EJB 2.1 CMP EntityBeans (CMP2)

Example simple-cmp2 can be browsed at https://github.com/apache/tomee/tree/master/examples/simple-cmp2

OpenEJB, the EJB Container for TomEE and Geronimo, does support all of EJB 1.1 to 3.1, including CMP2.

The CMP2 implementation is actually done by adapting the CMP2 bean into a JPA Entity dynamically at deploy time.

Appropriate subclasses, a JPA persistence.xml file and a mapping.xml file are generated at deployment time for the CMP2 EntityBeans and all the Entities will be then run on OpenJPA. This innovative code has been used as the sole CMP2 implementation in Geronimo for its J2EE 1.4, JavaEE 5 and JavaEE 6 certifications.

The persistence.xml and mapping.xml files generated at deploy time can be saved to disk and included in the application, allowing you to:

- gain finer control over persistence options
- slowly convert individual entities from CMP2 to JPA

Let's see an example.

Movies application

The following is a basic EJB 2.1 application consisting of one CMP2 Entity. For those that are reading this example out of curiosity and are not familiar with CMP2 or EJB 2.x, each CMP2 Entity is composed of two parts

- **A Home interface** which has data access methods like "find", "create", and "remove". This is essentially what people use `@Stateless` beans for today, but with difference that you do not need to supply the implementation of the interface — the container will generate one for you. This is partly what inspired the creation of the OpenEJB-specific Dynamic DAO feature.
- **An abstract EntityBean class** which declares the persistent "properties" of the entity without actually declaring any fields. It is the container's job to implement the actual methods and create the appropriate fields. OpenEJB will implement this bean as a JPA `@Entity` bean.

As such a CMP2 EntityBean is really just the description of a persistent object and the description of a data-access object. There is no actual code to write.

The majority of work in CMP2 is done in the xml:

- **ejb-jar.xml** mapping information, which describes the persistent properties of the entity and the queries for all **Home** find, create and remove methods. This information will be converted

by OpenEJB into a JPA mapping.xml file. All queries in the cmp2 part of the ejb-jar.xml are converted into named queries in JPA and generally everything is converted to its JPA equivalent.

CMP2 EntityBean, MovieBean

```
package org.superbiz.cmp2;

import javax.ejb.EntityBean;

public abstract class MovieBean implements EntityBean {

    public MovieBean() {
    }

    public Integer ejbCreate(String director, String title, int year) {
        this.setDirector(director);
        this.setTitle(title);
        this.setYear(year);
        return null;
    }

    public abstract java.lang.Integer getId();

    public abstract void setId(java.lang.Integer id);

    public abstract String getDirector();

    public abstract void setDirector(String director);

    public abstract String getTitle();

    public abstract void setTitle(String title);

    public abstract int getYear();

    public abstract void setYear(int year);

}
```

CMP2 Home interface, Movies

```
package org.superbiz.cmp2;

import javax.ejb.CreateException;
import javax.ejb.FinderException;
import java.util.Collection;

/**
 * @version $Revision$ $Date$
 */
interface Movies extends javax.ejb.EJBLocalHome {
    Movie create(String director, String title, int year) throws CreateException;

    Movie findByPrimaryKey(Integer primarykey) throws FinderException;

    Collection<Movie> findAll() throws FinderException;

    Collection<Movie> findByDirector(String director) throws FinderException;
}
```

