Java中classpath讲解及使用方式

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/79aaf9bc830da3816039657c43e28253.html

笔者之前对于classpath都没有什么深入的研究，之前的项目打包jar包都是按照网上的教程自己照着葫芦画瓢。但是因为最近碰到一些关于这方面的知识，因此索性觉得要好好补充一下这方面的知识。下面的文章主要是翻译自oralce官网关于设置classpath的说明setting the class path，并对其中一些地方进行了部分的补充说明，方便读者理解。

概要

classpath是java运行时环境搜索类和其他资源文件（比如jar\zip等资源）的路径。可以通过jdk工具（比如javac命令、java命令）后面的-classpath 参数设置classpath（这个可以通过装有java环境的dos窗口下输入sdktool的命令来查看可以输入的参数选项，见下图）

或通过设置classpath环境变量。该 -classpath选项是首选项，因为您可以为每个应用程序单独设置，而不会影响其他应用程序，而无需其他应用程序修改其值。

下面是这两种方式的说明示例：

（1）通过jdk的命令行来为每个application设置

这个设置主要是为每个application设置，并不会影响别的程序的classpath以及环境变量的classpath，他只在当前窗口生效。

c:> sdktool -classpath classpath1 ;classpath2 …

（2）设置classpath环境变量，这个是全局设置，对所有的应用程序生效

c:> set classpath=classpath1 ;classpath2 …

sdktool

一个命令行工具，例如java， javac，javadoc，或apt。有关列表，请参阅jdk工具。

classpath1 ;classpath2

.jar，.zip或.class文件的classpath。每个classpath应以文件名或目录结尾，针对不同的情况，设置classpath的格式方法如下：

（1）对于包含.class文件的.jar或.zip文件，classpath以.zip或.jar文件的名称结尾。

下面是示例，笔者自己将一个工程打包成jar包，同时指定这个jar包下的某个class文件运行。结果如下：

下面是工程的目录，其中工程中引用了其他jar包，笔者在打包的时候，用maven的assembly插件打包成了一个fat jar（想要了解具体打包方法，查看这两篇博文：

【java】打包jar包并用脚本执行

【maven】maven系列（二）——利用assembly插件打包

查看下面的classpath的值是怎么指定的。

（2）对于未命名包中的.class文件，classpath以包含.class文件的目录结束。比如一个a.class文件在d:path1\path2下，但是这个a.class没有package，那么如果想让jvm搜索到这个class文件，这时指定classpath的时候，可以使用d:path1\path2，因为classpath的默认路径是当前路径，因此在使用的时候，如果是在当前路径下打开dos窗口，classpath安史之乱的故事命令缺省，那么这个时候其效果与指定当前路径为classpath的效果等效。如下的示例：

test.java

```
public class test {    public static void main(string[] args) {       syste
                                    m.out.println("hello world!");    }}
```

这个类是没有package的。

下面是运行结果：

程序运行成功

（3）对于命名包中的.class文件，类路径以包含“ root ”包（完整包名称中的第一个包）的目录结束。

示例如下：

equals.java在名为string的package下

```
package string;public class equals {    public static void main(string[] args){
  string a=new string("test");       string b=new string("test");
                                    system.out.pr煮元宵intln(a==b);    }}
```

那么对于这种情况，指定classpath的方式如下：

通过上面几条命令，可以看出来正确的命令行格式如下：

java -classpath package_rootpath package_name.class_name

多个路径条目以分号分隔。使用set命令，等号（=）周围的空格可以忽略。

默认classpath是当前目录。设置 classpath变量或使用 -classpath命令行选项将覆盖该默认值，因此，如果要在搜索路径中包含当前目录，则必须在新设置中包含“ . ”（关于路径的说明可以参考笔者的另一篇博文）。

