Java Properties类使用详解

概述

Properties 继承于 Hashtable。表示一个持久的属性集，属性列表以key-value的形式存在，key和value都是字符串。
Properties 类被许多Java类使用。例如，在获取环境变量时它就作为System.getProperties()方法的返回值。
我们在很多**需要避免硬编码的应用场景**下需要使用properties文件来加载程序需要的配置信息，比如JDBC、MyBatis框架等。Properties类则是properties文件和程序的中间桥梁，不论是从properties文件读取信息还是写入信息到properties文件都要经由Properties类。

常见方法

除了从Hashtable中所定义的方法，Properties定义了以下方法：

序号	方法描述
1	**String getProperty(String key)** 用指定的键在此属性列表中搜索属性。
2	**String getProperty(String key, String defaultProperty)** 用指定的键在属性列表中搜索属性。
3	**void list(PrintStream streamOut)** 将属性列表输出到指定的输出流。
4	**void list(PrintWriter streamOut)** 将属性列表输出到指定的输出流。
5	**void load(InputStream streamIn) throws IOException** 从输入流中读取属性列表（键和元素对）。
6	**Enumeration propertyNames( )** 按简单的面向行的格式从输入字符流中读取属性列表（键和元素对）。
7	**Object setProperty(String key, String value)** 调用 Hashtable 的方法 put。
8	**void store(OutputStream streamOut, String description)** 以适合使用 load(InputStream)方法加载到 Properties 表中的格式，将此 Properties 表中的属性列表（键和元素对）写入输出流。

Properties类

下面我们从**写入、读取、遍历**等角度来解析Properties类的常见用法：

写入

Properties类调用setProperty方法将键值对保存到内存中，此时可以通过getProperty方法读取，propertyNames方法进行遍历，但是并没有将键值对持久化到属性文件中，故需要调用store方法持久化键值对到属性文件中。

```
package cn.habitdiary;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Date;
import java.util.Enumeration;
import java.util.Properties;

import junit.framework.TestCase;

public class PropertiesTester extends TestCase {

    public void writeProperties() {
        Properties properties = new Properties();
        OutputStream output = null;
        try {
            output = new FileOutputStream("config.properties");
            properties.setProperty("url", "jdbc:mysql://localhost:3306/");
            properties.setProperty("username", "root");
            properties.setProperty("password", "root");
            properties.setProperty("database", "users");//保存键值对到内存
            properties.store(output, "Steven1997 modify" + new
Date().toString());
                        // 保存键值对到文件中
        } catch (IOException io) {
            io.printStackTrace();
        } finally {
            if (output != null) {
                try {
                    output.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

读取

下面给出常见的六种读取properties文件的方式：

```
package cn.habitdiary;

```

```java
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Locale;
import java.util.Properties;
import java.util.PropertyResourceBundle;
import java.util.ResourceBundle;

/**
 * 读取properties文件的方式
 *
 */
public class LoadPropertiesFileUtil {

    private static String basePath =
"src/main/java/cn/habitdiary/prop.properties";
    private static String path = "";

    /**
     * 一、 使用java.util.Properties类的load(InputStream in)方法加载properties
文件
     *
     * @return
     */
    public static String getPath1() {

        try {
            InputStream in = new BufferedInputStream(new FileInputStream(
                    new File(basePath)));
            Properties prop = new Properties();

            prop.load(in);

            path = prop.getProperty("path");

        } catch (FileNotFoundException e) {
            System.out.println("properties文件路径书写有误，请检查！");
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return path;
    }

    /**
     * 二、 使用java.util.ResourceBundle类的getBundle()方法
     * 注意：这个getBundle()方法的参数只能写成包路径+properties文件名，否则将抛异常
     *
     * @return
     */
    public static String getPath2() {
        ResourceBundle rb = ResourceBundle
                .getBundle("cn/habitdiary/prop");
        path = rb.getString("path");
```