CMP2 mapping in ejb-jar.xml

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>MovieBean</ejb-name>
      <local-home>org.superbiz.cmp2.Movies</local-home>
      <local>org.superbiz.cmp2.Movie</local>
      <ejb-class>org.superbiz.cmp2.MovieBean</ejb-class>
      <persistence-type>Container</persistence-type>
      <prim-key-class>java.lang.Integer</prim-key-class>
      <reentrant>false</reentrant>
      <cmp-version>2.x</cmp-version>
      <abstract-schema-name>MovieBean</abstract-schema-name>
      <cmp-field>
        <field-name>id</field-name>
      </cmp-field>
      <cmp-field>
        <field-name>director</field-name>
      </cmp-field>
      <cmp-field>
        <field-name>year</field-name>
      </cmp-field>
      <cmp-field>
        <field-name>title</field-name>
      </cmp-field>
      <primkey-field>id</primkey-field>
      <query>
        <query-method>
          <method-name>findByDirector</method-name>
          <method-params>
            <method-param>java.lang.String</method-param>
          </method-params>
        </query-method>
        <ejb-ql>SELECT m FROM MovieBean m WHERE m.director = ?1</ejb-ql>
      </query>
      <query>
        <query-method>
          <method-name>findAll</method-name>
          <method-params/>
        </query-method>
        <ejb-ql>SELECT m FROM MovieBean as m</ejb-ql>
      </query>
    </entity>
  </enterprise-beans>
</ejb-jar>
```

openejb-jar.xml

```
<openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1">
  <enterprise-beans>
    <entity>
      <ejb-name>MovieBean</ejb-name>
      <key-generator xmlns="http://www.openejb.org/xml/ns/pkgen-2.1">
        <uuid/>
      </key-generator>
    </entity>
  </enterprise-beans>
</openejb-jar>
```

MoviesTest

```
package org.superbiz.cmp2;

import junit.framework.TestCase;

import javax.naming.Context;
import javax.naming.InitialContext;
import java.util.Collection;
import java.util.Properties;

/**
 * @version $Revision: 607077 $ $Date: 2007-12-27 06:55:23 -0800 (Thu, 27 Dec 2007) $
 */
public class MoviesTest extends TestCase {

    public void test() throws Exception {
        Properties p = new Properties();
        p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.core.LocalInitialContextFactory");
        p.put("movieDatabase", "new://Resource?type=DataSource");
        p.put("movieDatabase.JdbcDriver", "org.hsqldb.jdbcDriver");
        p.put("movieDatabase.JdbcUrl", "jdbc:hsqldb:mem:moviedb");

        p.put("movieDatabaseUnmanaged", "new://Resource?type=DataSource");
        p.put("movieDatabaseUnmanaged.JdbcDriver", "org.hsqldb.jdbcDriver");
        p.put("movieDatabaseUnmanaged.JdbcUrl", "jdbc:hsqldb:mem:moviedb");
        p.put("movieDatabaseUnmanaged.JtaManaged", "false");

        Context context = new InitialContext(p);

        Movies movies = (Movies) context.lookup("MovieBeanLocalHome");

        movies.create("Quentin Tarantino", "Reservoir Dogs", 1992);
        movies.create("Joel Coen", "Fargo", 1996);
        movies.create("Joel Coen", "The Big Lebowski", 1998);

        Collection<Movie> list = movies.findAll();
        assertEquals("Collection.size()", 3, list.size());