既不是目录也不是归档（.zip或.jar文件）和*的classpath会被忽略。

描述

类路径告诉jdk工具和应用程序在哪里可以找到第三方和用户定义的类 – 即不是java 扩展或java平台的一部分的类。类路径需要找到你用javac编译器编译的任何类——其默认值是当前目录，以方便地找到这些类。

jdk，jvm和其他jdk工具通过按顺序搜索java平台（bootstrap）类，任何扩展类和类路径来查找类。（有关搜索策略的详细信息，请参阅如何找到类。）大多数应用程序的类库都希望利用扩展机制。您只需要设置classpath，当您要加载的class满足下面的条件（a）不在当前目录或其任何子目录中的类时，（b）不在扩展机制指定的位置。

如果要从旧版本的jdk升级，启动设置可能包含classpath不再需要的设置。您应该删除任何不是特定于应用程序的设置，例如classes.zip。使用java虚拟机的某些第三方应用程序可能会修改您的classpath环境变量以包含它们使用的库。这样的双排旱冰鞋教程设置可以保持不变。

当您调用jvm或其他jdk工具或使用环境变量时，可以使用jdk工具’-classpath选项来更改classpath。使用该选项优于设置环境变量，因为您可以为每个应用程序单独设置，而不会影响其他应用程序，而无需其他应用程序修改其值。

类可以存储在目录（文件夹）或归档文件（这里指的归档文件也就是那些jar包和zip包）中。java平台类存储在 rt.jar。有关归档的更多详细信息以及有关类路径的工作原理的信息，请参阅本文档末尾附近的类路径和包名称。

重要说明：某些旧版本的jdk sofware 在默认的类路径中包含一个/classes条目。该目录存在供jdk软件使用，不应用于应用程序类。应用程序类应放在jdk等级目录之外。这样，安装新的jdk并不会强制您重新安装应用程序类。为了与旧版本兼容，使用/classes目录作为类库的应用程序将以当前版本运行，但不保证将在以后的版本中运行。

使用jdk工具的-classpath选项

jdk工具java，jdb，javac和javah有一个-classpath选项，用于替换由classpath环境变量指定的路径。这是改变classpath设置的推荐选项，因为每个应用程序都可以具有所需的classpath，而不会干扰任

何其他应用程序。

运行时工具java也有一个 -cp选项。此选项是-classpath的缩写。

对于非常特殊的情况，java和 javac都有选项，可以让您更改用于查找自己的类库的路径。然而，绝大多数用户将永远不需要使用这些选项。

使用classpath环境变量

一般来说，您使用-classpath 命令行选项要像上一节介绍的那样。本节介绍如何设置classpath环境变量，或者清除以前安装中留下的设置。

设置classpath

在classpath环境变量修饰的组命令。格式为：

set classpath = path1 ; path2…

路径应以指定驱动器的字母开头，例如c:\。这样，如果您碰巧切换到其他驱动器，则仍然会找到class文件。（如果设置的路径是d:\，那么jvm就会去d盘下找这个class，而不是在c盘下）

清除classpath

如果您的classpath环境变量设置为不正确的值，或者您的启动文件或脚本设置不正确的路径，可以classpath 使用以下命令取消设置：

c：> set classpath =

此命令仅在当前命令提示符窗口中取消classpath。您还应删除或修改启动设置，以确保classpath在将来的会话中拥有正确的设置。

更改启动设置

如果classpath变量在系统启动时设置，则要查找的位置取决于您的操作系统：

对于windows 95和98，检查autoexec.bat的 set命令。

其他（windows nt，windows 2000，…） 的系统，classpath
环境变量可以使用控制面板中的系统实用程序来设定。

了解类路径通配符

类路径条目可包含基名通配符*，这被认为等同于指定在扩展目录下的所有文件的列表.jar或.jar。例如，类路径条目 foo/* 指的就是foo目录下的所有jar文件。一个简单由*组成的类路径指的

就是当前目录中所有jar文件的列表。

通配符*的classpath写法对于class文件来说不适用（意思就是通配符*适用的对象是jar而不是class）。要在单个目录中匹配类和jar文件 foo，请使用foo;foo/*或
foo/*;foo。前者的顺序决定了foo文件目录下的类和资源加载在前，jar加载在后，后者反之亦然。

