EasyBeans Test Suite's Guide

Eduardo STUDZINSKI ESTIMA DE CASTRO, OW2 consortium
Gisele PINHEIRO SOUZA, OW2 consortium

EasyBeans Test Suite's Guide

by Eduardo STUDZINSKI ESTIMA DE CASTRO and Gisele PINHEIRO SOUZA

Publication date $Id: testguide.xml 3272 2008-05-21 09:53:35Z benoitf $

Abstract

The EasyBeans Test Suite guide is intended for developers wanting to develop tests to EJB3 Containers.

Table of Contents

Chapter 1. Goal

The test suite objective is to verify that the EasyBeans conforms with the EJB3 specification. The suite is under construction and everyone that wants to contribute is welcome. Anyone can write or add tests cases according to the guidelines below.

This test suite is designed for EasyBeans; however, with a few adaptations, it could work with others application servers.

Chapter 2. Description of the source tree

The test suite contains:

- `build.xml` - ant file to build, install and run the tests.
- `tests-common.xml` - ant file with the common definitions for all ant files.
- `tests-functions.xml` - ant file with common functions for all ant files.
- `src directory` - contains the source files.
- `lib directory` - contains the libraries used in the tests.
- `conf directory` - contains the files used to configure the tests.
- `examples directory` - contains the examples files.

The source directory contains all test cases as well as the auxiliary classes used to make the tests. These sources are divided in packages by functionality. The `src/java` folder has the package org.objectweb.easybeans.tests which contains the follows packages:

- annotations - tests cases for verifying the annotations.
- common - contains all support classes to do the tests.
  - asserts - contains alternative assert classes that are used during the tests.
  - core - contains the classes that extends the server features.
  - db - contains the classes that manipulates the database.
  - ejbs - contains the beans that are used in the tests.
  - enhancer - contains the classes that are used to test the enhancer.
  - exception - contains exceptions used during the test.
  - helper - contains the helper classes.
  - inheritance - contains the classes that are used in the inheritance tests.
  - interceptors - contains the classes that are used in the interceptor tests.
  - interfaces - contains the common interfaces that are used by different classes (beans, test class and others).
  - resources - contains the classes that are used in the resource tests.
- deploymentdesc - tests cases for verifying the deployment descriptor.
- enhancer - tests cases for verifying the enhancer.
- inheritance - tests cases for verifying the inheritance.
- interceptors - tests cases for verifying the interceptors.
- transaction - tests cases for verifying the transaction.

- taglets - contains javadoc tags used to format correctly the documentation.

The file `src/java/testng_conformance.xml` makes possible to run all tests in the same time. However, it is possible to execute a group of tests individually by using the XML file available in each test case package.

Chapter 3. Getting EasyBeans From the SVN Repository

Anyone can check out source code from the SVN server using the following command (for GUI SVN client use, configuration values are the same as for command line use):

svn checkout svn://svn.forge.objectweb.org/svnroot/easybeans/trunk/easybeans

Chapter 4. Building the suite from source

4.1. Requirements

First, all the requirements used in Developer's Guide building chapter [http://wiki.easybeans.org/xwiki/bin/Main/Documentation] need to be checked.

4.2. Building the suite

The `build.xml` file which is at the root of the `tests` folder will be used.

In the `tests` folder of the project (named EasyBeans by default), the command **ant install** needs to be launched.

Note

The command **ant -p** could be used to list the targets that are available.

Once the command has been run successfully, `output` and `tests/output` folders are created and each one contains a subfolder `classes`. The `output/classes` contains the EasyBeans Server classes and the `tests/output/classes` contains the classes used by the test suite. Furthermore, the binaries(.jar) used with EasyBeans Server are installed in the `ejb3s` folder.

ant clean.classes is used to clean the generated classes.

4.3. Test suite main ant targets

4.3.1. JavaDoc

Javadoc of Test Suite can be generated by using **ant javadoc**

The resulting documentation will be available in the `tests/output/javadoc` folder.

4.3.2. Compile

In the `tests` folder of the project (named EasyBeans by default), the command **ant compile** needs to be launched.

Once the command has been run successfully, `output` and `tests/output` folders are created and each one contains a subfolder `classes`. The `output/classes` contains the EasyBeans Server classes and the `tests/output/classes` contains the classes used by the test suite.

Chapter 5. Running the suite

5.1. Requirements

First, all the requirements used in building chapter need to be checked.

5.2. Using Ant

In the `tests` folder of the project (named EasyBeans by default), the following three commands need to be used:

1. **ant install**

   It will install the binaries required by the test suite in the folder `ejb3s`.

2. **ant run.server** or **run.server.debug** (Server debug mode)

   Then, EasyBeans server will be launched. When the following message is received, next step needs to be done:

   ```
   - - - - -[java] -INFO: -Waiting -requests...
   ```

3. **ant tests.conformance**

   All tests will be run and a report will be produced. The report will be available in `tests/output/reports/conformance` folder.

