转载 JAVA类路径

2010-03-04 10:35:00 datouuupp 阅读数 9738 更多

Java 类路径

Java 类路径告诉 *java* 解释器和 *javac* 编译器去哪里找它们要执行或导入的类。类（您可能注意到的那些 **.class* 文件）可以存储在目录或 *jar* 文件中，或者存储在两者的组合中，但是只有在它们位于类路径中的某个地方时，*Java* 编译器或解释器才可以找到它们。

在 *Windows* 中，类路径中的多个项是用分号分隔（ *;*）的，而在 *UNIX* 中，这些项是用冒号分隔（ *:*）的。在以下实例中，类路径中包括两个 *Cloudscape jar* 文件（ *cs.jar* 和 *cstools.jar* ），以及一个存储 **.class* 文件的目录位置（ *myDevDir* ）：

Windows 类路径:

c:/Cloudscape_10.0/lib/cs.jar;c:/Cloudscape_10.0/lib/cstools.jar;c:/myPath/myDevDir

UNIX 类路径:

/Cloudscape_10.0/lib/cs.jar:/Cloudscape_10.0/lib/cstools.jar:/myPath/myDevDir

本文中的其余实例使用的都是 *Windows* 语法，因此，如果您是在 *UNIX* 机器上，那么需要对语法进行相应的调整。

设置 *Java* 类路径

有三种方式设置 *Java* 类路径：

1. 永久地，通过在系统级上设置 *CLASSPATH* 环境变量来实现。

使用控制面板的系统设置来添加名为 *CLASSPATH* 的新变量，从而永久性地设置 *Windows* 环境变量。

UNIX 用户可以通过向 *.profile* 或 *.cshrc* 文件添加 *CLASSPATH* 变量来永久设置类路径。

2. 临时地，通过在命令窗口或 *shell* 中设置 *CLASSPATH* 环境变量来实现。

在 *Windows* 命令窗口中临时设置 *CLASSPATH*

C:/>set CLASSPATH=%CLOUDSCAPE_INSTALL%/lib/cs.jar;.;

如果是临时设置类路径，那么每次打开新的命令窗口时，都需要再次设置它。

3. 在运行时进行，每次启动 *Java* 应用程序和 *JVM*，都要指定类路径。

运行时使用 *-cp* 选项来指定类路径，这里的运行时是指启动应用程序和 *JVM* 时。

例如

C:/Cloudscape_10.0/demo/programs/simple>java -cp %CLOUDSCAPE_INSTALL%/lib/cs.jar; SimpleApp

检测问题

常见类路径错误

主要有两种类型的类路径问题。第一类问题发生在没有从类路径中找到您试图使用的 *Java* 类时，此时，它抛出一个 *java.lang.ClassNotFoundException* 异常。第二类问题发生在找到了您正试图使用的类，但没有找到它所导入的某个类时。本例中，在编译时显示了所导入的类，但在运行时，所导入的类没有包含在类路径中。这将抛出一个 *java.lang.NoClassDefFoundError* 异常。还有另一种考虑*NoClassDefFoundError* 的方式，也就是说，在编译当前执行的类时，所搜索的类定义是存在的，但在运行时却再也无法找到该定义了

如何可以解决这类问题呢？首先，检查类路径，验证库是否真正位于您认为的地方。例如，在 *Windows* 中使用该命令来输出类路径：

C:/my_dir>echo %CLASSPATH%

c:/Cloudscape_10.0/lib/cs.jar;c:/Cloudscape_10.0/lib/cstools.jar

然后，在 *CLASSPATH* 变量中查看每个路径，并用 *dir*（*Windows*）或 *ls*（*UNIX*）命令查看这些文件是否存在。

如果不知道类位于哪个 *jar* 文件中，可以用以下命令来检查：

jar -tvf cs.jar | more

该命令产生许多输出。如果正使用 *Linux* 或 *Unix*，或者在 *Windows* 中使用 *UNIX* 使用程序，那么您可以用 *grep* 筛选您所查找的类。例如，以下命令将查找 *com.ihost.cs.tools.sysinfo* 类：

