Building Tests for the ZEOS Library

Michael Seeger
<miseeger*at*users*dot*sourceforge*dot*net>

This document describes how to set up the ZEOS Build & Test environment and shows how to compile gui and console based test applications to test the ZEOSLib code. It also describes the setup that has to be made for testing the library with various databases.

Table of Contents

1. Requirements to build ZEOSLib tests 1
2. Build & Test Configuration 1
3. Database Configuration 3
4. Compiling Tests 4
5. Running Tests 4

1. Requirements to build ZEOSLib tests

In order to run the tests for the ZEOSLib you have to be sure that you have insalled the following software on your system:

- Delphi Compiler / IDE (min. professional versions 5, 6, 7 or 2005)
- DUnit Xtreme testing framework for Delphi [Homepage [http://dunit.sourceforge.net] | Download Page [https://sourceforge.net/projects/dunit]]
- Java Runtime Environment or SDK (Version 1.4.2) [Homepage [http://java.sun.com] | Download Page [http://java.sun.com/j2se/1.4.2/download.htm]]
- Jakarta Ant [Download Page [http://ant.apache.org] | Download Page [http://ant.apache.org/bindownload.cgi]]
- The supported SQL Servers you want to test with

Make sure that the environment variable JAVA_HOME is set to the directory where you installed the Java Runtine or SDK (e. g. JAVA_HOME = c:\java\j2sdk1.4.2_06).

Also set the environment variable ANT_HOME to the directory where you installed Jakarta ANT (e. g. ANT_HOME = C:\Programs\ANT).

2. Build & Test Configuration

Before you are able to start compiling the ZEOS test applications you have to configure the compiler environment. All settings for compiling building and testing are stored in a file called build.properties. To get this file just copy the build_template.properties file in build directory and name it build.properties. The build.properties file has four sections that are important for compiling the test applications: "common" and "compilers". There is another section that determines which tests will be executed by calling test.cmd batchfile. This section is not important for compiling tests but it will also be documented.

In a Windows environment it is recommended that you use double backslash as directory separator. "Normal" slash will work but causes some problems in a Delphi 9 environment. In a Unix / Linux environment it is recommended use the "normal" slash as direcrory separator.

2.1. Common Section

The first section is the common-section it contains common informations for building the test applications:

```
[common]
project.home=d:/workshop/zeosdbo_rework (1)
release.version =6.5.2-beta (2)
copy.verbose=false (3)
dunit.dir=d:/programme/borland/delphi7/dunit/src (4)
kunit.dir= (5)
```

(1) The project directory
(2) The version number of the ZEOSLib (put into version file).
(3) Determines whether copy operations shall displayed on screen.
(4) The source directory of DUnit.
(5) The source directory of DUnit (for Kylix).

2.2. Compiler Section

The second section is the compiler-section it contains informations about the active compilers to use for building the test applications. Each supported compiler has its own "section". It describes wether the compiler (prefix of the key (e. g. "delphi5")) is active and determines the installation directory of the compiler.

```
[compilers]
delphi5.active=false (1)
delphi5.home=C:/Program Files/Borland/Delphi5 (2)
:
delphi9.active=true
delphi9.home=d:\\programme\\borland\\bds\\3.0
delphi9.bpl.dir=c:\\dokumente und einstellungen\\user\\eigene dateien\\borland studio
(3)
:
kylix3.active=false (4)
kylix3.home=/opt/kylix3
```

(1) "true" if the compiler is installed (active), "false" if not.
(2) The installation path of the compiler (here: Delphi 5).
(3) Special for Delhi 9: you have to specify your BPL-dir because in Delphi 9 the standardized ...\projects\bpl doesn't exist anymore (here: german Delphi / WinXP example).
(4) Kylix 3 settings for compilation (Kylix 3 is not installed this is why kylix3.active=false)

2.3. Tests Section

This section is necessary to determine which tests will be run when executing the test.cmd batchfile (only makes sense when compiling the test applications as console applications).

```
[tests]
test.core=true (1)
test.parsesql=true (2)
test.dbc=true (3)
test.component=true (4)
test.bugreport=true (5)
```

```
test.performance=false (6)
```

(1) Execute core tests if set to "true".
(2) Execute parsing tests if set to "true".
(3) Execute connectivity tests if set to "true".
(4) Execute component tests if set to "true".
(5) Execute bug tests if set to "true".
(6) Execute performance tests if set to "true".

