Movies Complete Meta

Example movies-complete-meta can be browsed at https://github.com/apache/tomee/tree/master/examples/movies-complete-meta

Help us document this example! Click the blue pencil icon in the upper right to edit this page.

AddInterceptor

```
package org.superbiz.injection.tx;

import javax.interceptor.AroundInvoke;
import javax.interceptor.InvocationContext;

/**
 * @version $Revision$ $Date$
 */
public class AddInterceptor {

    @AroundInvoke
    public Object invoke(InvocationContext context) throws Exception {
        // Log Add
        return context.proceed();
    }
}
```

Add

```
package org.superbiz.injection.tx.api;

import org.superbiz.injection.tx.AddInterceptor;

import javax.annotation.security.RolesAllowed;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.interceptor.Interceptors;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Metatype
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)

public @interface Add {
    public interface $ {

        @Add
        @RolesAllowed({"Employee", "Manager"})
        @TransactionAttribute(TransactionAttributeType.REQUIRED)
        @Interceptors(AddInterceptor.class)
        public void method();
    }
}
```

Delete

```
package org.superbiz.injection.tx.api;

import org.superbiz.injection.tx.DeleteInterceptor;

import javax.annotation.security.RolesAllowed;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.interceptor.Interceptors;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Metatype
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)

public @interface Delete {
    public interface $ {

        @Delete
        @RolesAllowed({"Manager"})
        @TransactionAttribute(TransactionAttributeType.MANDATORY)
        @Interceptors(DeleteInterceptor.class)
        public void method();
    }
}
```

Metatype

```
package org.superbiz.injection.tx.api;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Metatype
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.ANNOTATION_TYPE)
public @interface Metatype {
}
```

MovieUnit

```
package org.superbiz.injection.tx.api;

import javax.persistence.PersistenceContext;
import javax.persistence.PersistenceContextType;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;


@Metatype
@Target({ElementType.METHOD, ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)

@PersistenceContext(name = "movie-unit", unitName = "movie-unit", type =
PersistenceContextType.EXTENDED)
public @interface MovieUnit {
}
```

Read

```
package org.superbiz.injection.tx.api;

import javax.annotation.security.PermitAll;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Metatype
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)

public @interface Read {
    public interface $ {

        @Read
        @PermitAll
        @TransactionAttribute(TransactionAttributeType.SUPPORTS)
        public void method();
    }
}
```

DeleteInterceptor

```
package org.superbiz.injection.tx;

import javax.interceptor.AroundInvoke;
import javax.interceptor.InvocationContext;

/**
 * @version $Revision$ $Date$
 */
public class DeleteInterceptor {

    @AroundInvoke
    public Object invoke(InvocationContext context) throws Exception {
        // Log Delete
        return context.proceed();
    }
}
```

Movie

```
package org.superbiz.injection.tx;

import javax.persistence.Entity;

@Entity
public class Movie {

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

    public Movie() {
    }

    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;
    }

}
```

Movies

```
package org.superbiz.injection.tx;

import org.superbiz.injection.tx.api.Add;
import org.superbiz.injection.tx.api.Delete;
import org.superbiz.injection.tx.api.MovieUnit;
import org.superbiz.injection.tx.api.Read;

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

//END SNIPPET: code

//START SNIPPET: code
@Stateful
public class Movies {

    @MovieUnit
    private EntityManager entityManager;

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

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

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

persistence.xml

