# Testing Security

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

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

# Movie

```
package org.superbiz.injection.secure;

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.secure;

//START SNIPPET: code

import javax.annotation.security.PermitAll;
import javax.annotation.security.RolesAllowed;
import javax.ejb.Stateful;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
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;

    @RolesAllowed({"Employee", "Manager"})
    public void addMovie(Movie movie) throws Exception {
        entityManager.persist(movie);
    }

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

    @PermitAll
    @TransactionAttribute(TransactionAttributeType.SUPPORTS)
    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.secure.Movie</class>

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

# MovieTest

```
package org.superbiz.injection.secure;

import junit.framework.TestCase;

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

//START SNIPPET: code

public class MovieTest extends TestCase {

    @EJB
    private Movies movies;

    @EJB(name = "ManagerBean")
    private Caller manager;

    @EJB(name = "EmployeeBean")
    private Caller employee;

    protected void setUp() 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");
```

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

    public void testAsManager() throws Exception {
        manager.call(new Callable() {
            public Object call() 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());
                return null;
            }
        });
    }

    public void testAsEmployee() throws Exception {
        employee.call(new Callable() {
            public Object call() 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) {
                    try {
                        movies.deleteMovie(movie);
                        fail("Employees should not be allowed to delete");
                    } catch (EJBAccessException e) {
                        // Good, Employees cannot delete things
                    }
                }

                // The list should still be three movies long
                assertEquals("Movies.getMovies()", 3, movies.getMovies().size());
                return null;
            }
        });
```

```
    }

    public void testUnauthenticated() throws Exception {
        try {
            movies.addMovie(new Movie("Quentin Tarantino", "Reservoir Dogs", 1992));
            fail("Unauthenticated users should not be able to add movies");
        } catch (EJBAccessException e) {
            // Good, guests cannot add things
        }

        try {
            movies.deleteMovie(null);
            fail("Unauthenticated users should not be allowed to delete");
        } catch (EJBAccessException e) {
            // Good, Unauthenticated users cannot delete things
        }

        try {
            // Read access should be allowed

            List<Movie> list = movies.getMovies();
        } catch (EJBAccessException e) {
            fail("Read access should be allowed");
        }
    }


    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 desired security scope.
     */

    @Stateless
    @RunAs("Manager")
    public static class ManagerBean implements Caller {

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

    @Stateless
    @RunAs("Employee")
    public static class EmployeeBean implements Caller {

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

```
        }
    }
}
```

# Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.injection.secure.MovieTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/testing-security
INFO - openejb.base = /Users/dblevins/examples/testing-security
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/testing-
security/target/classes
INFO - Found EjbModule in classpath: /Users/dblevins/examples/testing-
security/target/test-classes
INFO - Beginning load: /Users/dblevins/examples/testing-security/target/classes
INFO - Beginning load: /Users/dblevins/examples/testing-security/target/test-classes
INFO - Configuring enterprise application: /Users/dblevins/examples/testing-security
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 Stateless Container, type=Container, provider-
id=Default Stateless Container)
INFO - Auto-creating a container for bean ManagerBean: Container(type=STATELESS,
id=Default Stateless 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.secure.MovieTest:
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/testing-security" loaded.
```

```
INFO - Assembling app: /Users/dblevins/examples/testing-security
INFO - PersistenceUnit(name=movie-unit,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 405ms
INFO - Jndi(name="java:global/testing-
security/Movies!org.superbiz.injection.secure.Movies")
INFO - Jndi(name="java:global/testing-security/Movies")
INFO - Jndi(name="java:global/testing-
security/ManagerBean!org.superbiz.injection.secure.MovieTest$Caller")
INFO - Jndi(name="java:global/testing-security/ManagerBean")
INFO - Jndi(name="java:global/testing-
security/EmployeeBean!org.superbiz.injection.secure.MovieTest$Caller")
INFO - Jndi(name="java:global/testing-security/EmployeeBean")
INFO -
Jndi(name="java:global/EjbModule26174809/org.superbiz.injection.secure.MovieTest!org.s
uperbiz.injection.secure.MovieTest")
INFO -
Jndi(name="java:global/EjbModule26174809/org.superbiz.injection.secure.MovieTest")
INFO - Created Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Created Ejb(deployment-id=ManagerBean, ejb-name=ManagerBean, container=Default
Stateless Container)
INFO - Created Ejb(deployment-id=EmployeeBean, ejb-name=EmployeeBean,
container=Default Stateless Container)
INFO - Created Ejb(deployment-id=org.superbiz.injection.secure.MovieTest, ejb-
name=org.superbiz.injection.secure.MovieTest, container=Default Managed Container)
INFO - Started Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Started Ejb(deployment-id=ManagerBean, ejb-name=ManagerBean, container=Default
Stateless Container)
INFO - Started Ejb(deployment-id=EmployeeBean, ejb-name=EmployeeBean,
container=Default Stateless Container)
INFO - Started Ejb(deployment-id=org.superbiz.injection.secure.MovieTest, ejb-
name=org.superbiz.injection.secure.MovieTest, container=Default Managed Container)
INFO - Deployed Application(path=/Users/dblevins/examples/testing-security)
INFO - EJBContainer already initialized.  Call ejbContainer.close() to allow
reinitialization
INFO - EJBContainer already initialized.  Call ejbContainer.close() to allow
reinitialization
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.574 sec

Results :

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
```