```
59          return path;
60      }
61
62      /**
63       * 三、 使用java.util.PropertyResourceBundle类的构造函数
64       *
65       * @return
66       */
67      public static String getPath3() {
68          InputStream in;
69          try {
70              in = new BufferedInputStream(new FileInputStream(basePath));
71              ResourceBundle rb = new PropertyResourceBundle(in);
72              path = rb.getString("path");
73          } catch (FileNotFoundException e) {
74              // TODO Auto-generated catch block
75              e.printStackTrace();
76          } catch (IOException e) {
77              e.printStackTrace();
78          }
79          return path;
80      }
81
82      /**
83       * 四、 使用class变量的getResourceAsStream()方法
84       * 注意：getResourceAsStream()方法的参数按格式写到包路径+properties文件名+.后
   缀
85       *
86       * @return
87       */
88      public static String getPath4() {
89          InputStream in = LoadPropertiesFileUtil.class
90                  .getResourceAsStream("cn/habitdiary/prop.properties");
91          Properties p = new Properties();
92          try {
93              p.load(in);
94              path = p.getProperty("path");
95          } catch (IOException e) {
96              e.printStackTrace();
97          }
98          return path;
99      }
100
101     /**
102      * 五、
103      * 使用class.getClassLoader()所得到的java.lang.ClassLoader的
104      * getResourceAsStream()方法
105      * getResourceAsStream(name)方法的参数必须是包路径+文件名+.后缀
106      * 否则会报空指针异常
107      * @return
108      */
109     public static String getPath5() {
110         InputStream in = LoadPropertiesFileUtil.class.getClassLoader()
111                 .getResourceAsStream("cn/habitdiary/prop.properties");
112         Properties p = new Properties();
113         try {
114             p.load(in);
115             path = p.getProperty("path");
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        return path;
    }

    /**
     * 六、 使用java.lang.ClassLoader类的getSystemResourceAsStream()静态方法
     * getSystemResourceAsStream()方法的参数格式也是有固定要求的
     *
     * @return
     */
    public static String getPath6() {
        InputStream in = ClassLoader

.getSystemResourceAsStream("cn/habitdiary/prop.properties");
        Properties p = new Properties();
        try {
            p.load(in);
            path = p.getProperty("path");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return path;
    }

    public static void main(String[] args) {
        System.out.println(LoadPropertiesFileUtil.getPath1());
        System.out.println(LoadPropertiesFileUtil.getPath2());
        System.out.println(LoadPropertiesFileUtil.getPath3());
        System.out.println(LoadPropertiesFileUtil.getPath4());
        System.out.println(LoadPropertiesFileUtil.getPath5());
        System.out.println(LoadPropertiesFileUtil.getPath6());
    }
}
```

其中第一、四、五、六种方式都是先获得文件的输入流，然后通过Properties类的load(InputStream inStream)方法加载到Properties对象中，最后通过Properties对象来操作文件内容。
第二、三中方式是通过ResourceBundle类来加载Properties文件，然后ResourceBundle对象来操做properties文件内容。
其中最重要的就是每种方式加载文件时，文件的路径需要按照方法的定义的格式来加载，否则会抛出各种异常，比如空指针异常。

遍历

下面给出四种遍历Properties中的所有键值对的方法：

```
    /**
     * 输出properties的key和value
     */
    public static void printProp(Properties properties) {
        System.out.println("---------（方式一）------------");
        for (String key : properties.stringPropertyNames()) {
            System.out.println(key + "=" + properties.getProperty(key));
```

```
        }

        System.out.println("---------（方式二）------------");
        Set<Object> keys = properties.keySet();//返回属性key的集合
        for (Object key : keys) {
            System.out.println(key.toString() + "=" + properties.get(key));
        }

        System.out.println("---------（方式三）------------");
        Set<Map.Entry<Object, Object>> entrySet = properties.entrySet();
        //返回的属性键值对实体
        for (Map.Entry<Object, Object> entry : entrySet) {
            System.out.println(entry.getKey() + "=" + entry.getValue());
        }

        System.out.println("---------（方式四）------------");
        Enumeration<?> e = properties.propertyNames();
        while (e.hasMoreElements()) {
            String key = (String) e.nextElement();
            String value = properties.getProperty(key);
            System.out.println(key + "=" + value);
        }
    }
```