Jeff on Java

谁给我一滴水， 我便回报他整个大海。

< 2007年12月 >

日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论

留言簿(1)

- 给我留言
- 查看公开留言
- 查看私人留言

随笔分类

- Jeff On Java 2008(4)
- Web技术(4)
- 跟老刘学Java (7)

随笔档案

- 2008年1月 (11)
- 2007年12月 (7)

文章档案

- 2007年12月 (2)

搜索

搜索

最新评论

- 1. re: 中文排序

中文排序

摘要：在Java中，对一个数组或列表(在本文中统称为集合)中的元素排序，是一个很经常的事情。好在Sun公司在Java库中实现了大部分功能。如果集合中的元素实现了Comparable接口，调用Array或Collections的静态(static)方法sort，就可以直接对集合排序。程序员用不同的方式实现了Comparator接口，就可以用各自不同的方式排序。对于包含汉字的字符串来说，排序的方式主要有两种：一种是拼音，一种是笔画。本文就讲述如何实现这两种不同的比较器(Comparator)。

作者：Jeff 发表于：2007年12月21日 11:27 最后更新于： 2007年12月21日 12:38

排序概述

在Java中，对一个数组或列表(在本文中统称为集合)中的元素排序，是一个很经常的事情。好在Sun公司在Java库中实现了大部分功能。如果集合中的元素实现了Comparable接口，调用以下的静态(static)方法，就可以直接对集合排序。

```
// 数组排序方法
// 数组中的元素可以是像int这样的原生类型(primitive type)，也可以是像String这样实现了Comparable接口的类型，这里用type表示。
java.util.Arrays.sort(type[] a);

// 列表
public static <T> void sort(List<T> list)
```

以上的这些排序方式能满足大部分应用。但集合中的元素没有实现Comparable接口，或者集合中的元素要按一种特别的方式排序，这要怎么办？Sun公司早就想到了，并在Java库中提供上面两个方法的重载。

```
// 数组排序方法。
// 数组中的元素可以是像int这样的原生类型(primitive type)，也可以是像String这样实现了Comparable接口的类型，这里用type表
```

- chinese 类貌似找不到呢，在线等.....
  - --zhenshao
- 2. re: 中文排序 - 汉语拼音[未登录]
- 很干净
飞过海
啊啊
  - --111
- 3. re: 中文排序[未登录]
- 效率很低啊，怎么办呢
  - --andy
- 4. re: 中文排序[未登录]
- @vv0885
人家这个类是在下载那个sourceforge中的jar包的哦。
  - --xx
- 5. re: 中文排序
- 很不错 值得一看
  - --徐冬冬

示。

```
public static <T> void sort(T[] a, Comparator<? super T> c)

// 列表
public static <T> void sort(List<T> list, Comparator<? super T> c)
```

只要实现了Comparator接口，就可以按程序员自己的意思去排序了。对于包含汉字的字符串来说，排序的方式主要有两种：一种是拼音，一种是笔画。汉字是通过一定的编码方式存储在计算机上的，主要的编码有：Unicdoe、GB2312和GBK等。

Unicode 编码中的汉字

Unicode中编码表分为两块，一个是基本的，一个是辅助的。现在的大多数操作系统还不支持Unicode中辅助区域中的文字，如WinXp。

在Java中的字符就是Unicode码表示的。对于Unicode基本区域中的文字，用两个字节的内存存储，用一个char表示，而辅助区域中的文字用4个字节存储，因此辅助区域中的就要用两个char来表示了(表一种蓝色底就是辅助区域中的文字)。一个文字的unicode编码，在Java中统一用codePoint(代码点)这个概念。

中文和日文、韩文一样是表意文字，在Unicode中，中日韩三国(东亚地区)的文字是统一编码的。CJK代表的就是中日韩。在这里，我把这3中文字，都作为汉字处理了。(日语和韩语可能就是从汉语中衍生的吧！)

汉字在Unicode中的分布大致如下表：

	首字编码	尾字编码	个数
基本汉字	U4E00	U9FBF	20928
异性字	UF900	UFAFF	512
扩展A	U3400	U4D8F	512
扩展B	U20000	U2A6DF	42720
补充	U2F800	U2FA1F	544
其他			...

