Injection Of Datasource

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

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

Movie

```
package org.superbiz.injection;

/**
 * @version $Revision: 607077 $ $Date: 2007-12-27 06:55:23 -0800 (Thu, 27 Dec 2007) $
 */
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;
```

```
import javax.annotation.PostConstruct;
import javax.annotation.Resource;
import javax.ejb.Stateful;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

@Stateful
public class Movies {

    /**
     * The field name "movieDatabase" matches the DataSource we
     * configure in the TestCase via :
     * p.put("movieDatabase", "new://Resource?type=DataSource");
     * <p/>
     * This would also match an equivalent delcaration in an openejb.xml:
     * <Resource id="movieDatabase" type="DataSource"/>
     * <p/>
     * If you'd like the freedom to change the field name without
     * impact on your configuration you can set the "name" attribute
     * of the @Resource annotation to "movieDatabase" instead.
     */
    @Resource
    private DataSource movieDatabase;

    @PostConstruct
    private void construct() throws Exception {
        Connection connection = movieDatabase.getConnection();
        try {
            PreparedStatement stmt = connection.prepareStatement("CREATE TABLE movie (
director VARCHAR(255), title VARCHAR(255), year integer)");
            stmt.execute();
        } finally {
            connection.close();
        }
    }

    public void addMovie(Movie movie) throws Exception {
        Connection conn = movieDatabase.getConnection();
        try {
            PreparedStatement sql = conn.prepareStatement("INSERT into movie
(director, title, year) values (?, ?, ?)");
            sql.setString(1, movie.getDirector());
            sql.setString(2, movie.getTitle());
            sql.setInt(3, movie.getYear());
            sql.execute();
        } finally {
```

```
            conn.close();
        }
    }

    public void deleteMovie(Movie movie) throws Exception {
        Connection conn = movieDatabase.getConnection();
        try {
            PreparedStatement sql = conn.prepareStatement("DELETE from movie where
director = ? AND title = ? AND year = ?");
            sql.setString(1, movie.getDirector());
            sql.setString(2, movie.getTitle());
            sql.setInt(3, movie.getYear());
            sql.execute();
        } finally {
            conn.close();
        }
    }

    public List<Movie> getMovies() throws Exception {
        ArrayList<Movie> movies = new ArrayList<Movie>();
        Connection conn = movieDatabase.getConnection();
        try {
            PreparedStatement sql = conn.prepareStatement("SELECT director, title,
year from movie");
            ResultSet set = sql.executeQuery();
            while (set.next()) {
                Movie movie = new Movie();
                movie.setDirector(set.getString("director"));
                movie.setTitle(set.getString("title"));
                movie.setYear(set.getInt("year"));
                movies.add(movie);
            }
        } finally {
            conn.close();
        }
        return movies;
    }
}
```

MoviesTest

```
package org.superbiz.injection;

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 {

        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");

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

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

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.injection.MoviesTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
```

```
INFO - openejb.home = /Users/dblevins/examples/injection-of-datasource
INFO - openejb.base = /Users/dblevins/examples/injection-of-datasource
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-
datasource/target/classes
INFO - Beginning load: /Users/dblevins/examples/injection-of-datasource/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/injection-of-
datasource
WARN - Method 'lookup' is not available for 'javax.annotation.Resource'. Probably
using an older Runtime.
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 - Auto-linking resource-ref
'java:comp/env/org.superbiz.injection.Movies/movieDatabase' in bean Movies to
Resource(id=movieDatabase)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.injection.MoviesTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Enterprise application "/Users/dblevins/examples/injection-of-datasource"
loaded.
INFO - Assembling app: /Users/dblevins/examples/injection-of-datasource
INFO - Jndi(name="java:global/injection-of-
datasource/Movies!org.superbiz.injection.Movies")
INFO - Jndi(name="java:global/injection-of-datasource/Movies")
INFO -
Jndi(name="java:global/EjbModule1508028338/org.superbiz.injection.MoviesTest!org.super
biz.injection.MoviesTest")
INFO - Jndi(name="java:global/EjbModule1508028338/org.superbiz.injection.MoviesTest")
INFO - Created Ejb(deployment-id=Movies, ejb-name=Movies, container=Default Stateful
Container)
INFO - Created Ejb(deployment-id=org.superbiz.injection.MoviesTest, ejb-
name=org.superbiz.injection.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.MoviesTest, ejb-
name=org.superbiz.injection.MoviesTest, container=Default Managed Container)
INFO - Deployed Application(path=/Users/dblevins/examples/injection-of-datasource)
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.276 sec

Results :

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