3. Database Configuration

It is possible to run the ZEOS tests for a number of SQL servers that are currently running on the test machine. Therefor the Build and Test Environment has to be configured.

Setting up the database parameters is done in the test.properites file that is located in the "database" direcory. To get this file just copy the test_template.properties file and rename it to test.properties and then make your settings.

At the moment we do not execute performance tests so only the common section and the database sections are of importance for testing the ZEOS Library.

3.1. Common Section

The first section is the common-section it contains common informations about the databases to test by test applications:

```
[common]
common.connections=sqlite28,firebird15 (1)
```

(1) This key holds the active servernames of the SQL servers that will be tested in the test applications. Corresponding to the servers given here there has to exist a "database section" with the same name that determines the database settings for testing. The server names for this key have to be seperated by comma.

3.2. Database Settings

Corresponding to the active servers listed in the common.connections key (here: SQLite 2.8 and Firebird 1.5.x) there are configuration sections that contain the settings for running tests with the given database. Each section consists of the same keys so we will take the Firebird 1.5 database configuration to explain them:

```
[firebird15]
firebird15.protocol=firebird-1.5 (1)
firebird15.alias=firebird_zeoslib (2)
firebird15.host=localhost (3)
firebird15.port= (4)
firebird15.database=d:\SQLServerFarm\Firebird\15\Data\zeoslib.fdb (5)
firebird15.user=SYSDBA (6)
firebird15.password=masterkey (7)
firebird15.rebuild=yes (8)
firebird15.delimiter.type=SetTerm (9)
firebird15.delimiter=; (10)
firebird15.create.scripts=create_interbase.sql,populate_any.sql,populate_interbase.sq
firebird15.drop.scripts=drop_interbase.sql,drop_interbase_bugreport.sql
```

1. DBC protocol name
2. BDE alias for performance tests
3. Host name
4. Port number
5. Database name
6. User name
7. User password
7. User password
7. Flag to rebuild database before each test
9. SQL delimiter type in SQL scripts (Default, Go, SetTerm or EmptyLine)
10. SQL delimiter string
    SQL scripts to create database objects. All scripts have to be placed in the "database" directory (where the test.properties file is located).
    SQL scripts to drop database objects. All scripts have to be placed in the "database" directory (where the test.properties file is located).

Due to techniques used in the ZEOS Testframework classes it is necessary to set a prefix in front of the key that is identical to the section name followed by a dot. E. g.: firebird15.password - where "firebird15" is the repeated name of the section (see above).

4. Compiling Tests

When all the required Software is installed and you made the configurations that are necessary for your test environment you are ready to start compiling the test applications. This is easyly done by calling the batch script "compiletests.cmd" from commandline in "build" directory. When all applications are successfully compiled you are able to test ZEOSLib funcionality.

ZEOS BTE splits the tests into seven parts (separate applications):

- Bug Report Tests (ZTestBugReport.EXE)
- Component Tests (ZTestComponentAll.EXE)
- Core Tests (ZTestCoreAll.EXE)
- Database Connectivity (DBC) Tests (ZTestDbcAll.EXE)
- SQL Parser Tests (ZTestParseSqlAll.EXE)
- Performance Tests - currently not executed - (ZTestPerformance.EXE)

The applications listed above are compiled into the build directory of the corresponding package (e. g.: ...\packages\delphi7\build). Normally the applications are compiled to execute as console applications but if you want to run them as GUI applications just remove the comment from "{$DEFINE TESTGUI}" in the ZEOS.INC file. If you now call one of the test applications they are executed in a GUI. If you want to create console test applications you just have to comment the "{$DEFINE TESTGUI}" again (in ZEOS.INC).

5. Running Tests

As written in section "Compiling Tests", it is possible to execute special tests (e. g. only Bug Report Tests) by calling the EXE-file explicitly. Depending on the settings you will start one console application or one GUI application. You also may run all tests by running them sequentially via batch script (consider to compile all the tests as a console application!)

Before you run tests make sure that all the database servers you want to test are up and running!

To do run all tests (that are defined to run in section "tests" of the build.properties file), automatically, just type "test" from a commandline in "build" directory. The test result is output to screen and into a log file located in the "build" subdirectory "logs". The logfile is called "test-YYYYMMDD.log" where "YYYYMMDD" is the current date.