Agile Java

阿左 [1] Nobody [2]

January 15, 2013

[1]感谢读者
[2]感谢国家

Contents

List of Figures

List of Tables

Part I

基本概念

Chapter 1

开发环境

1.1 JUnit4

基本的 JUnit4 单元测试例子：

```
1  package net.jade;
2
3  import static org.junit.Assert.assertTrue;
4  import org.junit.BeforeClass;
5  import org.junit.AfterClass;
6  import org.junit.Before;
7  import org.junit.After;
8  import org.junit.Test;
9  import org.junit.Ignore;
10 import junit.framework.JUnit4TestAdapter;
11
12 public class HelloTest {
13
14   /**
15    * class setup must be static
16    */
17   @BeforeClass
18   public static void runBeforeClass() {
19     System.out.println("class setUp... ");
20   }
21
22   /**
23    * class tearDown must be static
```

```
24    */
25   @AfterClass
26   public static void runAfterClass() {
27     System.out.println("class tearDown... ");
28   }
29
30   @Before
31   public void setUp() {
32     System.out.println("func setUp... ");
33   }
34
35   @After
36   public void tearDown() {
37     System.out.println("func tearDown... ");
38   }
39
40   @Test
41   public void func01() {
42     System.out.println("func01... ");
43     assertTrue("hello".equals("hello"));
44   }
45
46   /**
47    * now can except for exception
48    */
49   @Test(expected=ArithmeticException.class)
50   public void func02() {
51     System.out.println("func02... ");
52     System.out.println("result is: " + (2/0));
53   }
54
55   /**
56    * this function will not run
57    * we want ignore this function
58    */
59   @Ignore
60   public void func03() {
61     System.out.println("func03... ");
62   }
63
64   /**
65    * test time out
```

```
66     */
67    @Test(timeout=500)
68    public void func04() {
69
70      System.out.println("func04... ");
71      try {
72        Thread.sleep(300);
73      } catch (InterruptedException ex) {
74        // do nothing
75      }
76    }
77
78    /**
79     * make junit4 programe alse can be used in
80     * junit3 environment
81     */
82    public static junit.framework.Test suite() {
83      return new JUnit4TestAdapter(HelloTest.class);
84    }
85  }
```

如果用 jdk 自带的方式编译与运行很麻烦：

```
1  #!/bin/bash
2  rm -rf build/classes
3  mkdir  build
4  mkdir  build/classes
5  javac -cp build/classes:lib/junit-4.8.2.jar \
6    -sourcepath src -d build/classes \
7    src/net/jade/*.java
8  java  -cp build/classes:lib/junit-4.8.2.jar \
9    org.junit.runner.JUnitCore net.jade.HelloTest
10 rm -rf build
```

有了 ant 的帮助就方便很多了：

```
1  <?xml version="1.0" ?>
2  <project name="simple" default="all" basedir=".">
3
4    <property name="projectName" value="Simple Project"/>
5
6    <property name="src.dir" value="src"/>
7    <property name="lib.dir" value="lib"/>
```

```
8
9   <property name="build.dir"        value="build"/>
10  <property name="build.classes"    value="${build.dir}/classes
        "/>
11  <property name="build.lib"        value="${build.dir}/lib"/>
12  <property name="build.pkg"        value="${build.dir}/pkg"/>
13  <property name="junit.output.dir" value="${build.dir}/
        junitreport"/>
14
15  <path id="compile.libs">
16    <fileset dir="${lib.dir}">
17      <include name="**/*.jar"/>
18    </fileset>
19    <pathelement location="${build.classes}"/>
20  </path>
21
22  <target name="clean" description="Remove all generated files.
        ">
23    <delete dir="${build.dir}" />
24  </target>
25
26  <target name="prepare" depends="clean"
27    description="Create build folders.">
28    <mkdir dir="${build.dir}"/>
29    <mkdir dir="${build.classes}"/>
30    <mkdir dir="${build.lib}"/>
31  </target>
32
33  <!-- compile. -->
34  <target name="compile" depends="prepare"
35    description="compile java scources.">
36    <javac srcdir="${src.dir}" destdir="${build.classes}"
37      includeantruntime="off">
38      <classpath refid="compile.libs"/>
39    </javac>
40  </target>
41
42  <!-- Run JUnit test classes. -->
43  <target name="junit" depends="compile">
44    <mkdir dir="${junit.output.dir}"/>
45    <junit fork="yes" printsummary="withOutAndErr"
46      haltonerror="yes" haltonfailure="yes" >
```

```
47        <formatter type="xml"/>
48        <classpath refid="compile.libs"/>
49        <test todir="${junit.output.dir}" name="net.jade.
              HelloTest"/>
50      </junit>
51    </target>
52
53    <!-- Generate JUnit report. -->
54    <target name="report"  depends="junit">
55      <junitreport todir="${junit.output.dir}">
56        <fileset dir="${junit.output.dir}">
57          <include name="TEST-*.xml"/>
58        </fileset>
59        <report format="frames" todir="${junit.output.dir}"/>
60      </junitreport>
61    </target>
62
63    <!-- Generate HTML format report. -->
64    <target name="jar" depends="report" description="compress jar
          .">
65      <jar basedir="${build.classes}" excludes="**/Test.class"
66        jarfile="${build.lib}/${projectName}.jar" />
67    </target>
68
69    <target name="all" depends="jar" description="all.">
70    </target>
71
72  </project>
```