```
<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.tx.Movie</class>

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

MoviesTest

```
package org.superbiz.injection.tx;

import junit.framework.TestCase;

import javax.annotation.security.RunAs;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.ejb.embeddable.EJBContainer;
import java.util.List;
import java.util.Properties;
import java.util.concurrent.Callable;

import static javax.ejb.TransactionAttributeType.REQUIRES_NEW;

/**
 * See the transaction-rollback example as it does the same thing
 * via UserTransaction and shows more techniques for rollback
 */
//START SNIPPET: code
public class MoviesTest extends TestCase {

    @EJB
    private Movies movies;

    @EJB(beanName = "TransactionBean")
    private Caller transactionalCaller;

    @EJB(beanName = "NoTransactionBean")
    private Caller nonTransactionalCaller;
```

```
    protected void setUp() 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");

        EJBContainer.createEJBContainer(p).getContext().bind("inject", this);
    }

    private void doWork() throws Exception {

        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());
    }

    public void testWithTransaction() throws Exception {
        transactionalCaller.call(new Callable() {
            public Object call() throws Exception {
                doWork();
                return null;
            }
        });
    }

    public void testWithoutTransaction() throws Exception {
        try {
            nonTransactionalCaller.call(new Callable() {
                public Object call() throws Exception {
                    doWork();
                    return null;
                }
            });
            fail("The Movies bean should be using TransactionAttributeType.MANDATORY"
);
        } catch (javax.ejb.EJBException e) {
            // good, our Movies bean is using TransactionAttributeType.MANDATORY as we
want
        }
    }
```

```
    public static interface Caller {
        public <V> V call(Callable<V> callable) throws Exception;
    }

    /**
     * This little bit of magic allows our test code to execute in
     * the scope of a container controlled transaction.
     */
    @Stateless
    @RunAs("Manager")
    @TransactionAttribute(REQUIRES_NEW)
    public static class TransactionBean implements Caller {

        public <V> V call(Callable<V> callable) throws Exception {
            return callable.call();
        }
    }

    @Stateless
    @RunAs("Manager")
    @TransactionAttribute(TransactionAttributeType.NEVER)
    public static class NoTransactionBean implements Caller {

        public <V> V call(Callable<V> callable) throws Exception {
            return callable.call();
        }
    }
}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.injection.tx.MoviesTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/movies-complete-meta
INFO - openejb.base = /Users/dblevins/examples/movies-complete-meta
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/movies-complete-
meta/target/test-classes
```

```
INFO - Found EjbModule in classpath: /Users/dblevins/examples/movies-complete-
meta/target/classes
INFO - Beginning load: /Users/dblevins/examples/movies-complete-meta/target/test-
classes
INFO - Beginning load: /Users/dblevins/examples/movies-complete-meta/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/movies-complete-
meta
INFO - Configuring Service(id=Default Stateless Container, type=Container, provider-
id=Default Stateless Container)
INFO - Auto-creating a container for bean TransactionBean: Container(type=STATELESS,
id=Default Stateless Container)
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.tx.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/movies-complete-meta" loaded.
INFO - Assembling app: /Users/dblevins/examples/movies-complete-meta
INFO - PersistenceUnit(name=movie-unit,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 408ms
INFO - Jndi(name="java:global/movies-complete-
meta/TransactionBean!org.superbiz.injection.tx.MoviesTest$Caller")
INFO - Jndi(name="java:global/movies-complete-meta/TransactionBean")
INFO - Jndi(name="java:global/movies-complete-
meta/NoTransactionBean!org.superbiz.injection.tx.MoviesTest$Caller")
INFO - Jndi(name="java:global/movies-complete-meta/NoTransactionBean")
INFO - Jndi(name="java:global/movies-complete-
meta/Movies!org.superbiz.injection.tx.Movies")
INFO - Jndi(name="java:global/movies-complete-meta/Movies")
INFO -
Jndi(name="java:global/EjbModule1861413442/org.superbiz.injection.tx.MoviesTest!org.su
perbiz.injection.tx.MoviesTest")
INFO -
Jndi(name="java:global/EjbModule1861413442/org.superbiz.injection.tx.MoviesTest")
INFO - Created Ejb(deployment-id=NoTransactionBean, ejb-name=NoTransactionBean,
container=Default Stateless Container)
INFO - Created Ejb(deployment-id=TransactionBean, ejb-name=TransactionBean,
container=Default Stateless Container)
INFO - Created Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Created Ejb(deployment-id=org.superbiz.injection.tx.MoviesTest, ejb-
```

```
name=org.superbiz.injection.tx.MoviesTest, container=Default Managed Container)
INFO - Started Ejb(deployment-id=NoTransactionBean, ejb-name=NoTransactionBean,
container=Default Stateless Container)
INFO - Started Ejb(deployment-id=TransactionBean, ejb-name=TransactionBean,
container=Default Stateless Container)
INFO - Started Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Started Ejb(deployment-id=org.superbiz.injection.tx.MoviesTest, ejb-
name=org.superbiz.injection.tx.MoviesTest, container=Default Managed Container)
INFO - Deployed Application(path=/Users/dblevins/examples/movies-complete-meta)
INFO - EJBContainer already initialized.  Call ejbContainer.close() to allow
reinitialization
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.869 sec

Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```