表一

在这些编码区间，有些编码是保留的。

GB2312编码

GB2312是中华人民共和国最早的计算机汉字编码方式。大概有6000多个汉字，这些汉字是按拼音顺序编码的。这6000多个汉字都是简体中文字。

GBK编码

GB2312的扩展，并兼容GB2312。扩展后的汉字大概有2万多个，其中有简体汉字也有繁体汉字。

拼音排序

拼音有好几种方式，其中最主要的是中华人民共和国的汉语拼音 Chinese Phonetic。对汉字的排序有两种：一种是宽松的，能够按拼音排序最常用的汉字，另一种是严格的，能够按拼音排序绝大部分大部分汉字。

宽松的拼音排序法

原理：汉字最早是GB2312编码，收录了六千多个汉字，是按拼音排序的，编码是连续的。 后来出现了GBK编码，对GB2312进行了扩展，到了两万多汉字，并且兼容GB2312，也就是说GB2312中的汉字编码是原封不动搬到GBK中的(在GBK编码中[B0-D7]区中)。

如果我们只关心这6000多个汉字的顺序，就可以用下面的方法实现汉字宽松排序。

```
/**
* @author Jeff
*
* Copyright (c) 复制或转载本文，请保留该注释。
*/

package chinese.utility;

import java.text.Collator;
import java.util.Comparator;
import java.util.Locale;

public class PinyinSimpleComparator implements
Comparator<String> {
    public int compare(String o1, String o2) {
        return
Collator.getInstance(Locale.CHINESE).compare(o1, o2);
    }
}
```

在对[孙, 孟, 宋, 尹, 廖, 张, 徐, 昆, 曹, 曾,怡]这几个汉字排序，结果是：[曹, 昆, 廖, 孟, 宋, 孙, 徐, 尹, 曾, 张, 怡]。最后一个 **怡** 有问题，不该排在最后的。

注意：这个程序有两个不足

- 由于gb2312中的汉字编码是连续的，因此新增加的汉字不可能再按照拼音顺序插入到已有的gb2312编码中，所以新增加的汉字不是按拼音顺序排的。
- 同音字比较的结果不等于0 。

下面的测试代码可以证明

```
/**
* @author Jeff
*
* Copyright (c) 复制或转载本文，请保留该注释。
*/

/**
* 非常用字(怡)
*/
@Test
public void testNoneCommon() {
    Assert.assertTrue(comparator.compare("怡", "张") >
0);
}

/**
* 同音字
*/
@Test
```

```
public void testSameSound() {
    Assert.assertTrue(comparator.compare("怕", "帕") !=
0);
}
```

严格的拼音排序法

为了解决宽松的拼音的两点不足，可以通过实现汉语拼音的函数来解决。goolge下看到sf上有个pinyin4j的项目，可以解决这个问题，pinyin4j的项目地址是：http://pinyin4j.sourceforge.net/。

实现代码：

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility;

import java.util.Comparator;
import net.sourceforge.pinyin4j.PinyinHelper;

public class PinyinComparator implements
Comparator<String> {

    public int compare(String o1, String o2) {

        for (int i = 0; i < o1.length() && i <
o2.length(); i++) {

            int codePoint1 = o1.charAt(i);
            int codePoint2 = o2.charAt(i);

            if
(Character.isSupplementaryCodePoint(codePoint1)
                    ||
Character.isSupplementaryCodePoint(codePoint2)) {
                i++;
            }

            if (codePoint1 != codePoint2) {
                if
(Character.isSupplementaryCodePoint(codePoint1)
                        ||
Character.isSupplementaryCodePoint(codePoint2)) {
                    return codePoint1 - codePoint2;
                }

