Testing Security 3

Example testing-security-3 can be browsed at https://github.com/apache/tomee/tree/master/examples/testing-security-3

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();
    }
}
```

MyLoginProvider

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

import org.apache.openejb.core.security.jaas.LoginProvider;

import javax.security.auth.login.FailedLoginException;
import java.util.Arrays;
import java.util.List;

public class MyLoginProvider implements LoginProvider {


    @Override
    public List<String> authenticate(String user, String password) throws
FailedLoginException {
        if ("paul".equals(user) && "michelle".equals(password)) {
            return Arrays.asList("Manager", "rockstar", "beatle");
        }

        if ("eddie".equals(user) && "jump".equals(password)) {
            return Arrays.asList("Employee", "rockstar", "vanhalen");
        }

        throw new FailedLoginException("Bad user or password!");
    }
}
```

org.apache.openejb.core.security.jaas.LoginProvider

```
org.superbiz.injection.secure.MyLoginProvider
```

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.ejb.EJB;
import javax.ejb.EJBAccessException;
import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.util.List;
import java.util.Properties;

public class MovieTest extends TestCase {

    @EJB
    private Movies movies;

    private Context getContext(String user, String pass) throws NamingException {
        Properties p = new Properties();
        p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.core.LocalInitialContextFactory");
        p.setProperty("openejb.authentication.realmName", "ServiceProviderLogin");
        p.put(Context.SECURITY_PRINCIPAL, user);
        p.put(Context.SECURITY_CREDENTIALS, pass);

        return new InitialContext(p);
    }

    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 {
        final Context context = getContext("paul", "michelle");

        try {
            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());
        } finally {
            context.close();
        }
    }

    public void testAsEmployee() throws Exception {
        final Context context = getContext("eddie", "jump");

        try {
            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());
        } finally {
            context.close();
```

```
        }
    }

    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 void testLoginFailure() throws NamingException {
        try {
            getContext("eddie", "panama");
            fail("supposed to have a login failure here");
        } catch (javax.naming.AuthenticationException e) {
            //expected
        }

        try {
            getContext("jimmy", "foxylady");
            fail("supposed to have a login failure here");
        } catch (javax.naming.AuthenticationException e) {
            //expected
        }
    }
}
```

Running

```
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.injection.secure.MovieTest
INFO -
********************************************************************************
INFO - OpenEJB http://tomee.apache.org/
INFO - Startup: Fri Jul 20 08:42:53 EDT 2012
INFO - Copyright 1999-2012 (C) Apache OpenEJB Project, All Rights Reserved.
INFO - Version: 4.1.0
INFO - Build date: 20120720
INFO - Build time: 08:33
INFO -
********************************************************************************
INFO - openejb.home = /home/boto/dev/ws/openejb_trunk/openejb/examples/testing-
security-3
INFO - openejb.base = /home/boto/dev/ws/openejb_trunk/openejb/examples/testing-
security-3
INFO - Created new singletonService
org.apache.openejb.cdi.ThreadSingletonServiceImpl@38ee6681
INFO - Succeeded in installing singleton service
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
INFO - Cannot find the configuration file [conf/openejb.xml].  Will attempt to create
one for the beans deployed.
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 - Creating TransactionManager(id=Default Transaction Manager)
INFO - Creating SecurityService(id=Default Security Service)
INFO - Creating Resource(id=movieDatabase)
INFO - Beginning load: /home/boto/dev/ws/openejb_trunk/openejb/examples/testing-
security-3/target/classes
INFO - Configuring enterprise application:
/home/boto/dev/ws/openejb_trunk/openejb/examples/testing-security-3
INFO - Auto-deploying ejb Movies: EjbDeployment(deployment-id=Movies)
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 - Creating Container(id=Default Stateful Container)
INFO - Using directory /tmp for stateful session passivation
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 - Creating Container(id=Default Managed Container)
INFO - Using directory /tmp for stateful session passivation
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 - Creating Resource(id=movieDatabaseNonJta)
INFO - Adjusting PersistenceUnit movie-unit <non-jta-data-source> to Resource ID
'movieDatabaseNonJta' from 'movieDatabaseUnmanaged'
INFO - Enterprise application
"/home/boto/dev/ws/openejb_trunk/openejb/examples/testing-security-3" loaded.
INFO - Assembling app: /home/boto/dev/ws/openejb_trunk/openejb/examples/testing-
security-3
SEVERE - JAVA AGENT NOT INSTALLED. The JPA Persistence Provider requested installation
of a ClassFileTransformer which requires a JavaAgent.  See
http://tomee.apache.org/3.0/javaagent.html
INFO - PersistenceUnit(name=movie-unit,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 268ms
INFO - Jndi(name="java:global/testing-security-
3/Movies!org.superbiz.injection.secure.Movies")
INFO - Jndi(name="java:global/testing-security-3/Movies")
INFO - Existing thread singleton service in SystemInstance()
org.apache.openejb.cdi.ThreadSingletonServiceImpl@38ee6681
INFO - OpenWebBeans Container is starting...
INFO - Adding OpenWebBeansPlugin : [CdiPlugin]
INFO - All injection points are validated successfully.
INFO - OpenWebBeans Container has started, it took 170 ms.
INFO - Created Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Started Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Deployed
Application(path=/home/boto/dev/ws/openejb_trunk/openejb/examples/testing-security-3)
20-Jul-2012 8:42:55 AM null openjpa.Runtime
INFO: Starting OpenJPA 2.2.0
20-Jul-2012 8:42:56 AM null openjpa.jdbc.JDBC
INFO: Using dictionary class "org.apache.openjpa.jdbc.sql.HSQLDictionary" (HSQL
Database Engine 2.2.8 ,HSQL Database Engine Driver 2.2.8).
20-Jul-2012 8:42:57 AM null openjpa.Enhance
INFO: Creating subclass and redefining methods for "[class
org.superbiz.injection.secure.Movie]". This means that your application will be less
efficient than it would if you ran the OpenJPA enhancer.
INFO - Logging in
INFO - Logging out
INFO - EJBContainer already initialized.  Call ejbContainer.close() to allow
reinitialization
INFO - Logging in
INFO - Logging out
INFO - EJBContainer already initialized.  Call ejbContainer.close() to allow
reinitialization
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.069 sec

Results :
```

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