Java 读取资源、配置文件

作者：kevin2020
原文链接：https://ld246.com/article/1513782913405
来源网站：链滴

Class与ClassLoader

写java代码时常常需要加载一些外部的资源，通常我们会使用全路径名加载一份资源，比如：C:\Users XXX\Desktop\abc.jpg

但是，有些时候我们需要加载的是源代码路径下的资源或者配置文件等等，更习惯于使用相对路径，者直接给一个文件名，就希望能够找到我们需要的配置文件。

如何做到?

常见的方法是使用了 class.getResource 或 classloader.getResource

并且使用绝对路径是一个很差的方式，它会导致你的程序移植性很差,尽量使用相对路径

这两个方法看起来很相似，他们直接有什么区别?

查看源代码:

Class.getResourceAsStream(String name)

```
public InputStream getResourceAsStream(String name) {
    name = resolveName(name);
    ClassLoader cl = getClassLoader0();
    if (cl==null) {
        // A system class.
        return ClassLoader.getSystemResourceAsStream(name);
    }
    return cl.getResourceAsStream(name);
}
```

上面的代码可以看出， Class.getResourceAsStream(String name) 最终还是调用了 classloader.getesourceAsStream(String name) 。

但是两者还是有一些区别的，注意 name =resolveName(name)这一行， Class.getResourceAsStream(String name)在这里做了一些处理:

Class.resolveName(String name)

```
private String resolveName(String name) {
    if (name == null) {
        return name;
    }
    if (!name.startsWith("/")) {
        Class c = this;
        while (c.isArray()) {
```

```
            c = c.getComponentType();
        }
        String baseName = c.getName();
        int index = baseName.lastIndexOf('.');
        if (index != -1) {
            name = baseName.substring(0, index).replace('.', '/')
                +"/"+name;
        }
    } else {
        name = name.substring(1);
    }
    return name;
}
```

现在看这段代码还有点云里雾绕，不妨写几行代码测试一下，看看这段代码到底在干嘛：

```
public class App {
    public static void main(String[] args) {
        System.out.println("App.class.getClassLoader().getResource(\"\") : " + App.class.getClass
oader().getResource(""));
        System.out.println("App.class.getClassLoader().getResource(\"/\") : " + App.class.getClass
oader().getResource("/"));
        System.out.println("App.class.getResource(\"\") : " + App.class.getResource(""));
        System.out.println("App.class.getResource(\"/\") : " + App.class.getResource("/"));
    }
}
```

输出

```
App.class.getClassLoader().getResource("") : file:/D:/workspace/eclipse/cluster/TestClassloade
/bin/
App.class.getClassLoader().getResource("/") : null
```

App.class.getResource("") : file:/D:/workspace/eclipse/cluster/TestClassloader/bin/space/yukai

App.class.getResource("/") : file:/D:/workspace/eclipse/cluster/TestClassloader/bin/

虽然上面的代码使用了getResource，但与getResourceAsStream大同小异。

可以看到，

calssloder.getResource("")方法返回了classpath根路径（eclipse工程中，编译生成的类文件存放在/in目录下）；

calssloder.getResource("/")方法返回null，说明calssloder.getResource不支持以”/“开头的参数;

class.getResource("")方法返回了App.class所在的路径;

class.getResource("/")与calssloder.getResource("")表现一致，返回了classpath的根路径

再回顾上面的代码，是否有一点明白了呢?

读取jar包所在的位置

有时候需要知道jar包所在的位置，比如我们的项目需要一个默认的日志文件输出路径，这个路径就可是运行时jar包所在的目录。如何获取jar包所在的目录?

URL url = App.class.getProtectionDomain().getCodeSource().getLocation();

String path = url.toURL().getPath();

注意，上面的方法仅适用jdk1.5及以上

附实际使用方式

```
1.  package com.zkn.newlearn.others;
2.
3.  import java.io.IOException;
4.  import java.io.InputStream;
5.  import java.util.Properties;
6.
7.  import com.zkn.newlearn.gof.singleton.SimpleFactoryTest01;
8.
9.  /**
10.   * 读取资源文件的五种方式
11.   * @author zkn
12.   */
13.
14.  public class ClassReadResourceDemo {
15.
16.      public static void main(String[] args) {
17.          /**
18.           * 第一种方式 用类加载器读取资源文件。
19.           * 适用情形：资源文件和类文件在不在同一目录都可以。
20.           * 注意：getResourceAsStream里的参数要
```

```
         * 写资源文件的全限定路径，包名+文件名
         * 开头千万不要写"/"
         */
        InputStream is = ClassReadResourceDemo.class.
                getClassLoader().getResourceAsStream("com/zkn/newlearn/io/config.propertie
");
        Properties prop = new Properties();
        try {
            prop.load(is);
            System.out.println(prop.getProperty("key"));
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        /**
         * 第二种写法：用class.getResourceAsStream()(其实还是用的类加载器)
         * 适用情形：如果资源文件和类文件在同一包下，直接写资源文件的名称就行了，
         * 注意：资源文件的名称前面不需要加 “/”
         */
        is = ClassReadResourceDemo.class.getResourceAsStream("config.properties");
        try {
            prop.load(is);
            System.out.println(prop.getProperty("key"));
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        /**
         * 第三种写法：用class.getResourceAsStream()(其实还是用的类加载器)
         * 适用情形：这个写法适用的情形是资源文件和类文件不在同一个目录下的情况
         * 注意：开头一定要加上” / “
         */
        is = ClassReadResourceDemo.class.getResourceAsStream("/com/zkn/newlearn/io/c
nfig.properties");
        try {
            prop.load(is);
            System.out.println(prop.getProperty("key"));
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        /**
         * 第四种写法:用class.getResourceAsStream()
         * 适用情形：这种写法适用于资源文件在根目录下的情况
         * 注意：文件名称前面一定要加上” / “
         */
        is = ClassReadResourceDemo.class.getResourceAsStream("/config.properties");
        try {
            prop.load(is);
            System.out.println(prop.getProperty("key"));
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
73.         /**
74.          * 第五种写法：用类加载器来读取资源文件
75.          * 适用情形：资源文件在跟目录下
76.          * 注意：资源文件名称前面一定不要加”/“
77.          */
78.         is = ClassReadResourceDemo.class.getClassLoader().getResourceAsStream("config.p
operties");
79.         try {
80.             prop.load(is);
81.             System.out.println(prop.getProperty("key"));
82.             is.close();
83.         } catch (IOException e) {
84.             e.printStackTrace();
85.         }
86.     }
87. }
```