Part II

常用工具

Chapter 2

日期时间处理

2.1 格里高利历

通过 GregorianCalendar 进行日期操作：

```
1  package example;
2
3  import java.util.Date;
4  import java.util.Calendar;
5  import java.util.GregorianCalendar;
6
7
8  /**
9   * 这是 <b一个简单的类></b> 注释。
10  *
11  * @author Jade
12  * @author 阿左
13  *
14  *
15  */
16 public class CalendarExample{
17
18   /**
19    * create date
20    * @param year
21    *  year
22    * @param month
23    *  month
```

```
24      * @param day
25      *  day
26      */
27     public Date createDate(int year, int month, int day){
28       Calendar cal = new GregorianCalendar();
29       cal.clear();
30       cal.set(Calendar.YEAR, year);
31       cal.set(Calendar.MONTH, month-1);
32       cal.set(Calendar.DAY_OF_MONTH, day);
33       return cal.getTime();
34     }
35
36     /**
37      * add days.
38      *
39      * @param date
40      *  ori date
41      * @param dayNum
42      *  how many day to add
43      *
44      */
45     public Date addDay(Date date, int dayNum) {
46       Calendar cal = new GregorianCalendar();
47       cal.setTime(date);
48       cal.add(Calendar.DAY_OF_YEAR, dayNum);
49       return cal.getTime();
50     }
51 }
```

Chapter 3

文本

3.1 换行符

在不同操作系统下取得换行符：

```
1 package stringtools;
2
3 public class StringConstans {
4
5   public static final String NEW_LINE = System.getProperty("
        line.separator");
6
7 }
```

3.2 枚举类型

```
1 package stringtools;
2
3 public enum Gender {
4   female, male
5 }
```

```
1 package stringtools;
2
3 public enum Color {
```

```
4
5     RED(255, 0, 0), BLUE(0, 0, 255), GREEN(0, 255, 0), //
6     YELLOW(255, 255, 0), BLACK(0, 0, 0), WHITE(0, 255, 0);
7
8     private int redValue;
9     private int greenValue;
10    private int blueValue;
11
12    private Color(int rv, int gv, int bv) {
13      this.redValue = rv;
14      this.greenValue = gv;
15      this.blueValue = bv;
16
17    }
18
19    public String toString() {
20      return super.toString() + "(" + redValue + "," + greenValue
            + ","
21          + blueValue + ")";
22    }
23
24  }
```

```
1   package test;
2
3   import static org.junit.Assert.assertEquals;
4
5   import org.junit.After;
6   import org.junit.Before;
7   import org.junit.Test;
8
9   import stringtools.Gender;
10  import stringtools.Color;
11
12  /**
13   * @author morgan
14   *
15   */
16  public class EnumTest {
17
18    @Test
19    public void testGender() {
```

```
20      Gender g = Gender.male;
21      assertEquals("male", g.toString());
22      assertEquals(g, Gender.valueOf("male"));
23    }
24
25    @Test
26    public void testColor() {
27      Color c = Color.RED;
28      assertEquals("RED(255,0,0)", c.toString());
29      assertEquals(Color.RED, c.valueOf("RED"));
30    }
31
32  }
```