TomEE and Arquillian

TomEE has several arquillian adapter flavors:

- openejb-embedded: a plain embedded OpenEJB supporting most of EE features
- tomee-embedded: a full TomEE running in the same JVM
- tomee-remote: a standard TomEE running in its own process as in production
- tomee-webapp (not recommanded): an adapter starting from a Tomcat and installing tomee-webapp

Embedded or Remote?

Big advantage of embedded adapters is to be able to debug as usual. However it has few drawbacks which can make you rething this choice:

- JVM resources are available where it will likely not be the case in war mode (src/main/resources typically)
- You can mix server and client side features when writing a test
- Classloading is a bit different by design and less isolated (test dependencies) so you can get runtime surprises when really deploying

To summarize: the choice is the trade off you choose between easiness and reality of the simulation.

TIP	in TomEE build we build the same tests against all tomee adapters in the same build/module, this means you can use embedded adapter in dev and activate remote tomee too (not only cause then if there is a failure you don't know if you missed it locally or if it is due to the switch of adapter) on your continuous integration platform.
NOTE	all configurations have defaults

OpenEJB Embedded

Coordinates

```
<dependency>
  <groupId>org.apache.tomee</groupId>
  <artifactId>arquillian-openejb-embedded</artifactId>
  <version>${tomee7.version}
</dependency>
```

arquillian.xml

Name	Description

properties	container properties, as in conf/system.properties (not in xml format)
preloadClasses	some class to load (ie enforce static block initialization)
startDefaultScopes	should CDI default scopes be started (includes @RequestScoped)
singleDeploymentByArchiveName	names of archives (or true for all) to dploy a single time

Sample:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<arquillian
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://jboss.org/schema/arquillian
http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
  <container qualifier="openejb" default="true">
    <configuration>
      <property name="properties">
        # used to not have a single DataSource and be able to test the resource
resolution
        db1 = new://Resource?type=DataSource
        db1.JdbcUrl = jdbc:hsqldb:mem:db1

        # will preload both classes, simple comma separated qualified names work too
        openejb.arquillian.predeploy-archives =
org.company.openejb.arquillian.openejb.archive.[SimpleArchive|SimpleArchive2]
      </property>
    </configuration>
  </container>
</arquillian>
```

TomEE Embedded

Coordinates

```
<dependency>
  <groupId>org.apache.tomee</groupId>
  <artifactId>arquillian-tomee-embedded</artifactId>
  <version>${tomee7.version}
</dependency>
```

Configuration

Name	Description

exportConfAsSystemProperty	export system properties with adapter prefix(es) (ex: httpPort will be set as tomee.httpPort)
httpsPort	the HTTPS port
httpPort	the HTTP port
stopPort	the shutdown port
dir	where to create the TomEE work dir (a fake file layout is created for Tomcat needs)
appWorkingDir	where to dump applications (`@Deployment`)
host	which host to use
stopHost	which port to use to shutdown TomEE (port on Server configuration)
stopCommand	which command to use to shutdown TomEE
serverXml	where is the provided server.xml
portRange	when port are set to -1 TomEE adapter will generate an available port, if specified the range will be used
preloadClasses	which classes to initialize during container startup
quickSession	should the session use a Random instead of SecureRandom (useful when the machine doesn't have a lot of entropy)
unsafeEjbd	should EJB allow all classes
unpackWars	unpackWARs value in server.xml
properties	container properties
webContextToUseWithEars	sometimes you can need this to adjust which context the adapter uses to find the ArquillianServletRunner
keepServerXmlAsThis	don't replace ports etc in server.xml and use it like it has been provided when serverXml is set
singleDumpByArchiveName	dump only once `@Deployment` archives using the name as key
singleDeploymentByArchiveName	deploy only once `@Deployment` archives using the name as key
ssl	should https be activated
withEjbRemote	should EJBd remote be activated
keystoreFile	if ssl is set to true the keystore location
keystorePass	if ssl is set to true the keystore password

keystoreType	if ssl is set to true the keystore type
clientAuth	should SSL connector use clientAuth
keyAlias	if ssl is set to true the key to use
sslProtocol	if ssl is set to true the protocol to use
users	a map of users (properties syntax)
roles	user roles (properties syntax)
webResourcesCached	should resources be cached or not (`DefaultServlet` caching)

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<arquillian
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  http://jboss.org/schema/arquillian
  http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
 <container qualifier="tomee" default="true">
  <configuration>
   <property name="serverXml">conf/server.xml</property>

   <!-- port = -1 means random -->
   <property name="httpPort">-1</property>
   <property name="stopPort">-1</property>

   <!-- ssl -->
   <property name="httpsPort">-1</property>
   <property name="ssl">false</property>
   <property name="keystoreFile">keystore-path</property>
   <property name="keystorePass">changeit</property>
   <property name="keystoreType">JKS</property>
   <property name="clientAuth">false</property>
   <property name="keyAlias">alias</property>
   <property name="sslProtocol">protocol</property>

   <!-- where to create TomEE files -->
   <property name="dir">target/tomee-embedded</property>

   <!-- where to dump on disk applications to deploy -->
   <property name="appWorkingDir">target/working-dir</property>

   <!-- optional - limit the port allowed when random -->
   <property name="portRange">20001-30000</property>

   <!-- container config -->
   <property name="properties">
    # same as embedded case
   </property>

   <!-- Deployer config -->
   <property name="deployerProperties">
    # openejb.deployer.binaries.use=true
    # openejb.deployer.forced.appId=[name]
    # openejb.deployer.save-deployments=false
   </property>
  </configuration>
 </container>
</arquillian>
```