                String pinyin1 = pinyin((char)
codePoint1);
                String pinyin2 = pinyin((char)
codePoint2);

                if (pinyin1 != null && pinyin2 !=
null) { // 两个字符都是汉字
                    if (!pinyin1.equals(pinyin2)) {
                        return
pinyin1.compareTo(pinyin2);
                    }
                } else {
```

```
                    return codePoint1 - codePoint2;
                }
            }
        }
        return o1.length() - o2.length();
    }

    /**
     * 字符的拼音，多音字就得到第一个拼音。不是汉字，就return null。
     */
    private String pinyin(char c) {
        String[] pinyins = PinyinHelper.toHanyuPinyinStringArray(c);
        if (pinyins == null) {
            return null;
        }
        return pinyins[0];
    }
}
```

测试：

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility.test;

import java.util.Comparator;

import org.junit.Assert;
import org.junit.Test;

import chinese.utility.PinyinComparator;

public class PinyinComparatorTest {

    private Comparator<String> comparator = new PinyinComparator();

    /**
     * 常用字
     */
    @Test
    public void testCommon() {
        Assert.assertTrue(comparator.compare("孟", "宋") < 0);
    }

    /**
     * 不同长度
     */
    @Test
    public void testDifferentLength() {
        Assert.assertTrue(comparator.compare("他奶奶的", "他奶奶的熊") < 0);
    }
```

```
    /**
     * 和非汉字比较
     */
    @Test
    public void testNoneChinese() {
        Assert.assertTrue(comparator.compare("a", "阿")
< 0);
        Assert.assertTrue(comparator.compare("1", "阿")
< 0);
    }

    /**
     * 非常用字(怡)
     */
    @Test
    public void testNoneCommon() {
        Assert.assertTrue(comparator.compare("怡",
"张") < 0);
    }

    /**
     * 同音字
     */
    @Test
    public void testSameSound() {
        Assert.assertTrue(comparator.compare("怕",
"帕") == 0);
    }

    /**
     * 多音字(曾)
     */
    @Test
    public void testMultiSound() {
        Assert.assertTrue(comparator.compare("曾经",
"曾迪") > 0);
    }

}
```

我的这样严格的拼音排序还是有有待改进的地方，看上面测试代码的最后一个测试，就会发现：程序不会根据语境来判断多音字的拼音，仅仅是简单的取多音字的第一个拼音。

笔画排序

要按笔画排序，就要实现笔画比较器。

```
class StokeComparator implements Comparator<String>
```

如果有个方法可以求得汉字的笔画数，上面的功能就很容易实现。如何求一个汉字的笔画数？最容易想到的就是查表法。建一个汉字笔画数表，如：

汉字	Unicode编码	笔画数
一	U4E00	1
二	U4E8C	2
龍	U9F8D	16
…	…	…

表二

如果是连续的、按unicode编码排好顺序的表，实际存储在笔画数表中的只需最后一列就够了。

那如何建这个表呢？这个表存储在哪里？

建汉字笔画数表

现在大多数系统还只能支持Unicode中的基本汉字那部分汉字，编码从U9FA6-U9FBF。所以我们只建这部分汉字的笔画表。汉字笔画数表，我们可以按照下面的方法生成：

- 用java程序生成一个文本文件(Chinese.csv)。包括所有的从U9FA6-U9FBF的字符的编码和文字。利用excel的按笔画排序功能，对Chinese.csv文件中的内容排序。
- 编写Java程序分析Chinese.csv文件，求得笔画数, 生成ChineseStroke.csv。矫正笔画数，重新按汉字的Unicode编码对ChineseStroke.csv文件排序。
- 只保留ChineseStroke.csv文件的最后一列，生成Stroke.csv。

