如果经常会发生变化的数据我们可以定义在配置文件上。 比如说：数据库的用户名与密码。

# Properties类与配置文件

### Properties配置文件说明

Properties类对应.properties文件。文件内容是键值对，键值对之间使用"="或空格隔开。开头是"#"的表示注释

Properties类在加载.properties文件时使用的iso8859-1的编码。所以这个文件中的中文要特殊处理：如果这个配置文件中有中文就必须要进行转义，使用native2ascii.exe命令操作:

**native2ascii d:/my.properties d:/my2.properties**

使用Properties类中的load(InputStream) 方法可以加载配置文件，使用其中的store(OutputStream) 方法可以保存配置到指定文件。

更多的信息可以看Properties类的API文档。

### 加载配置文件

**public** **static** **void** testLoadProperties() **throws** Exception {

Properties properties = **new** Properties();

InputStream in = **new** FileInputStream("E:/itcast/config.properties");

properties.load(in); // 加载

in.close();

System.*out*.println(properties);

}

### 写配置文件

**public** **static** **void** testStoreProperties() **throws** Exception {

// 准备配置信息

Properties properties = **new** Properties();

properties.setProperty("name", "李四");

properties.setProperty("age", "20");

// 准备

OutputStream out = **new** FileOutputStream("d:/my.properties");

String comments = "这是我的配置文件";

// 写出去

properties.store(out, comments);

out.close();

}

作业：使用properties读取配置文件，读取数据库的用户名、密码。并且打包成jar包。

### 使用Properties类

|  |
| --- |
| **public** **class** DBUtil {    **static** Properties *properties* = **new** Properties();    **static**{  **try** {  Class clazz = DBUtil.**class**;  InputStreamReader fileReader =  **new** InputStreamReader(clazz.getResourceAsStream("/db.properties"));  *properties*.load(fileReader);  } **catch** (IOException e) {  e.printStackTrace();  }  }  **public** **static** String getUserName(){  String userName =*properties*.getProperty("userName");  **return** userName;  }    **public** **static** String getPassword(){  **return** *properties*.getProperty("password");  }  **public** **static** **void** main(String[] args) {  System.*out*.println("用户名："+ *getUserName*());  System.*out*.println("密码: "+ *getPassword*());  }  } |

package cn.itcast.path;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.io.InputStream;

import java.util.Properties;

/\*

如果经常会发生变化的数据我们可以定义在配置文件上。 比如说：数据库的用户名与密码。

配置文件的路径应该如何写 呢？

绝对路径：一个文件的完整路径信息。一般绝对路径是包含有盘符 的。 绝对路径的缺陷： 因为绝对路径是有盘符开头的，有些系统是没有盘符的。

相对路径: 相对路径是相对于当前程序的路径。当前路径就是执行java命令的时候，控制台所在的路径。

类文件路径 :类文件路径就是使用了classpath的路径找对应的资源文件。

如果需要使用到类文件路径首先先要获取到一个Class对象。

\*/

public class DBUtil {

static Properties properties ;

static{

try {

properties = new Properties();

//去加载配置文件 /

Class clazz = DBUtil.class;

InputStream inputStream = clazz.getResourceAsStream("/db.properties"); // "/"代表了Classpath的路径。 getResourceAsStream 该方法里面使用的路径就是使用了类文件路径。

properties.load(inputStream);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static void main(String[] args) {

System.out.println("当前路径："+ new File(".").getAbsolutePath() );

System.out.println("用户名："+ properties.getProperty("userName")+" 密码："+properties.getProperty("password"));

}

}