TomEE Remote

IMPORTANT	if a server is already started on host:port then it will be used instead of starting the configured TomEE type.

To use a custom instance with arquillian ensure to have ejbd and tomee webapp activated. A way is to have in `conf/system.properties` these entries:

```
tomee.remote.support=true
openejb.system.apps=true

# you can customize it depending the security level you need on the instance
tomee.serialization.class.whitelist =
tomee.serialization.class.blacklist =
org.codehaus.groovy.runtime.,org.apache.commons.collections.functors.,org.apache.xalan
,java.lang.Process
```

For really remote instances (= not on localhost) you need the `deployerProperties` of previous snippet too:

```
<?xml version="1.0" encoding="UTF-8"?>
<arquillian
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  http://jboss.org/schema/arquillian
  http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
 <container qualifier="tomee" default="true">
  <configuration>
   <!-- ... -->
   <property name="deployerProperties">
    openejb.deployer.binaries.use=true
    openejb.deployer.save-deployments=false
   </property>
  </configuration>
 </container>
</arquillian>
```

Coordinates

```
<dependency>
  <groupId>org.apache.tomee</groupId>
  <artifactId>arquillian-tomee-remote</artifactId>
  <version>${tomee7.version}
</dependency>
```

Configuration

Name	Description
exportConfAsSystemProperty	export system properties with adapter prefix(es) (ex: httpPort will be set as tomee.httpPort)
httpsPort	the HTTPS port
httpPort	the HTTP port
stopPort	the shutdown port
dir	where to create the TomEE work dir (a fake file layout is created for Tomcat needs)
appWorkingDir	where to dump applications (`@Deployment`)
host	which host to use
stopHost	which port to use to shutdown TomEE (port on Server configuration)
stopCommand	which command to use to shutdown TomEE
serverXml	where is the provided server.xml
portRange	when port are set to -1 TomEE adapter will generate an available port, if specified the range will be used
preloadClasses	which classes to initialize during container startup
quickSession	should the session use a Random instead of SecureRandom (useful when the machine doesn't have a lot of entropy)
unsafeEjbd	should EJB allow all classes
unpackWars	unpackWARs value in server.xml
properties	container properties
webContextToUseWithEars	sometimes you can need this to adjust which context the adapter uses to find the ArquillianServletRunner
keepServerXmlAsThis	don't replace ports etc in server.xml and use it like it has been provided when serverXml is set
singleDumpByArchiveName	dump only once `@Deployment` archives using the name as key
singleDeploymentByArchiveName	deploy only once `@Deployment` archives using the name as key
groupId	the maven groupId of the TomEE (or not) artifact

