Agile Java

阿左 [1]　　Nobody [2]

January 17, 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

基本工具

2.1 日期时间处理

2.1.1 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  public class CalendarExample{
8
9    public static Date createDate(int year, int month, int day){
10     Calendar cal = new GregorianCalendar();
11     cal.clear();
12     cal.set(Calendar.YEAR, year);
13     cal.set(Calendar.MONTH, month-1);
14     cal.set(Calendar.DAY_OF_MONTH, day);
15     return cal.getTime();
16   }
17
18   public static Date addDay(Date date, int dayNum) {
19     Calendar cal = new GregorianCalendar();
20     cal.setTime(date);
21     cal.add(Calendar.DAY_OF_YEAR, dayNum);
```

```
22      return cal.getTime();
23    }
24
25  }
```

2.2 文本

2.2.1 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  }
```

2.3 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  public class EnumTest {
13
14    @Test
15    public void testGender() {
16      Gender g = Gender.male;
17      assertEquals("male", g.toString());
18      assertEquals(g, Gender.valueOf("male"));
19    }
20
21    @Test
22    public void testColor() {
23      Color c = Color.RED;
24      assertEquals("RED(255,0,0)", c.toString());
25      assertEquals(Color.RED, c.valueOf("RED"));
26    }
```

```
27
28 }
```

2.4 数学

2.4.1 NaN 与无穷大

NaN 表示非数字，定义在java.lang.Float 与java.lang.Double 中。这两个类中同样还定义了正负无穷大的常量POSITIVE_INFINITY 和NEGATIVE_INFINITY。整数除以 0 会导致错误，但 double 和 float 会在数学上生产合理的无穷大。

```
1  package test;
2
3  import static org.junit.Assert.assertEquals;
4  import static org.junit.Assert.assertTrue;
5  import static org.junit.Assert.assertFalse;
6
7  import org.junit.After;
8  import org.junit.Before;
9  import org.junit.Test;
10
11 import java.lang.Double;
12
13 public class MathTest {
14
15   @Test
16   public void testNaN() {
17     // boolean express alway return false
18     assertFalse(Double.NaN > 0.0);
19     assertFalse(Double.NaN < 0.0);
20     assertFalse(Double.NaN == 0.0);
21   }
22
23   @Test
24   public void testInfinity() {
25     double x = 1.0;
26     double tolerance = 0.5;
27
28     assertEquals( Double.POSITIVE_INFINITY,
29       Double.POSITIVE_INFINITY * 100, tolerance );
```

```
30      assertEquals( Double.NEGATIVE_INFINITY,
31        Double.POSITIVE_INFINITY * -1, tolerance );
32
33      assertEquals( Double.POSITIVE_INFINITY, x / 0.0, tolerance
            );
34      assertEquals( Double.NEGATIVE_INFINITY, x /-0.0, tolerance
            );
35
36      assertEquals(  0.0, x / Double.POSITIVE_INFINITY, tolerance
             );
37      assertEquals( -0.0, x / Double.NEGATIVE_INFINITY, tolerance
             );
38      assertEquals(   x , x % Double.POSITIVE_INFINITY, tolerance
             );
39
40      assertTrue( Double.isNaN( x  % 0.0) );
41      assertTrue( Double.isNaN(0.0 / 0.0) );
42      assertTrue( Double.isNaN(0.0 % 0.0) );
43
44      assertEquals( Double.POSITIVE_INFINITY,
45        Double.POSITIVE_INFINITY / x, tolerance );
46      assertEquals( Double.NEGATIVE_INFINITY,
47        Double.NEGATIVE_INFINITY / x, tolerance );
48
49      assertTrue( Double.isNaN(Double.POSITIVE_INFINITY % x) );
50
51      assertTrue(
52          Double.isNaN(Double.POSITIVE_INFINITY /Double.
                POSITIVE_INFINITY) );
53      assertTrue(
54          Double.isNaN(Double.POSITIVE_INFINITY %Double.
                POSITIVE_INFINITY) );
55      assertTrue(
56          Double.isNaN(Double.POSITIVE_INFINITY /Double.
                NEGATIVE_INFINITY) );
57      assertTrue(
58          Double.isNaN(Double.POSITIVE_INFINITY %Double.
                NEGATIVE_INFINITY) );
59      assertTrue(
60          Double.isNaN(Double.POSITIVE_INFINITY /Double.
                POSITIVE_INFINITY) );
61      assertTrue(
```

