Injection Of Entitymanager

Example injection-of-entitymanager can be browsed at https://github.com/apache/tomee/tree/master/examples/injection-of-entitymanager

This example shows use of `@PersistenceContext` to have an `EntityManager` with an `EXTENDED` persistence context injected into a `@Stateful bean`. A JPA `@Entity` bean is used with the `EntityManager` to create, persist and merge data to a database.

Creating the JPA Entity

The entity itself is simply a pojo annotated with `@Entity`. We create one called `Movie` which we can use to hold movie records.

```
package org.superbiz.injection.jpa;

import javax.persistence.Entity;

@Entity
public class Movie {

    @Id @GeneratedValue
    private long id;

    private String director;
    private String title;
    private int year;

    public Movie() {
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public Movie(String director, String title, int year) {
        this.director = director;
        this.title = title;
        this.year = year;
    }

    public String getDirector() {
        return director;
    }
```

```
    public void setDirector(String director) {
        this.director = director;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }
}
```

Configure the EntityManager via a persistence.xml file

The above `Movie` entity can be created, removed, updated or deleted via an `EntityManager` object. The `EntityManager` itself is configured via a `META-INF/persistence.xml` file that is placed in the same jar as the `Movie` entity.

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="1.0">

  <persistence-unit name="movie-unit">
    <jta-data-source>movieDatabase</jta-data-source>
    <non-jta-data-source>movieDatabaseUnmanaged</non-jta-data-source>
    <class>org.superbiz.injection.jpa.Movie</class>

    <properties>
      <property name="openjpa.jdbc.SynchronizeMappings" value=
"buildSchema(ForeignKeys=true)"/>
    </properties>
  </persistence-unit>
</persistence>
```

Notice that the `Movie` entity is listed via a `<class>` element. This is not required, but can help when testing or when the `Movie` class is located in a different jar than the jar containing the `persistence.xml` file.

Injection via @PersistenceContext

The `EntityManager` itself is created by the container using the information in the `persistence.xml`, so to use it at runtime, we simply need to request it be injected into one of our components. We do this via `@PersistenceContext`

The `@PersistenceContext` annotation can be used on any CDI bean, EJB, Servlet, Servlet Listener, Servlet Filter, or JSF ManagedBean. If you don't use an EJB you will need to use a `UserTransaction` begin and commit transactions manually. A transaction is required for any of the create, update or delete methods of the EntityManager to work.

```
package org.superbiz.injection.jpa;

import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.PersistenceContextType;
import javax.persistence.Query;
import java.util.List;

@Stateful
public class Movies {

    @PersistenceContext(unitName = "movie-unit", type = PersistenceContextType
.EXTENDED)
    private EntityManager entityManager;

    public void addMovie(Movie movie) throws Exception {
        entityManager.persist(movie);
    }

    public void deleteMovie(Movie movie) throws Exception {
        entityManager.remove(movie);
    }

    public List<Movie> getMovies() throws Exception {
        Query query = entityManager.createQuery("SELECT m from Movie as m");
        return query.getResultList();
    }
}
```

This particular `EntityManager` is injected as an `EXTENDED` persistence context, which simply means that the `EntityManager` is created when the `@Stateful` bean is created and destroyed when the `@Stateful` bean is destroyed. Simply put, the data in the `EntityManager` is cached for the lifetime of the `@Stateful` bean.

The use of `EXTENDED` persistence contexts is **only** available to `@Stateful` beans. See the JPA Concepts page for an high level explanation of what a "persistence context" really is and how it is significant

to JPA.

MoviesTest

Testing JPA is quite easy, we can simply use the `EJBContainer` API to create a container in our test case.

```
package org.superbiz.injection.jpa;

import junit.framework.TestCase;

import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;
import java.util.List;
import java.util.Properties;

//START SNIPPET: code
public class MoviesTest extends TestCase {

    public void test() throws Exception {

        final 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");

        final Context context = EJBContainer.createEJBContainer(p).getContext();

        Movies movies = (Movies) context.lookup("java:global/injection-of-
entitymanager/Movies");

        movies.addMovie(new Movie("Quentin Tarantino", "Reservoir Dogs", 1992));
        movies.addMovie(new Movie("Joel Coen", "Fargo", 1996));
        movies.addMovie(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());
    }
}
```

Running

When we run our test case we should see output similar to the following.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.injection.jpa.MoviesTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/injection-of-entitymanager
INFO - openejb.base = /Users/dblevins/examples/injection-of-entitymanager
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
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 - Found EjbModule in classpath: /Users/dblevins/examples/injection-of-
entitymanager/target/classes
INFO - Beginning load: /Users/dblevins/examples/injection-of-
entitymanager/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/injection-of-
entitymanager
INFO - Configuring Service(id=Default Stateful Container, type=Container, provider-
id=Default Stateful Container)
INFO - Auto-creating a container for bean Movies: Container(type=STATEFUL, id=Default
Stateful Container)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.injection.jpa.MoviesTest:
Container(type=MANAGED, id=Default Managed 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/injection-of-entitymanager"
loaded.
INFO - Assembling app: /Users/dblevins/examples/injection-of-entitymanager
INFO - PersistenceUnit(name=movie-unit,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 462ms
INFO - Jndi(name="java:global/injection-of-
entitymanager/Movies!org.superbiz.injection.jpa.Movies")
INFO - Jndi(name="java:global/injection-of-entitymanager/Movies")
INFO -
```

```
Jndi(name="java:global/EjbModule1461341140/org.superbiz.injection.jpa.MoviesTest!org.s
uperbiz.injection.jpa.MoviesTest")
INFO -
Jndi(name="java:global/EjbModule1461341140/org.superbiz.injection.jpa.MoviesTest")
INFO - Created Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Created Ejb(deployment-id=org.superbiz.injection.jpa.MoviesTest, ejb-
name=org.superbiz.injection.jpa.MoviesTest, container=Default Managed Container)
INFO - Started Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Started Ejb(deployment-id=org.superbiz.injection.jpa.MoviesTest, ejb-
name=org.superbiz.injection.jpa.MoviesTest, container=Default Managed Container)
INFO - Deployed Application(path=/Users/dblevins/examples/injection-of-entitymanager)
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.301 sec

Results :

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