JUnit

概要

面倒なテストを楽にするもの。
何度も同じテストを自動的に行えるので、プログラムの改修を気兼ねなくできる。

注意

4.1 から awt や swing の実行画面がなくなった。

- junit.awtui.TestRunner
- junit.swingui.TestRunner

がなくなり、

- junit.textui.TestRunner

だけになった。
これしか使わないのであまり気にしなくてイイ。

http://www.ne.jp/asahi/hishidama/home/tech/java/junit.html
4.1 からアノテーションを使えるようになった。

	JUnit3.8	JUnit4.1
ライブラリのインポート	TestCase クラスをインポートする。	Assert クラスの各メソッドを static インポートする。Test アノテーション等をインポートする。
テストクラスの定義	TestCase を継承する必要がある。	特になし。
テストメソッドの定義	「public void test 〜 ()」test で始まる名前で、public void で引数なしのメソッドが実行対象。	「@Test public void 〜 ()」@Test アノテーションを付けた public void で引数なしのメソッドが実行対象。
テストメソッド実行前後	setUp()・tearDown() が各テストメソッドの実行前後に呼ばれる。	@Before・@After を付けたメソッドが、各テストメソッドの実行前後に呼ばれる。
全テストの実行前後	なし。	@BeforeClass・@AfterClass を付けたメソッドが、全テストの実行前後に呼ばれる。

3.8 の実装例

4.1 の実装例

準備

http://www.junit.org/
からダウンロードして、

```
junit-****.jar
```

にクラスパスを通すだけ。

サンプル

このソースをテストしたい場合

実行は

```
java junit.textui.TestRunner TestSample
```

TestSample の main からテストを実行することもできる。
JUnit4 以降の場合
Junit3 以前の場合
実行は

```
java TestSample
```

DBUnit

概要

DB へのアクセスを JUnit でテストするもの。
元データや結果のデータをエクセルファイルで準備することも出来る。
便利。

準備

http://muimi.com/j/test/dbunit/

http://dbunit.sourceforge.net/
からダウンロード。
エクセルを読み書きするには、Jakarta POI が必要。
その他に必要なライブラリは、DBUnit の pom.xml に書かれている。

- slf4j-api-1.6.6.jar
- slf4j-log4j12-1.6.6.jar
- log4j-1.2.15.jar
- poi-3.2-FINAL-20081019.jar
- dbunit-2.4.8.jar
- junit-4.10.jar

あたりがあれば、とりあえず動くはず。

サンプル

テストの対象となるクラス

テストを実施するクラス。

■データを作成する

エクセルでデータを作成する。
とりあえず、DB をエクセルにエクスポートしてそれを編集して使う。
エクスポートするために簡単なプログラムを作る。

これで出力されたエクセルを編集して、テストのインプットや結果の比較対象を作成する。

■ log4j.properties

log4j を内部で使っているので、設定ファイルが無いと怒られる。
log4j.properties をクラスパスが通った場所に置く。
中身はテキトウ。

```
### direct log messages to stdout ###
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %5p %c{1} - %m%n

### direct messages to file mylog.log ###
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=mylog.log
log4j.appender.file.Append=true
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d %5p %c{1} - %m%n

#log4j.rootLogger=debug, stdout, file
# log4j.rootLogger=warn, stdout, file
log4j.rootLogger=error, stdout, file
```

実行は

```
java junit.textui.TestRunner TestSample
```

処理日付などの値が必要な場合

テストの結果として、処理日付などが必要な場合は ReplacementDataSet を使って
期待値を書き換えることができる。

http://hamasyou.com/archives/000207

一部のカラムを比較対象から除外する

大きく 2 つの方法がある。
Assertiont#assertEqualsIgnoreCols を使う方法と
DefaultColumnFilter を使う方法

■ assertEqualsIgnoreCols を使う

```
Assertion.assertEqualsIgnoreCols(actualTable, expectedTable, new String[] {"update"});
```

■ DefaultColumnFilter を使う

```
ITable filteredActualTable = DefaultColumnFilter.excludeColumn(actualTable, new String[]
```

```
{"upd_timestamp"});
  ITable  filteredExpectedTable  =  DefaultColumnFilter.excludeColumn(expectedTable,  new  String[]
{"upd_timestamp"});
```

結果をソートして比較する

SotedTable を使うことでテーブルのデータをソートできる。これを利用して結果をソートして比較できる。

```
String[] sortColumns = new String[] {"id", "name"}
 ITable sortedActualTable = new SortedTable(actualTable, sortColumns);
 ITable sortedExpectedTable = new SortedTable(expectedTable, sortColumns);

 // 期待値と実際のデータの比較
 Assertion.assertEquals(sortedExpectedTable, sortedActualTable);
```

xlsx をデータとして利用する

列数が 256 個を超えたりする場合、xlsx を使いたくなるが dbUnit が xlsx に対応してくれていない。

対応するための機能は既にあるので、それを利用する。

■ xssf の利用

dbUnit の
・XlsDataSet
・XlsDataSetWriter
・XlsTable
を xlsx を使えるようにする。
具体的には

```
org.apache.poi.hssf.usermode
```

を

```
org.apache.poi.ss.usermodel
```

に変更する。

```
new Workbook
```

は、

```
WorkbookFactory.create()
```

にする。

セルの値にかかわらず、セルを文字列として扱う場合

Apache POI を参考に XlsDataSetWriter.java をいじる。

XlsDataSetWriter#write

Assertion#assertEquals 時のエラーを全て表示する方法

標準のままだと、比較するテーブルの差異が複数あっても、１つの差異しか表示してくれない。
普通はそれで問題ない。テストをする上では、差異があるかどうかが重要である。

ところが、テストのついでにデバッグ的なこともしたい場合は、どこがどのように異なっているかを全て知りたい。
そこで、差異をすべて表示する Assertion を作成する。

■ DbUnitAssert.java

一部抜粋
compareData メソッドの中で

Assertion.assertEquals 時に値をトリムする方法

各テーブルの項目が文字列の場合に、右側の空白をトリムして比較したい場合、
ITable を継承して独自の Table オブジェクトを作ってしまう。

複雑な変換とか ReplacementDataSet ではやりきれない場合は、この方法を使うといいかもしれない。

■ TrimTable.java

■呼出

```
Assertion.assertEquals(new TrimTable(expectedTable), actualTable);
```

AmbiguousTableNameException が発生する場合

環境によっては、

```
AmbiguousTableNameException
```

が発生する場合がある。複数のスキーマで同名のオブジェクトがあると発生することがある。
その場合は、DatabaseConnection のコンストラクタにスキーマを指定する。

```
DatabaseConnection con = new DatabaseConnection(db.getConnection(), "TEST_SCHEMA");
```

みたいな感じ。

環境まるごと

とりあえず、自分が実験した環境をまるごと置いておく。

・基本的なもの

xlsx に対応したものを含めたもの

・xlsx 対応やテーブルの差異すべて表示や項目のトリム処理を含めたもの