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

        assertEquals("Movies.findAll()", 0, movies.findAll().size());
    }
}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.cmp2.MoviesTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/simple-cmp2/target
INFO - openejb.base = /Users/dblevins/examples/simple-cmp2/target
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=movieDatabaseUnmanaged, type=Resource, provider-
id=Default JDBC Database)
INFO - Configuring Service(id=movieDatabase, type=Resource, provider-id=Default JDBC
Database)
INFO - Found EjbModule in classpath: /Users/dblevins/examples/simple-
cmp2/target/classes
INFO - Beginning load: /Users/dblevins/examples/simple-cmp2/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/simple-
cmp2/target/classpath.ear
INFO - Configuring Service(id=Default CMP Container, type=Container, provider-
id=Default CMP Container)
INFO - Auto-creating a container for bean MovieBean: Container(type=CMP_ENTITY,
id=Default CMP Container)
INFO - Configuring PersistenceUnit(name=cmp)
INFO - Adjusting PersistenceUnit cmp <jta-data-source> to Resource ID 'movieDatabase'
from 'null'
INFO - Adjusting PersistenceUnit cmp <non-jta-data-source> to Resource ID
'movieDatabaseUnmanaged' from 'null'
INFO - Enterprise application "/Users/dblevins/examples/simple-
cmp2/target/classpath.ear" loaded.
INFO - Assembling app: /Users/dblevins/examples/simple-cmp2/target/classpath.ear
INFO - PersistenceUnit(name=cmp,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 160ms
INFO - Jndi(name=MovieBeanLocalHome) --> Ejb(deployment-id=MovieBean)
INFO - Jndi(name=global/classpath.ear/simple-cmp2/MovieBean!org.superbiz.cmp2.Movies)
--> Ejb(deployment-id=MovieBean)
INFO - Jndi(name=global/classpath.ear/simple-cmp2/MovieBean) --> Ejb(deployment-
id=MovieBean)
INFO - Created Ejb(deployment-id=MovieBean, ejb-name=MovieBean, container=Default CMP
Container)
INFO - Started Ejb(deployment-id=MovieBean, ejb-name=MovieBean, container=Default CMP
Container)
INFO - Deployed Application(path=/Users/dblevins/examples/simple-
cmp2/target/classpath.ear)
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.919 sec
```

```
Results :

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

CMP2 to JPA

As mentioned OpenEJB will implement the abstract CMP2 `EntityBean` as a JPA `@Entity`, create a `persistence.xml` file and convert all `ejb-jar.xml` mapping and queries to a JPA `entity-mappings.xml` file.

Both of these files will be written to disk by setting the system property `openejb.descriptors.output` to `true`. In the testcase above, this can be done via the `InitialContext` parameters via code like this:

```
Properties p = new Properties();
p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.core.LocalInitialContextFactory");
```

```
// setup the data sources as usual...
```

```
// write the generated descriptors
p.put("openejb.descriptors.output", "true");
```

```
Context context = new InitialContext(p);
```

Below are the generated `persistence.xml` and `mapping.xml` files for our CMP2 `EntityBean`

CMP2 to JPA generated persistence.xml file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="1.0">
    <persistence-unit name="cmp" transaction-type="JTA">
        <jta-data-source>movieDatabase</jta-data-source>
        <non-jta-data-source>movieDatabaseUnmanaged</non-jta-data-source>
        <mapping-file>META-INF/openejb-cmp-generated-orm.xml</mapping-file>
        <class>openejb.org.superbiz.cmp2.MovieBean</class>
        <properties>
            <property name="openjpa.jdbc.SynchronizeMappings"
            value="buildSchema(ForeignKeys=true, Indexes=false, IgnoreErrors=true)"/>
            <property name="openjpa.Log" value="DefaultLevel=INFO"/>
        </properties>
    </persistence-unit>
</persistence>
```

All of this `persitence.xml` can be changed, however the `persistence-unit` must have the `name` fixed to `cmp`.

CMP2 to JPA generated mapping file

Note that the `persistence.xml` above refers to this mappings file as `META-INF/openejb-cmp-generated-orm.xml`. It is possible to rename this file to whatever name you prefer, just make sure to update the `<mapping-file>` element of the `cmp` persistence unit accordingly.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm" version="1.0">
    <entity class="openejb.org.superbiz.cmp2.MovieBean" name="MovieBean">
        <description>simple-cmp2#MovieBean</description>
        <table/>
        <named-query name="MovieBean.findByDirector(java.lang.String)">
            <query>SELECT m FROM MovieBean m WHERE m.director = ?1</query>
        </named-query>
        <named-query name="MovieBean.findAll">
            <query>SELECT m FROM MovieBean as m</query>
        </named-query>
        <attributes>
            <id name="id">
                <generated-value strategy="IDENTITY"/>
            </id>
            <basic name="director"/>
            <basic name="year"/>
            <basic name="title"/>
        </attributes>
    </entity>
</entity-mappings>
```