



Java 核心技术(进阶)

第三章 高级文本处理

第三节 Java高级字符串处理

华东师范大学 陈良育

大纲



- 正则表达式
- 其他字符串操作
 - 集合和字符串互转
 - 字符串转义
 - 变量名字格式化
 - 从字符串到输入流



正则表达式(1)

- 如何识别给定字符串为合法的邮箱地址

- a@b.com ✓
- a@@b.com ✗
- a@b ✗
- @a.com ✗
- a@b@c.com ✗

- 如何认定一个字符串满足一定的规律



正则表达式(2)

- 正则表达式(Regular Expression)
 - 规则表达式，计算机科学的一个基础概念
 - 用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”
 - $^[A-Za-z]^+$ ，代表着一个字符串，只能由26英文字母组成
 - 作用
 - 测试字符串内的模式
 - 识别/替换文本
 - 提取文本



正则表达式(3)

- 正则表达式独立于特定语言 (Java, Perl, Python, PHP...)
- 正则表达式的匹配模板
 - 定界符
 - 原子
 - 特殊功能字符(元字符)
 - 模式修正符
- 推荐书籍：精通正则表达式(第三版)，Jeffrey E.F.Friedl著，余晟翻译，电子工业出版社，2012.



Java的正则表达式



- java.util.regex 包
 - **Pattern** 正则表达式的编译表示
 - compile 编译一个正则表达式为Pattern对象
 - matcher 用Pattern对象匹配一个字符串，返回匹配结果
 - **Matcher**
 - Index Methods(位置方法) // start(), start(int group), end(), end(int group)
 - Study Methods(查找方法) // lookingAt(), find(), find(int start), matches()
 - Replacement Methods(替换方法) //replaceAll(String replacement)
- 查看例子



其他字符串操作

- 字符串和集合互转
 - $[1,2,3]$, “1,2,3”
- 字符串转义
 - 对关键字符转义
- 变量名字格式化
 - 名字驼峰命名
- 字符串输入流
 - 将字符串转为一个输入流
 - 输入流可以定义为Scanner, 这是Online Judge的实现原理

总结



- 灵活使用正则表达式
- 多使用第三方库：Apache Commons Lang. Guava等
 - github.com
 - mvnrepository.com
 - www.open-open.com
 -



代码(1) MatcherDemo.java

```
public class MatcherDemo {  
  
    private static final String REGEX = "\\bdog\\b"; //\b表示边界  
    private static final String INPUT = "dog dog dog doggie dogg";  
  
    public static void main(String[] args) {  
        //检查字符串里面有多少个dog  
        Pattern p = Pattern.compile(REGEX);  
  
        Matcher m = p.matcher(INPUT);  
        int count = 0;  
        while (m.find()) {  
            count++;  
            System.out.println("Match number " + count);  
            System.out.println("start(): " + m.start());  
            System.out.println("end(): " + m.end());  
        }  
    }  
}
```



代码(2) MatchesLooking.java

```
public class MatchesLooking {  
  
    private static final String REGEX = "foo";  
    private static final String INPUT =  
        "foo";  
    private static Pattern pattern;  
    private static Matcher matcher;  
  
    public static void main(String[] args) {  
  
        // Initialize  
        pattern = Pattern.compile(REGEX);  
        matcher = pattern.matcher(INPUT);  
  
        System.out.println("Current REGEX is: "  
            + REGEX);  
        System.out.println("Current INPUT is: "  
            + INPUT);  
  
        System.out.println("lookingAt(): "  
            + matcher.lookingAt()); //部分匹配  
        System.out.println("matches(): "  
            + matcher.matches()); //完全匹配  
    }  
}
```



代码(3) RegexDemo.java

```
public class RegexDemo {  
  
    private static String REGEX = "a*b"; /*表示限定前面的a可以有0或者多个。  
    private static String INPUT = "aabfooaabfoaabfoobcdd";  
    private static String REPLACE = "-";  
  
    public static void main(String[] args) {  
        Pattern p = Pattern.compile(REGEX);  
        Matcher m = p.matcher(INPUT); // get a matcher object  
        StringBuffer sb = new StringBuffer();  
        //全部替换  
        while(m.find()){  
            m.appendReplacement(sb, REPLACE);  
        }  
        //将最后的尾巴字符串附上  
        m.appendTail(sb);  
        System.out.println(sb.toString());  
    }  
}
```




代码(4) ReplaceDemo.java

```
public class ReplaceDemo {  
  
    private static String REGEX = "dog";  
    private static String INPUT =  
        "The dog says meow. All dogs say meow.";  
    private static String REPLACE = "cat";  
  
    public static void main(String[] args) {  
        Pattern p = Pattern.compile(REGEX);  
        // get a matcher object  
        Matcher m = p.matcher(INPUT);  
        INPUT = m.replaceAll(REPLACE); //把所有的dog都换成cat  
        System.out.println(INPUT);  
    }  
}
```