子目录不是递归搜索。例如， foo/*仅在查找jar文件foo，而不包括foo/bar，foo/baz等子目录。

扩展类路径中枚举目录中的jar文件的顺序未指定（the order in which the jar files in a directory are enumerated in the expanded class path is not specified，这一句没看懂，读者如果有了解的，可以留言回复一下，谢谢！），并且可能因平台而异，甚至在同一台机器上也会随时变化。构建良好的应用程序不应取决于任何特定的顺序。如果需要特定的顺序，则jar文件可以在类路径中显式枚举。

通配符的扩展这一过程是在调用程序的main方法之前，在类加载过程期间。每个包含通配符的输入类路径将被枚举为这个目录下的所有jar文件路径。例如，如果目录foo包含
a.jar，b.jar和c.jar，然后将类路径foo/*扩展为
foo/a.jar;foo/b.jar;foo/c.jar，并且该字符串将是系统属性的值 java.class.path。

classpath环境变量和-classpath（或 -cp）的命令行选项没什么不同的。也就是说，通配符在所有情况下都是一样适用的。但是，类路径通配符在class-path jar-
manifest头文件中并不符合要求（这一句话也不是很理解，希望读者指教！）。

了解类路径和包名称

java类被组织成被映射到文件系统中的目录的包。但是，与文件系统不同，无论何时指定包名称，都可以指定整个 包名称，而不是它的一部分。例如，对于包名称 java.awt.button是总是指定为java.awt。

例如，假设您希望java运行时可以找到cool.class该类在utility.myapp包下。如果该目录的路径是c:\java\myclasses\utility\myapp，那么设置的classpath就应该包含c:\java\myclasses。

要运行该应用程序，可以使用以下jvm命令：

c：> java -classpath c：\ java \ myclasses utility.myapp.cool

当应用程序运行时，jvm使用这个classpath设置来查找utility.myapp包中定义的任何其他类（也包括cool类）。

请注意，命令中指定了整个包名称。如果是设置classpath为c:\java\myclasses\utility并使用命令 java myapp.cool，这种做法是不可能成功的。jvm找不到这个类。

（你可能想知道什么定义了一个类的包名称，答案是包名是类的一部分，不能修改，除了重新编译该类）。

注意：包指定机制的一个有趣的后果是，作为同一包的一部分的文件实际上可能存在于不同的目

录中。每个类的包名称将相同，但是每个文件的路径可以从类路径中的不同目录开始。

文件夹和归档文件

当类存储在目录（文件夹）中时，比如存储在c:\java\myclasses\utility\myapp，那么classpath条目指向包含包名称的第一个元素的目录。也就是package的上一级目录（在这种情况下，classpath就是c:\java\mycl2020录取通知书查询入口asses，因为包名是utility.myapp。）

但是当类存储在归档文件（.zip或.jar文件）中时，类路径条目是.zip或.jar文件的路径并包含.zip或.jar文件。例如，要使用.jar文件中的类库，命令将如下所示：

c：> java -classpath c：\ java \ myclasses \ myclasses.jar utility.myapp.cool

多重指定

要查找c:\java\myclasses目录下的类以及c:\java\otherclasses目录下的类，那么请将类路径设置为：

c：> java -classpath c：\ java \ myclasses; c：\ java \ otherclasses …

请注意，这两个路径以分号分隔。

指定顺序

多个classpath的顺序很重要。ja南辕北辙告诉我们什么道理va解释器将按照它们在classpath变量中出现的顺序来查找目录中的类。在上面的例子中，java解释器将首先在c:\java\myclasses目录下查找类。只有在该目录中找不到具有正确名称的类时，解释器才会在c:\java\otherclasses目录中查找。

以上就是本文的全部内容，希望对大家的学习有所帮助，也希望大家多多支持www.887551.com。