Properties类是在学Java基础时接触到的，只是从来没用过，这也是第一次遇到，因此在这儿在复习一下：

有这么几个问题：

一、Properties类是什么，用来干嘛的？

二、如何使用Properties呢？

针对两面两个问题，具体来看：

## 一、Properties类是什么，用来干嘛的：

Properties属性文件在JAVA应用程序中是经常可以看得见的，也是特别重要的一类文件。它用来配置应用程序的一些信息，不过这些信息一般都是比较少的数据，没有必要使用数据库文件来保存，而使用一般的文本文件来保存，如果是通过File直接保存的话，可能在存储和读取上都不是很方便，但如果保存为Properties文件就不一样了，属性文件都有键值对应的，在JAVA的包中，有提供专门的操作属性文件的类。这个类就是 java.uitl.Properties类，由于Properties类是一个集合类，所以，Properties会将属性以集合的方式读写。

Properties类继承自Hashtable类，采用键值对的存储方式，在使用Properties类管理属性文件时有什么方便的呢？Properties类有专门的读写方法来读写Properties属性文件，不用担心读写的格式问题，只要为Properties类提供一个读写流即可。Properties用于读写属性文件的方法分别是：

Java代码

```
1       //读取属性文件流的方法
2         public void load(InputStream inStream) throws IOException {}
3         //写属性文件流的方法
4   public void store(OutputStream out, String comments) throws IOExcepti
on {}
```

## 二、如何使用Properties呢?

首先，我们来看看如何从一个属性文件中读取属性。

假定我们已经新建了一个属性文件，名为prop.properties，内容如下:

Java代码

```
1    sitename=abcjava
2    siteurl=www.abcjava.com
```

我们要做的第一步就是要将文件读取到Properties类对象中，由于load有一个参数是InputStream，所以我们可以用 InputStream的子类FileInputStream将属性文件读取到Properties对象中，知道prop.properties的路径，我们就用FileInputStream(String name)构造函数:

Java代码

```
1    Properties prop = new Properties();//属性集合对象
2      FileInputStream fis = new FileInputStream("prop.properties");//
属性文件流
3      prop.load(fis);//将属性文件流装载到Properties对象中
```

接下来我们将做的事情就是如果读取一个属性，因为属性文件中的每一行都是一个键值对应，所以每一行都代表了一个属性对象，每一行都将以键和值的关系存储到Properties中，Properties类提供了getProperty(String key)方法用来通过键名读取键值，当key在属性集合中找不到时又想为key在程序中赋予一个值时可以使用public String getProperty(String key, String defaultValue)方法，这个方法的意思就是用指定的键在属性列表中搜索属性。如果在属性列表中未找到该键，则接着递归检查默认属性列表及其默认值。如果未找到属性，则此方法返回默认值变量:

Java代码

```
1    //获取属性值，sitename已在文件中定义
```

```
2        System.out.println("获取属性值:
sitename=" + prop.getProperty("sitename"));
3        //获取属性值，country未在文件中定义，将在此程序中返回一个
默认值，但并不修改属性文件
4        System.out.println("获取属性值:
country=" + prop.getProperty("country", "中国"));
```

在知道怎么读取属性文件之后我们还有一个很重要的事情就是要修改和添加新的属性到属性文件，这里就是使用public void store(OutputStream out, String comments)方法，这个方法是将属性集合写到一个OutputStream流中，同InputStream流一样，这里同样是使用其子类 FileOutputStream(String name)，这里就不多说了。

在保存属性集合到文件之前，我们还有一件事情就是如何修改和添加新的属性到属性集合，这里使用了一个方法就是setProperty(String key, String value)，这个方法就是当属性集合中存在指定的key时，就修改这个key的值，如果不存在，就新建一个key，同样是通过键值关系保存的，但值得注意的是，Properties类继承自Hashtable，所以也可以用Hashtable的put和putAll方法保存，但强烈反对使用这两个方法，因为它们允许调用方插入其键或值不是 Strings 的项。相反，应该使用 setProperty 方法。如果在“有危险”的 Properties 对象（即包含非 String 的键或值）上调用 store 或 save 方法，则该调用将失败。那好，下面我们就来看看修改、添加和保存属性的程序:

```
//修改sitename的属性值
prop.setProperty("sitename", "Boxcode");
//添加一个新的属性studio
prop.setProperty("studio", "Boxcode Studio");
//文件输出流
```

```
FileOutputStream fos = new FileOutputStream("prop.properties");
//将Properties集合保存到流中
prop.store(fos, "Copyright (c) Boxcode Studio");
fos.close();//关闭流
```

接下就是整个程序的源代码:

Java代码

```
1  import java.io.FileInputStream;
2  import java.io.FileOutputStream;
3  import java.util.Properties;
4
5  public class PropertyEditor {
6      public static void main(String[] args) throws Exception {
7          Properties prop = new Properties();// 属性集合对象

8          FileInputStream fis = new FileInputStream("prop.properties");// 属性文件输入流
9          prop.load(fis);// 将属性文件流装载到Properties对象中
10         fis.close();// 关闭流
11
12         // 获取属性值，sitename已在文件中定义
13         System.out.println("获取属性值:sitename=" + prop.getProperty("sitename"));
14         // 获取属性值，country未在文件中定义，将在此程序中返回一个默认值，但并不修改属性文件
```

```
15              System.out.println("获取属性值:
country=" + prop.getProperty("country", "中国"));
16
17              // 修改sitename的属性值
18              prop.setProperty("sitename", "Boxcode");
19              // 添加一个新的属性studio
20              prop.setProperty("studio", "Boxcode Studio");
21              // 文件输出流
22
        FileOutputStream fos = new FileOutputStream("prop.properties");
23              // 将Properties集合保存到流中
24              prop.store(fos, "Copyright (c) Boxcode Studio");
25              fos.close();// 关闭流
26          }
27      }
```

在我们知道如何读写一个属性文件之后，我们仍然还有很多需要注意的问题，因为load和store方法都是按照ISO-8859-1的编码方式读写属性流文件的，而lLatin1 的字符和某些特殊字符，而对于非Latin1 的字符和某些特殊字符，则要使用与字符和字符串字面值所用的类似转义序列，以值和元素的形式来表示它们。所以当我们在处理中文时，不可以在直接修改属性文件时，将中文的值赋予给属性，而是要在JAVA程序中通过setProperty方法给属性赋予中文的值，因为这样store会将中文转换成 unicode码，在读取时系统会将读取到的unicode码按系统的编码打印出来，对于中文系统，通常是GBK码，这样中文才能够正常显示。

到这里，就不难理解程序中的AppConfig.java了，下面是具体代码:

```
/**
 * 应用程序配置类：用于保存用户相关信息及设置
 *
 * @author 火蚁 (http://my.oschina.net/LittleDY)
 * @version 1.0
 * @created 2014-04-21
 */
public class AppConfig {
	private final static String APP_CONFIG = "config";

	public final static String CONF_APP_UNIQUEID = "APP_UNIQUEID";

	public final static String CONF_LOAD_IMAGE = "perf_loadimage";

	public final static String CONF_HTTPS_LOGIN = "perf_httpslogin";

	public final static String CONF_RECEIVENOTICE = "perf_receivenotice";

	public final static String CONF_VOICE = "perf_voice";

	public final static String CONF_CHECKUP = "perf_checkup";

	public final static String CONF_FRIST_START = "isFristStart";

	//默认存放图片的路径
	public final static String DEFAULT_SAVE_IMAGE_PATH = Environment
				.getExternalStorageDirectory()
				+ File.separator
				+ "OSChina"
				+ File.separator
				+ "git"
				+ File.separator;

	//默认存放文件的路径
	public final static String DEFAULT_SAVE_FILE_PATH = Environment
				.getExternalStorageDirectory()
				+ File.separator
				+ "OSChina"
				+ File.separator
				+ "git"
				+ File.separator
				+ "download"
				+ File.separator;

	private Context mContext;
	private static AppConfig appConfig;

	public static AppConfig getAppConfig(Context context) {
		if (appConfig == null) {
			appConfig = new AppConfig();
```

```
                appConfig.mContext = context;
        }
        return appConfig;
}

/**
 * 获取Preference设置
 */
public static SharedPreferences getSharedPreferences(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context);
}

/**
 * 是否加载显示文章图片
 */
public static boolean isLoadImage(Context context) {
        return getSharedPreferences(context).getBoolean(CONF_LOAD_IMAGE, true);
}

/**
 * 获得用户的token
 *
 * @return
 */
public String getPrivateToken() {
        return get(Contanst.PROP_KEY_PRIVATE_TOKEN);
}

public String get(String key) {
        Properties props = get();
        return (props != null) ? props.getProperty(key) : null;
}

public Properties get() {
        FileInputStream fis = null;
        Properties props = new Properties();
        try {
                // 读取files目录下的config
                // fis = activity.openFileInput(APP_CONFIG);

                // 读取app_config目录下的config
                File dirConf = mContext.getDir(APP_CONFIG, Context.MODE_PRIVATE);
                fis = new FileInputStream(dirConf.getPath() + File.separator
                                + APP_CONFIG);

                props.load(fis);
        } catch (Exception e) {
        } finally {
                try {
                        fis.close();
```

```
                } catch (Exception e) {
                }
            }
            return props;
        }

        private void setProps(Properties p) {
            FileOutputStream fos = null;
            try {
                // 把config建在files目录下
                // fos = activity.openFileOutput(APP_CONFIG, Context.MODE_PRIVATE);

                // 把config建在(自定义)app_config的目录下
                File dirConf = mContext.getDir(APP_CONFIG, Context.MODE_PRIVATE);
                File conf = new File(dirConf, APP_CONFIG);
                fos = new FileOutputStream(conf);

                p.store(fos, null);
                fos.flush();
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                try {
                    fos.close();
                } catch (Exception e) {
                }
            }
        }

        public void set(Properties ps) {
            Properties props = get();
            props.putAll(ps);
            setProps(props);
        }

        public void set(String key, String value) {
            Properties props = get();
            props.setProperty(key, value);
            setProps(props);
        }

        public void remove(String... key) {
            Properties props = get();
            for (String k : key)
                props.remove(k);
            setProps(props);
        }
}
```