代码(5) ReplaceDemo2.java

```
public class ReplaceDemo2 {  
  
    private static String REGEX = "a*b"; // *表示限定前面的a可以有0或者多个。  
    private static String INPUT =  
        "aabfooaabfooabfoob";  
    private static String REPLACE = "-";  
  
    public static void main(String[] args) {  
        Pattern p = Pattern.compile(REGEX);  
        // get a matcher object  
        Matcher m = p.matcher(INPUT);  
        INPUT = m.replaceAll(REPLACE);  
        System.out.println(INPUT);  
    }  
}
```




代码(6) RegexTest.java

```
public class RegexTest {  
  
    public static void main(String[] args) {  
  
        regularExpression();  
    }  
  
    public static void regularExpression() {  
        String REGEX_EMAIL = "^\\w+((-\\w+)|(\\.\\w+))*@[A-Za-z0-9]+((\\.\\w+)[A-Za-z0-9]+)*\\.([A-Za-z0-9]+)$";  
        Pattern pattern = Pattern.compile(REGEX_EMAIL);  
  
        String[] emails = new String[]{"123^@qq.com", "name_321@163.com", "+whatever*72@gmail.com"};  
        for(String email : emails) {  
            Matcher matcher = pattern.matcher(email);  
            if (matcher.matches()) {  
                System.out.println(email + " is valid email.");  
            } else {  
                System.out.println(email + " is not valid email.");  
            }  
        }  
    }  
}
```

代码(7) String2List.java



```
public class String2List {  
    public static void main(String[] args) {  
        List<String> names = new LinkedList<String>();  
        names.add("Xiaohong");  
        names.add("Xiaoming");  
        names.add("Daming");  
        names.add("Xiaohei");  
  
        //从ArrayList变到字符串  
        String str1 = String.join(",", names); //String.join, JDK 8 引入  
        System.out.println(str1);  
  
        String str2 = StringUtils.join(names, ","); //Apache Commons Lang  
        System.out.println(str2);  
  
        //从字符串变回ArrayList  
        List<String> names2 = Arrays.asList(str2.split(","));  
        for(String name:names2)  
        {  
            System.out.println(name);  
        }  
  
        //StringUtils 可以支持更多数据类型  
        List<Integer> ids = new ArrayList<Integer>();  
        ids.add(1);  
        ids.add(3);  
        ids.add(5);  
        String str3 = StringUtils.join(ids, ",");  
        System.out.println(str3);  
    }  
}
```



代码(8) EscapeString.java

```
public class EscapeString {  
  
    public static void main(String[] args) {  
        String str = "He didn't say, \"Stop!\"";  
  
        //转义  
        String escapedStr = StringEscapeUtils.escapeJava(str);  
        System.out.println("escape" + ":" + escapedStr);  
  
        //从转义字符串转回来  
        String str2 = StringEscapeUtils.unescapeJava(escapedStr);  
        System.out.println("unescape" + ":" + str2);  
    }  
}
```




代码(9) GuavaUtil.java

```
public class GuavaUtil {  
    public static void main(String[] args) throws Exception {  
        List<Integer> list = new ArrayList<Integer>();  
        list.add(123);  
        list.add(456);  
        System.out.println(list);  
  
        //直接初始化List数组  
        List<Integer> integers = Lists.newArrayList(123, 456);  
        System.out.println(integers);  
  
        //拆分字符串，忽略空字符串  
        Iterable<String> split = Splitter.on(',')  
            .trimResults()  
            .omitEmptyStrings()  
            .split("123,321,, abc");  
  
        for (String s : split) {  
            System.out.println(s);  
        }  
    }  
}
```



代码(10) GuavaUtil.java



```
String a = "123,321,, abc";
String[] as = a.split(",");

for(int i=0;i<as.length;i++)
{
    if(null == as[i] || as[i].length()<=0)
    {
        continue;
    }
    else
    {
        System.out.println(as[i].trim());
    }
}
}
```


代码(11) String2InputStream.java



```
public class String2InputStream {  
  
    public static void main(String[] args) {  
        //构造字符串列表  
        List<String> names = new LinkedList<String>();  
        names.add("Xiaohong");  
        names.add("Xiaoming");  
        names.add("Daming");  
        names.add("Xiaohei");  
  
        //合并为一个字符串，以逗号相连  
        String nameStr = String.join(",", names);  
  
        //将字符串作为默认的输入流  
        InputStream in = IOUtils.toInputStream(nameStr, Charsets.toCharset("UTF-8"));  
        //重置系统的输入流  
        System.setIn(in);  
  
        //模拟键盘输入 这也是OJ平台测试用例输入的原理  
        //此处也可以换成一个文件输入流  
        Scanner sc = new Scanner(System.in);  
        sc.useDelimiter(",");  
        while(sc.hasNext())  
        {  
            System.out.println(sc.next());  
        }  
    }  
}
```



谢谢!