在这里下载上面3个步骤生成的3个文件。

生成Chinese.csv的Java程序

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility.preface;

import java.io.IOException;
import java.io.PrintWriter;

public class ChineseCoder {

    public static void main(String[] args) throws
IOException {
        PrintWriter out = new
PrintWriter("Chinese.csv");
        // 基本汉字
        for(char c = 0x4E00; c <= 0x9FA5; c++) {
            out.println((int)c + "," + c);
        }
        out.flush();
        out.close();

    }

}
```

初始化笔画数

从Excel排序过后的Chinese.csv文件来看，排好序的文件还是有一定规律的。在文件的第9行-12行可以看出：逐行扫描的时候，当unicode会变小了，笔画数也就加1。

20059,乛
20101,丿
19969,丁
19970,丂

用下面的Java程序分析吧。

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility.preface;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Stroke {

    /**
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws
IOException {
        Scanner in = new Scanner(new
File("Chinese.csv"));
        PrintWriter out = new
PrintWriter("ChineseStroke.csv");
        String oldLine = "999999";
        int stroke = 0;
        while (in.hasNextLine()) {
            String line = in.nextLine();
            if (line.compareTo(oldLine) < 0) {
                stroke++;
            }
            oldLine = line;
            out.println(line + "," +
stroke);
        }
        out.flush();
        out.close();
        in.close();
    }

}
```

上面用的这个规律有问题吗？有问题，从ChineseStroke.csv文件抽取最后几个汉字就发现，笔画数不对。为什么呢？

- 笔画数可能不是连续的。
- n+1笔画数的最小Unicode码可能比n笔画数的最大Unicode码要大

我们要人工核对ChineseStroke文件，但只要核对在笔画变化的那几个汉字的笔画数。最后，我发现，只有笔画数多于30的少数几个汉字的笔画数不对。核对并矫正笔画数后，用Excel按Unicode重新排序，去掉汉字和Unicode两列，只保留笔画数那列，得到Stroke.csv文件。

求得笔画数的方法和笔画比较器方法

求得笔画数的方法测试代码：

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility.test;

import static org.junit.Assert.assertEquals;

import org.junit.Before;
import org.junit.Test;
import chinese.utility.Chinese;

public class StrokeTest {

    Chinese chinese;

    @Before
    public void setUp() {
        chinese = new Chinese();
    }

    @Test
    public void testStroke() {
        assertEquals(1, chinese.stroke('一'));
    }

    @Test
    public void testStroke2() {
        assertEquals(2, chinese.stroke('二'));
    }

    @Test
    public void testStroke16() {
        assertEquals(16, chinese.stroke('龍'));
    }

    @Test
    public void testStrokeABC() {
        assertEquals(-1, chinese.stroke('a'));
    }

}
```

求得笔画数的方法代码

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility;

import java.util.Comparator;

public class StrokeComparator implements
Comparator<String> {

    public int compare(String o1, String o2) {

        Chinese chinese = new Chinese();
```

```
        for (int i = 0; i < o1.length() && i <
o2.length(); i++) {
            int codePoint1 = o1.codePointAt(i);
            int codePoint2 = o2.codePointAt(i);
            if (codePoint1 == codePoint2)
                continue;

            int stroke1 = chinese.stroke(codePoint1);
            int stroke2 = chinese.stroke(codePoint2);

            if (stroke1 < 0 || stroke2 < 0) {
                return codePoint1 - codePoint2;
            }

            if (stroke1 != stroke2) {
                return stroke1 - stroke2;
            }
        }

        return o1.length() - o2.length();
    }
}
```

笔画比较器测试

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility.test;

import java.util.Comparator;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;

import chinese.utility.StrokeComparator;

public class StrokeComparatorTest {

    private Comparator<String> comparator;
    @Before
    public void setUp() {
        comparator = new StrokeComparator();
    }