Note

After using **ant tests.conformance**, the JUnit Report format could be generated using the command **ant report.junit.format**. The output file `junit-noframes.html` will be available in `tests/output/reports/conformance` folder.

Chapter 6. How to write a test

The example shows the basis of a test with all steps used to build a EasyBeans test.

6.1. Requirements

This example only requires Java 5 knowledges.

6.2. Writing the test class code

This example tests if a business method can be invoked from a bean, and if the bean can return an Integer value without modifications.

The source code(`TestExample.java`) can be found in the `tests/examples` folder of the project (named EasyBeans by default):

```
package -org.objectweb.easybeans.tests.examples;

import -static -org.objectweb.easybeans.tests.common.helper.EJBHelper.getBeanRemoteInstance;
import -static -org.testng.Assert.assertEquals;

import -org.objectweb.easybeans.tests.common.ejbs.base.ItfExample;
import -org.objectweb.easybeans.tests.common.ejbs.stateless.containermanaged.SLSBExample;
import -org.testng.annotations.Configuration;
import -org.testng.annotations.Test;

/**
 -* -This -is -an -example -of -a -EasyBeans -Test -Suite -Class.
 -* -@reference -It -is -used -to -specify -the -document -that -the -tests -cover. -Example:
 -* - - - - - - - - - - -JSR220-PROPOSED -FINAL
 -* -@requirement -It -is -used -to -specify -the -classes -and -files -needed -to -run -the
 -* - - - - - - - - - - - - - -tests. -Exampe: -EasyBeans -must -be -running -and -the -bean
 -* - - - - - - - - - - - - - -org.objectweb.easybeans.tests.common.ejbs.stateless.containermanaged.SFSBExample
 -* - - - - - - - - - - - - - -must -be -deployed.
 -* -@setup -It -is -used -to -specify -the -classes -and -files -needed -to -run -the -test.
 -* -@author -Eduardo -Studzinski -Estima -de -Castro
 -* -@author -Gisele -Pinheiro -Souza
 -*/
public -class -TestExample -{

 - - - -/**
 - - - - -* -Constant.
 - - - - -*/
 - - - -private -static -final -Integer -INPUT -= -new -Integer(1);

 - - - -/**
 - - - - -* -Bean -used -in -tests.
 - - - - -*/
 - - - -private -ItfExample<Integer> -bean;

 - - - -/**
 - - - - -* -Gets -a -new -bean -instance -used -during -the -tests.
 - - - - -* -@throws -Exception -if -an -error -occurs -during -the -setup.
 - - - - -*/
 - - - -@Configuration(beforeTestMethod -= -true)
 - - - -public -void -setup() -throws -Exception -{
 - - - - - - - - -// -Gets -a -bean -instance.
 - - - - - - - - -bean -= -getBeanRemoteInstance(SLSBExample.class, -ItfExample.class);
 - - - -}

 - - - -/**
 - - - - -* -Indicates -the -test -description. -Example: -Tests -if -the -bean -can -return -a
 - - - - -* -value -without -modifications.
 - - - - -* -@input -It -is -used -to -specify -the -classes -and -files -needed -to -run -the
 - - - - -* - - - - - - - -test. -Example: -Integer -value.
 - - - - -* -@output -It -is -used -to -specify -the -classes -and -files -needed -to -run -the
 - - - - -* - - - - - - - - -test. -Example: -The -same -input -integer.
 - - - - -* -@throws -Exception -if -an -error -occurs -during -the -test.
 - - - - -*/
 - - - -@Test
 - - - -public -void -test00() -throws -Exception -{
 - - - - - - - - -// -Output -value, -it -must -be -the -same -as -the -input.
 - - - - - - - - -Integer -output -= -bean.getValue(INPUT);
```

```
- - - - - - - -// -Test -if -input -and -output -are -equal.
- - - - - - - -assertEquals(INPUT, -output, -"The -input -and -output -values -should -be -equal.");
- - - -}
}
```

6.3. TestNG Annotations

There are two mainly annotations in TestNG: `@Configuration` and `@Test`.

6.3.1. @Configuration

This annotation is used to make any method a setup or a teardown method. In addition, the @Configuration makes unnecessary to follow any naming convention in method names. The mainly properties that could be used are:

- `afterTest` - if true, the method will be run after each test.
- `afterTestClass`- if true, the method will be run after all the tests in the test class.
- `beforeTestClass`- if true, the method will be run after the test class instantiation and before tests methods.
- `beforeTestMethod`- if true, the method will be run before any test method.

