Application Composer

Example application-composer can be browsed at https://github.com/apache/tomee/tree/master/examples/application-composer

The `org.apache.openejb.junit.ApplicationComposer` is JUnit test runner modeled after the way we've done testing internally in OpenEJB for years (since about 2006). It involves no classpath scanning at all. If you want something to be in the app, you must build it directly in your testcase.

With the `ApplicationComposer` you can do identical testing that OpenEJB uses internally, but with limited dependency on OpenEJB itself. The main dependency is on the code that is used to build the actual applications:

```
<dependency>
  <groupId>org.apache.openejb</groupId>
  <artifactId>openejb-jee</artifactId>
  <version>4.0.0-beta-1</version>
</dependency>
```

Composing an Application

The main difference to the embedded `EJBContainer` API is building the application in the test code. This is done with one or more methods in the test case annotated with `org.apache.openejb.junit.Module` using the following format:

```
@Module
public <return-value> <module-name>() {

Where **module-name** is the name you wish to use for that module and **return-value**
can be any one of the following:

 - java.lang.Class
 - java.lang.Class[]
 - org.apache.openejb.jee.EjbJar
 - org.apache.openejb.jee.EnterpriseBean
 - org.apache.openejb.jee.Application
 - org.apache.openejb.jee.Connector
 - org.apache.openejb.jee.Beans
 - org.apache.openejb.jee.jpa.unit.Persistence
 - org.apache.openejb.jee.jpa.unit.PersistenceUnit

# Example

Used in an actual testcase, that might look like so:

import junit.framework.TestCase;
```

```
import org.apache.openejb.jee.EjbJar;
import org.apache.openejb.jee.StatefulBean;
import org.apache.openejb.jee.jpa.unit.PersistenceUnit;
import org.apache.openejb.junit.ApplicationComposer;
import org.apache.openejb.junit.Configuration;
import org.apache.openejb.junit.Module;
import org.junit.Test;
import org.junit.runner.RunWith;

import javax.annotation.Resource;
import javax.ejb.EJB;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.transaction.UserTransaction;
import java.util.List;
import java.util.Properties;

@RunWith(ApplicationComposer.class)
public class MoviesTest extends TestCase {

    @EJB
    private Movies movies;

    @Resource
    private UserTransaction userTransaction;

    @PersistenceContext
    private EntityManager entityManager;

    @Module
    public PersistenceUnit persistence() {
        PersistenceUnit unit = new PersistenceUnit("movie-unit");
        unit.setJtaDataSource("movieDatabase");
        unit.setNonJtaDataSource("movieDatabaseUnmanaged");
        unit.getClazz().add(Movie.class.getName());
        unit.setProperty("openjpa.jdbc.SynchronizeMappings",
"buildSchema(ForeignKeys=true)");
        return unit;
    }

    @Module
    public EjbJar beans() {
        EjbJar ejbJar = new EjbJar("movie-beans");
        ejbJar.addEnterpriseBean(new StatefulBean(MoviesImpl.class));
        return ejbJar;
    }

    @Configuration
    public Properties config() throws Exception {
        Properties p = new Properties();
        p.put("movieDatabase", "new://Resource?type=DataSource");
```

```
        p.put("movieDatabase.JdbcDriver", "org.hsqldb.jdbcDriver");
        p.put("movieDatabase.JdbcUrl", "jdbc:hsqldb:mem:moviedb");
        return p;
    }

    @Test
    public void test() throws Exception {

        userTransaction.begin();

        try {
            entityManager.persist(new Movie("Quentin Tarantino", "Reservoir Dogs",
1992));
            entityManager.persist(new Movie("Joel Coen", "Fargo", 1996));
            entityManager.persist(new Movie("Joel Coen", "The Big Lebowski", 1998));

            List<Movie> list = movies.getMovies();
            assertEquals("List.size()", 3, list.size());

            for (Movie movie : list) {
                movies.deleteMovie(movie);
            }

            assertEquals("Movies.getMovies()", 0, movies.getMovies().size());

        } finally {
            userTransaction.commit();
        }
    }
}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.composed.MoviesTest
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=movieDatabase, type=Resource, provider-id=Default JDBC
Database)
INFO - Configuring enterprise application: /Users/dblevins/examples/application-
composer/MoviesTest
WARN - Method 'lookup' is not available for 'javax.annotation.Resource'. Probably
using an older Runtime.
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
```

```
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.composed.MoviesTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Configuring Service(id=Default Stateful Container, type=Container, provider-
id=Default Stateful Container)
INFO - Auto-creating a container for bean MoviesImpl: Container(type=STATEFUL,
id=Default Stateful Container)
INFO - Configuring PersistenceUnit(name=movie-unit)
INFO - Auto-creating a Resource with id 'movieDatabaseNonJta' of type 'DataSource for
'movie-unit'.
INFO - Configuring Service(id=movieDatabaseNonJta, type=Resource, provider-
id=movieDatabase)
INFO - Adjusting PersistenceUnit movie-unit <non-jta-data-source> to Resource ID
'movieDatabaseNonJta' from 'movieDatabaseUnmanaged'
INFO - Enterprise application "/Users/dblevins/examples/application-
composer/MoviesTest" loaded.
INFO - Assembling app: /Users/dblevins/examples/application-composer/MoviesTest
INFO - PersistenceUnit(name=movie-unit,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 449ms
INFO - Jndi(name=org.superbiz.composed.MoviesTestLocalBean) --> Ejb(deployment-
id=org.superbiz.composed.MoviesTest)
INFO -
Jndi(name=global/MoviesTest/EjbModule2027711095/MoviesTest!org.superbiz.composed.Movie
sTest) --> Ejb(deployment-id=org.superbiz.composed.MoviesTest)
INFO - Jndi(name=global/MoviesTest/EjbModule2027711095/MoviesTest) --> Ejb(deployment-
id=org.superbiz.composed.MoviesTest)
INFO - Jndi(name=MoviesImplLocal) --> Ejb(deployment-id=MoviesImpl)
INFO - Jndi(name=global/MoviesTest/movie-
beans/MoviesImpl!org.superbiz.composed.Movies) --> Ejb(deployment-id=MoviesImpl)
INFO - Jndi(name=global/MoviesTest/movie-beans/MoviesImpl) --> Ejb(deployment-
id=MoviesImpl)
INFO - Created Ejb(deployment-id=org.superbiz.composed.MoviesTest, ejb-
name=MoviesTest, container=Default Managed Container)
INFO - Created Ejb(deployment-id=MoviesImpl, ejb-name=MoviesImpl, container=Default
Stateful Container)
INFO - Started Ejb(deployment-id=org.superbiz.composed.MoviesTest, ejb-
name=MoviesTest, container=Default Managed Container)
INFO - Started Ejb(deployment-id=MoviesImpl, ejb-name=MoviesImpl, container=Default
Stateful Container)
INFO - Deployed Application(path=/Users/dblevins/examples/application-
composer/MoviesTest)
INFO - Undeploying app: /Users/dblevins/examples/application-composer/MoviesTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.221 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```