```
62          Double.isNaN(Double.POSITIVE_INFINITY %Double.
                POSITIVE_INFINITY) );
63      assertTrue(
64          Double.isNaN(Double.POSITIVE_INFINITY /Double.
                NEGATIVE_INFINITY) );
65      assertTrue(
66          Double.isNaN(Double.POSITIVE_INFINITY %Double.
                NEGATIVE_INFINITY) );
67
68    }
69
70  }
```

2.4.2 通过位逻辑处理权限

记录权限的枚举类：

```
1  package example;
2
3  public enum UserAuth {
4    ADD(1),EDIT(2),DELETE(4),SEARCH(8);
5
6    private int mask;
7
8    UserAuth(int mask) {
9      this.mask = mask;
10   }
11
12   public int getMask() {
13     return this.mask;
14   }
15 }
```

系统管理员类：

```
1  package example;
2
3  public class SysAdmin {
4    private int authValue = 0x0;
5
6    public void setAuth(UserAuth ... flags) {
```

```
    for(UserAuth f: flags)
      this.authValue |= f.getMask();
  }

  public void unsetAuth(UserAuth ... flags) {
    for(UserAuth f: flags)
      this.authValue &= ~f.getMask();
  }

  public boolean hasAuth(UserAuth ... flags) {
    boolean r = true;
    for(UserAuth f: flags)
      if(f.getMask() != (this.authValue & f.getMask()))
      { r = false; break; }
    return r;
  }

  public int getAuthValue() {
    return this.authValue;
  }
}
```

权限判断：

```
package test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertFalse;
import org.junit.Test;
import junit.framework.JUnit4TestAdapter;

import java.util.Date;

import example.UserAuth;
import example.SysAdmin;

public class UserAuthTest {

  private static SysAdmin a = new SysAdmin();
  private static UserAuth add = UserAuth.ADD;
  private static UserAuth edit = UserAuth.EDIT;
```

```
19    private static UserAuth del = UserAuth.DELETE;
20    private static UserAuth find = UserAuth.SEARCH;
21
22    @Test
23    public void testAuth() {
24      a.setAuth(add, del);
25      assertEquals(Integer.valueOf("101",2).intValue(), a.
            getAuthValue());
26      assertTrue(  a.hasAuth(add, del)  );
27      assertFalse( a.hasAuth(edit, find) );
28      assertTrue(  a.hasAuth(add)  );
29      assertTrue(  a.hasAuth(del)  );
30      assertFalse( a.hasAuth(edit) );
31      assertFalse( a.hasAuth(find) );
32
33      a.setAuth(edit);
34      a.setAuth(find);
35      a.unsetAuth(add, del);
36      assertFalse(  a.hasAuth(add, del)  );
37      assertTrue( a.hasAuth(edit, find) );
38      assertFalse(  a.hasAuth(add)  );
39      assertFalse(  a.hasAuth(del)  );
40      assertTrue( a.hasAuth(edit) );
41      assertTrue( a.hasAuth(find) );
42
43      a.unsetAuth(edit);
44      assertFalse(  a.hasAuth(add, del, edit)  );
45      assertTrue( a.hasAuth(find) );
46      assertFalse(  a.hasAuth(add)  );
47      assertFalse(  a.hasAuth(del)  );
48      assertFalse( a.hasAuth(edit) );
49      assertTrue( a.hasAuth(find) );
50    }
51
52  }
```

2.4.3 异或操作实现奇偶检验

基本的思想就是数一下位的值为 1 的个数是奇数还是偶数：