    /**
     * 相同笔画数
     */
    @Test
    public void testCompareEquals() {
        Assert.assertTrue(comparator.compare("一",
"丨") == 0);
    }
    /**
     * 不同笔画数
     */
    @Test
    public void testCompare() {
```

```
        Assert.assertTrue(comparator.compare("一",
"二") < 0);
        Assert.assertTrue(comparator.compare("唔",
"马") > 0);
    }
    /**
     * 长度不同
     */
    @Test
    public void testCompareDefficultLength() {
        Assert.assertTrue(comparator.compare("二", "二
一") < 0);
    }
    /**
     * 非汉字的比较
     */
    @Test
    public void testABC() {
        Assert.assertTrue(comparator.compare("一", "a")
> 0);
        Assert.assertTrue(comparator.compare("a", "b")
< 0);
    }
}
```

笔画比较器

```
/**
  * @author Jeff
  *
  * Copyright (c) 复制或转载本文，请保留该注释。
  */
package chinese.utility.test;

import java.util.Comparator;

import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;

import chinese.utility.StrokeComparator;

public class StrokeComparatorTest {

    private Comparator<String> comparator;
    @Before
    public void setUp() {
        comparator = new StrokeComparator();
    }

    /**
     * 相同笔画数
     */
    @Test
    public void testCompareEquals() {
        Assert.assertTrue(comparator.compare("一",
"丨") == 0);
    }
    /**
     * 不同笔画数
```

```
     */
    @Test
    public void testCompare() {
        Assert.assertTrue(comparator.compare("一",
"二") < 0);
        Assert.assertTrue(comparator.compare("唔",
"马") > 0);
    }
    /**
     * 长度不同
     */
    @Test
    public void testCompareDefficultLength() {
        Assert.assertTrue(comparator.compare("二", "二
一") < 0);
    }
    /**
     * 非汉字的比较
     */
    @Test
    public void testABC() {
        Assert.assertTrue(comparator.compare("一", "a")
> 0);
        Assert.assertTrue(comparator.compare("a", "b")
< 0);
    }
}
```

其他程序的汉字排序

Microsoft在这方面做得比较好。如Sql server 2000，Word和Excel都能按拼音和笔画排序。而Oracle只能是采取宽松拼音排序法。

posted on 2007-12-21 11:29 Jeff Lau 阅读(9390) 评论(10)
编辑 收藏 所属分类: 跟老刘学Java

评论:

re: 中文排序 2007-12-21 12:42 | sitinspring
好文章! 回复 更多评论

re: 中文排序 2007-12-23 10:47 | ci
好..... 回复 更多评论

re: 中文排序 2008-01-03 18:52 | hill911
好文章
值得研究 回复 更多评论

re: 中文排序 2008-08-12 17:35 | vv0885

长知识，谢谢前辈！

但是我没有找到Chinese这个类！ 回复 更多评论

re: 中文排序 2008-12-02 11:48 | Arix
笔画排序的资料帮了很大的忙，感谢。 回复 更多评论

re: 中文排序 2011-01-10 09:52 | 唐永军
谢谢。学习了。
回复 更多评论

re: 中文排序 2011-08-01 22:59 | 徐冬冬
很不错 值得一看 回复 更多评论

re: 中文排序[未登录] 2011-10-19 12:59 | xx
@vv0885
人家这个类是在下载那个sourceforge中的jar包的哦。 回复 更多评论

re: 中文排序[未登录] 2011-11-07 17:36 | andy
效率很低啊，怎么办呢 回复 更多评论

re: 中文排序 2013-02-21 16:57 | zhenshao
chinese 类貌似找不到呢，在线等..... 回复 更多评论

新用户注册 刷新评论列表

只有注册用户登录后才能发表评论。

网站导航:
博客园 IT新闻 知识库 C++博客 博问 管理
相关文章:
Java多线程编程(二)
Java多线程编程(一)
中文数字
中文排序 - 笔画
中文排序 - 汉语拼音
跟老刘学Java (一)