The LocalInitialContextFactory completely supports remote interfaces and all spec required pass-by-value (serialization) semantics without the need for network sockets. This example shows the use of OpenEJB in an embedded environment where connection **outside** the
vm is required.

The Code

For this example we have a simple Stateful bean called TelephoneBean as defined below. As a simple way of demonstrating the state we have to methods: speak and listen. You call *speak* and pass in some text, then you call *listen* to get your answer.

bean

{snippet:id=code|url=openejb3/examples/telephone-stateful/src/main/java/org/superbiz/telephone/TelephoneBean.java|lang=java}

business interface

{snippet:id=code|url=openejb3/examples/telephone-stateful/src/main/java/org/superbiz/telephone/Telephone.java|lang=java} EJB3 Notes
The bean class uses the annotation **@Remote** but does not specify a list of interfaces as is normally required. Per EJB3 rules, if the bean implements exactly **one business interface** it may use @Remote with no other values and that business interface is then implied to be a remote business interface. The same rule applies to identical usage of @Local.

The critical thing to know is that if you add another interface the rules change and require that you specify both interfaces in the @Remote annotation as in @Remote({Telephone.class, SecondInterface.class}).

Embedding

We're going to embed OpenEJB3 into a plain JUnit TestCase as a simple means of demonstrating the remote capabilities. We'll do the embedding in our test setUp method, then will make two test methods:

- one for invoking the bean's remote interface via the **LocalInitialContextFactory** which goes straight against the embedded container system
- one for invoking the bean's remote interface via the **RemoteInitialContextFactory** which connects to a Socket and communicates to the embedded container system over the ejbd protocol.

setUp

{snippet:id=setup|url=openejb3/examples/telephone-stateful/src/test/java/org/superbiz/telephone/TelephoneTest.java|lang=java}

LocalInitialContextFactory: making in-vm calls to a remote business interface

{snippet:id=localcontext|url=openejb3/examples/telephone-stateful/src/test/java/org/superbiz/telephone/TelephoneTest.java|lang=java}

RemoteInitialContextFactory: making networked calls to a remote business interface

This is the part you would want to do in apps that are running a different VM than the one in which the ejb container is embedded. These "client" VMs need only have the the **openejb-client jar** in their classpath and connect to OpenEJB via the RemoteInitialContextFactory like any other remote EJB client.

{snippet:id=remotecontext|url=openejb3/examples/telephone-stateful/src/test/java/org/superbiz/telephone/TelephoneTest.java|lang=java}

Maven setup

{snippet:id=desc|url=openejb3/examples/telephone-stateful/pom.xml} {snippet:id=openejbdep|url=openejb3/examples/telephone-stateful/pom.xml|lang=xml}

Running

Running the example is fairly simple. In the "telephone-stateful" directory of the examples zip, just run:

```
$ mvn clean install
```

Which should create output like the following.

```
------------------------------------------------------- T E S T S ------------------------------------------------------- Running org.superbiz.telephone.TelephoneTest Apache
OpenEJB 3.0 build: 20080408-04:13 http://openejb.apache.org/ INFO - openejb.home = /Users/dblevins/work/openejb-3.0/examples/telephone-stateful
INFO - openejb.base = /Users/dblevins/work/openejb-3.0/examples/telephone-stateful 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 Service(id=Default JDK 1.3 ProxyFactory, type=ProxyFactory, provider-id=Default JDK 1.3
ProxyFactory) INFO - Found EjbModule in classpath: /Users/dblevins/work/openejb-3.0/examples/telephone-stateful/target/classes INFO - Configuring
app: /Users/dblevins/work/openejb-3.0/examples/telephone-stateful/target/classes INFO - Configuring Service(id=Default Stateful Container,
type=Container, provider-id=Default Stateful Container) INFO - Auto-creating a container for bean TelephoneBean: Container(type=STATEFUL, id=Default
Stateful Container) INFO - Loaded Module: /Users/dblevins/work/openejb-3.0/examples/telephone-stateful/target/classes INFO - Assembling app: /Users
/dblevins/work/openejb-3.0/examples/telephone-stateful/target/classes INFO - Jndi(name=TelephoneBeanRemote) --> Ejb(deployment-
id=TelephoneBean) INFO - Created Ejb(deployment-id=TelephoneBean, ejb-name=TelephoneBean, container=Default Stateful Container) INFO -
Deployed Application(path=/Users/dblevins/work/openejb-3.0/examples/telephone-stateful/target/classes) ** Starting Services ** NAME IP PORT ejbd
127.0.0.1 4201 admin thread 127.0.0.1 4200 ------- Ready! Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.89 sec Results : Tests run: 2,
Failures: 0, Errors: 0, Skipped: 0
```