```
1  package example;
```

```
2
3  public class ParityChecker {
4    public byte checksum(byte [] bytes) {
5      byte checksum = bytes[0];
6      for(int i=1; i<bytes.length; i++)
7        checksum ^= bytes[i];
8      return checksum;
9    }
10 }
```

更加严格的检验除了给整个字节流加一位检验以外，还给每一个字节加上一个检验位。

2.4.4 BitSet

java.util.BitSet 类封装了一个以二进制位为元素的向量，并且长度可变方便进行位操作。这个类的优点不多，但是它的范围超过 int 类的取值范围。

2.4.5 数字的不同进制显示

```
1  package test;
2
3  import static org.junit.Assert.assertEquals;
4  import org.junit.Test;
5  import junit.framework.JUnit4TestAdapter;
6
7  import java.util.Date;
8
9  import example.UserAuth;
10 import example.SysAdmin;
11
12 public class NumberStringTest {
13
14   @Test
15   public void testNumberString() {
16     assertEquals("101", Integer.toBinaryString(5));
17     assertEquals("21",  Integer.toOctalString(17));
18     assertEquals("32",  Integer.toHexString(50));
19
20     assertEquals("101", Integer.toString( 5, 2));
```

```
21      assertEquals("21",  Integer.toString(17, 8));
22      assertEquals("32",  Integer.toString(50,16));
23
24      assertEquals((int)253,  (int)Integer.decode("0xFD"));
25      assertEquals((int)253,  (int)Integer.decode("0XFD"));
26      assertEquals((int)253,  (int)Integer.decode( "#FD"));
27      assertEquals((int)15,   (int)Integer.decode( "017"));
28      assertEquals((int)10,   (int)Integer.decode(  "10"));
29
30      assertEquals((int)-253,  (int)Integer.decode("-0xFD"));
31      assertEquals((int)-253,  (int)Integer.decode("-0XFD"));
32      assertEquals((int)-253,  (int)Integer.decode( "-#FD"));
33      assertEquals((int)-15,   (int)Integer.decode( "-017"));
34      assertEquals((int)-10,   (int)Integer.decode(  "-10"));
35    }
36
37  }
```

2.4.6 随机数

Math 类提供的 random 方法返回一个从 0.0 到 1.0 之间的 double 类伪随机数。

java.util.Random 功能更全面，产生 boolean、byte、int、long、float、double、甚至高斯型结果的伪随机数。如：Random 类的nextBoolean 方法根据提供的种子（没有种子就用系统时间当种子）返回布尔型的随机数，相同的种子产相同的数字序列。

还有一个java.util.SecureRandom 类用来生成标准的、强加密的伪随机数。

Chapter 3

输入输出

3.1 字符流

```
1  package example;
2
3  import java.util.Date;
4  import java.util.Calendar;
5  import java.util.GregorianCalendar;
6
7  public class CalendarExample{
8
9    public static Date createDate(int year, int month, int day){
10     Calendar cal = new GregorianCalendar();
11     cal.clear();
12     cal.set(Calendar.YEAR, year);
13     cal.set(Calendar.MONTH, month-1);
14     cal.set(Calendar.DAY_OF_MONTH, day);
15     return cal.getTime();
16   }
17
18   public static Date addDay(Date date, int dayNum) {
19     Calendar cal = new GregorianCalendar();
20     cal.setTime(date);
21     cal.add(Calendar.DAY_OF_YEAR, dayNum);
22     return cal.getTime();
23   }
24
```

```
25 }
```

```
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 public class EnumTest {
13
14   @Test
15   public void testGender() {
16     Gender g = Gender.male;
17     assertEquals("male", g.toString());
18     assertEquals(g, Gender.valueOf("male"));
19   }
20
21   @Test
22   public void testColor() {
23     Color c = Color.RED;
24     assertEquals("RED(255,0,0)", c.toString());
25     assertEquals(Color.RED, c.valueOf("RED"));
26   }
27
28 }
```