artifactId	the maven artifactId of the TomEE (or not) artifact
version	the maven version of the TomEE (or not) artifact
classifier	the maven classifier of the TomEE (or not) artifact
type	the maven type of the TomEE (or not) artifact (should be zip)
removeUnusedWebapps	should default webapps (ROOT, manager, ...) be removed
ajpPort	the ajp port if used
conf	a folder to synchronize with TomEE conf folder
bin	a folder to synchronize with TomEE bin folder
lib	a folder to synchronize with TomEE lib folder
endorsed	a folder to synchronize with TomEE endorsed folder
javaagent	a list (flat format) of javaagent to add when launching tomee, can use maven coordinates if prefixed with `mvn:`
additionalLibs	a list (flat format) of library to add to TomEE libraries, can use paths of maven coordinates when prefixed with `mvn:`
cleanOnStartUp	should TomEE folder be deleted on startup if exists
debug	should the container run in debug mode (`-Dopenejb.server.debug=true` activates it without touching the configuration)
debugPort	if activated which port to use to debug
catalina_opts	equivalent to `CATALINA_OPTS` environment variable
simple_log	should logs be inline
deployerProperties	deployer properties, useful when not deploying on an instance managed by the build (remote instance typically)

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<arquillian
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  http://jboss.org/schema/arquillian
```

```
  http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
 <container qualifier="tomee" default="true">
  <configuration>
   <property name="serverXml">conf/server.xml</property>

   <!-- tomee zip to use -->
   <property name="groupId">org.apache.tomee</property>
   <property name="artifactId">apache-tomee</property>
   <property name="version">LATEST</property>
   <property name="type">zip</property>

   <!-- tomee provided files, ignored by default -->
   <property name="bin">src/test/tomee/bin</property>
   <property name="conf">src/test/tomee/conf</property>
   <property name="lib">src/test/tomee/lib</property>

   <!--
    remote debugging,
    -Dopenejb.server.debug can activate it too
   -->
   <property name="debug">false</property>
   <property name="debugPort">5005</property>

   <!-- nice one line logging -->
   <property name="simpleLog">true</property>

   <!-- jvm config -->
   <property name="catalina_opts">-XX:-UseParallelGC</property>

   <!-- remove if exist -->
   <property name="cleanOnStartUp">true</property>

   <!-- remove default webapps -->
   <property name="removeunusedWebapps">true</property>

   <!-- port = -1 means random -->
   <property name="httpPort">-1</property>
   <property name="stopPort">-1</property>

   <!-- where to create TomEE -->
   <property name="dir">target/apache-tomee</property>

   <!-- where to dump on disk applications to deploy -->
   <property name="appWorkingDir">target/working-dir</property>

   <!-- optional - limit the port allowed when random -->
   <property name="portRange">20001-30000</property>

   <!-- container config -->
   <property name="properties">
    # same as embedded case
```

```
    </property>

    <!-- we monitor the test with sirona -->
    <property name="javaagent">
      mvn:org.apache.sirona:sirona-javaagent:0.2-incubating:jar:shaded
    </property>

    <!-- Deployer config -->
    <property name="deployerProperties">
     # openejb.deployer.binaries.use=true
     # openejb.deployer.forced.appId=[name]
     # openejb.deployer.save-deployments=false
    </property>

  </configuration>
 </container>
</arquillian>
```

Multiple instances

With arquillian you can create cluster or isolated instances. Here is a sample `arquillian.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<arquillian xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://jboss.org/schema/arquillian
    http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
 <group qualifier="tomee-cluster">
  <container qualifier="tomee-1">
   <configuration>
    <property name="httpPort">-1</property>
    <property name="stopPort">-1</property>
    <property name="ajpPort">-1</property>
    <property name="dir">target/tomee1</property>
    <property name="appWorkingDir">target/wd1</property>
   </configuration>
  </container>
  <container qualifier="tomee-2">
   <configuration>
    <property name="httpPort">-1</property>
    <property name="stopPort">-1</property>
    <property name="ajpPort">-1</property>
    <property name="dir">target/tomee2</property>
    <property name="appWorkingDir">target/wd2</property>
   </configuration>
  </container>
 </group>
</arquillian>
```

Then in your test just specify the container you are testing against:

```
@RunWith(Arquillian.class)
public class MultipleTomEETest {
 @Deployment(name = "war1", testable = false)
 @TargetsContainer("tomee-1")
 public static WebArchive war1() {
  return /* ... */;
 }

 @Deployment(name = "war2", testable = false)
 @TargetsContainer("tomee-2")
 public static WebArchive war2() {
  return /* ... */;
 }

 @Test
 @OperateOnDeployment("war1")
 public void testRunningInDep1(
    @ArquillianResource URL url) {
   // test on tomee 1, url is contextual
 }

 @Test
 @OperateOnDeployment("war2")
 public void testRunningInDep1(
    @ArquillianResource URL url) {
   // test on tomee 1, url is contextual
 }
}
```