In the example, the method `setup()` will request a new bean instance before any test method.

```
- - - -/**
- - - - -* -Gets -a -new -bean -instance -used -during -the -tests.
- - - - -* -@throws -Exception -if -an -error -occurs -during -the -setup.
- - - - -*/
- - - -@Configuration(beforeTestMethod -= -true)
- - - -public -void -setup() -throws -Exception -{
- - - - - - - - -// -Gets -a -bean -instance.
- - - - - - - - -bean -= -getBeanRemoteInstance(SLSBExample.class, -ItfExample.class);
- - - -}
```

6.3.2. @Test

This annotation is used to define a method that will be run as a test and it is not necessary follow any naming convention. In the example, the method `test00()` has the annotation and it will be a test case:

```
- - - -@Test
- - - -public -void -test00() -throws -Exception -{
- - - - - - - - - -...
- - - -}
```

If it is necessary to disable this test, the `enabled` property could be used:

```
- - - -@Test(enabled -= -false)
- - - -public -void -test00() -throws -Exception -{
- - - - - - - - - -...
- - - -}
```

6.4. Helper Classes

To make easier the test development, some helper classes are available in the package org.objectweb.easybeans.tests.common.helper

In the example, the EJBHelper class is used to get a bean instance:

```
bean -= -getBeanRemoteInstance(SLSBExample.class, -ItfExample.class);
```

The method `getBeanRemoteInstance()` is used to a remote access of a bean and the method `getBeanLocalInstance()` is used to a local access of a bean. They need two parameters:

1. The bean class;

2. The bean interface;

6.5. Writing the XML configuration file:

Once the test class is defined, it is necessary to write the XML configuration file for TestNG. The following example indicates that the tests inside the package org.objectweb.easybeans.tests.examples must be run:

```
<!DOCTYPE -suite -SYSTEM -"http://beust.com/testng/testng-1.0.dtd">
<suite -name="Test -Suite -Example" -verbose="1">
 - -<test -name="Test -Example">
 - - - -<packages>
 - - - - - -<package -name="org.objectweb.easybeans.tests.examples"/>
 - - -</packages>
 -</test>
</suite>
```

The TestNG XML configuration file also supports the `classes/class` tags that define which classes must be run. This following example indicates that only the tests in the TestExample class must be run:

```
<!DOCTYPE -suite -SYSTEM -"http://beust.com/testng/testng-1.0.dtd">
<suite -name="Test -Suite -Example" -verbose="1">
 - -<test -name="Test -Example">
 - - - -<classes>
 - - - - - -<class -name="org.objectweb.easybeans.tests.examples.TestExample"/>
 - - -</classes>
 -</test>
</suite>
```

6.6. Additional Information

Additional information about TestNG could be found in www.testng.org [http://www.testng.org].

Chapter 7. How to contribute?

Anyone can contribute to the tests according with the follow guideline. Contributions are welcome.

7.1. JavaDoc comment convention

The test classes respect a comment convention that has some custom tags. The custom tags are:

- `@reference` - used in class comment. It is used to specify the document that the tests cover.
- `@requirement` - used in class comment. It is used to specify the classes and files needed to run the tests.
- `@setup` - used in class comment. It is used to specify the items that must be set up before the test execution.
- `@input` - used in method comment. It is used to specify the test inputs.
- `@output` - used in method comment. It is used to specify the test expected outputs.

7.2. Code convention

The Test Suite and the EasyBeans has the same code convention. You can see the convention details in EasyBeans code convention [http://www.easybeans.org/doc/developerguide/en/integrated/developerguide.html#code_convention].

7.3. Methodology for writing tests

- It is highly recommended the use of inheritance and generics types to avoid code duplication.

  The tests must be independents, so after a test execution everything must be clear. It makes possible to run the same test many times without problems.

- In the test methods, the **try/catch** is used only when the exception is expected.

7.4. Adding a test suite

The first thing needed is defining where the test classes should be in. After that, the TestNG XML file must be created to run the tests. Each package has a XML file to run all package tests. In addition, an entry for each test package is needed in the file `src/main/java/testng_conformance.xml`.

In the `tests/examples` folder of the project (named EasyBeans by default), there is a TestNG XML file example called `testng_xml`:

```
<!DOCTYPE -suite -SYSTEM -"http://beust.com/testng/testng-1.0.dtd">
<suite -name="Test -Suite -Example" -verbose="1">
 - -<test -name="Test -Example">
 - - - -<packages>
 - - - - - -<package -name="org.ow2.easybeans.tests.examples"/>
 - - -</packages>
 -</test>
</suite>
```

7.5. Submmiting a contribution

The submmition has the same rule that the EasyBeans contributions. You can see details in EasyBeans contribution [http://www.easybeans.org/doc/developerguide/en/integrated/developerguide.html#contributing].

Chapter 8. Current conformance test results

The EasyBeans tests results will be available as soon as possible.