C:/Cloudscape_10.0/lib>jar -tvf cs.jar | grep -i com.ihost.cs.tools.sysin

分析idea运行一个springBoot项目时的命令

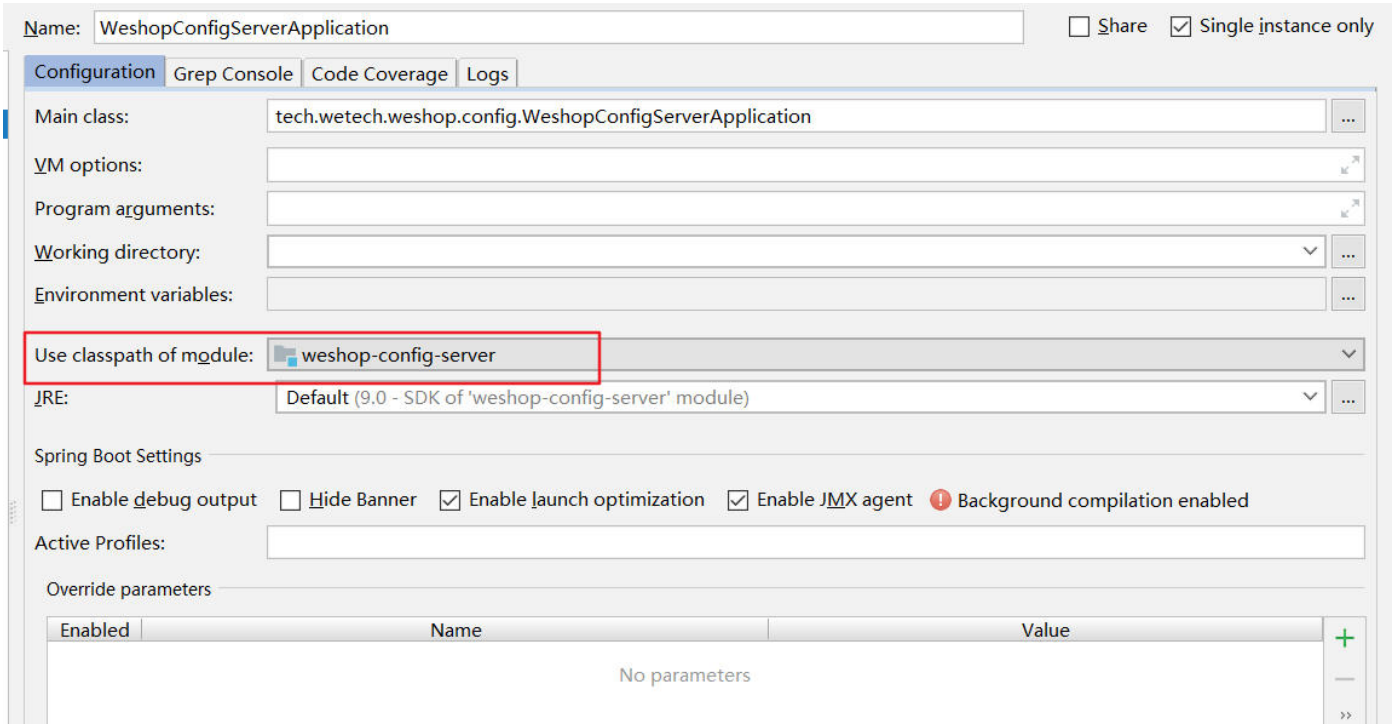
```
# 有些地方没有换行，，后面还有很多

#java命令（带位置）
"C:\Program Files\Java\jdk-9.0.4\bin\java"
#其他参数
 -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:53234,suspend=y,server=n -XX:TieredStopAtLevel=1 -noverify
#-D环境变量
 -Dspring.output.ansi.enabled=always -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=53233 -Dco
#其他参数
 -javaagent:C:\Users\TJR_S\.IntelliJIdea2017.3\system\captureAgent\debugger-agent.jar=C:\Users\TJR_S\AppData\Local
#-D环境变量
 -Dfile.encoding=UTF-8
#运行类时设置类路径，不需要加默认类路径'.'
-classpath "
    #项目本身路径
D:\project\weshop\weshop-config-server\target\classes;
    #所有maven引入依赖类路径
E:\MavenRepository\org\springframework\cloud\spring-cloud-config-server\2.1.4.RELEASE\spring-cloud-config-server-2
    #不知名类路径路径
C:\Program Files\JetBrains\IntelliJ IDEA 2017.3.5\lib\idea_rt.jar
"
#运行的类
tech.wetech.weshop.config.WeshopConfigServerApplication
```

WEB-INF/classes,lib才是classpath，我指的是web项目
你说的classpath是资源入口，外部不可访问；
另外你说的环境变量
CLASSPATH是什么？它的作用是什么？

它是javac编译器的一个环境变量。它的作用与import、package关键字有关。当你写下improt java.util.*时，编译器面对import关键字时，就知道你要引入java.util这个package中的类；但是编译器如何知道你把这个package放在哪里了呢？所以你首先得告诉编译器这个package的所在位置；如何告诉它呢？就是设置CLASSPATH啦 :) 如果java.util这个package在c:/jdk/ 目录下，你得把c:/jdk/这个路径设置到CLASSPATH中去！当编译器面对import java.util.*这个语句时，它先会查找CLASSPATH所指定的目录，并检视子目录java/util是否存在，然后找出名称吻合的已编译文件（.class文件）。如果没有找到就会报错！CLASSPATH有点像c/c++编译器中的INCLUDE路径的设置哦，是不是？当c/c++编译器遇到include 这样的语句，它是如何运作的？哦，其实道理都差不多！搜索INCLUDE路径，检视文件！当你自己开发一个package时，然后想要用这个package中的类；自然，你也得把这个package所在的目录设置到CLASSPATH中去！CLASSPATH的设定，对JAVA的初学者而言是一件棘手的事。所以Sun让JAVA2的JDK更聪明一些。你会发现，在你安装之后，即使完全没有设定CLASSPATH，你仍然能够编译基本的JAVA程序，并且加以执行

idea会自己设置类路径，这里可以设置



设置java类路径

2018-06-22 18:15:54 qq_37284484 阅读数 2931 更多

都知道java程序的启动方式为：

java -cp 类路径 全限定的类名 参数1 参数2 参数3

在上面的调用中，初学者可能会在两个地方掉进坑中：

1.java命令：在windows上，它是没有显示地写上exe后缀的可执行程序。大家都知道在计算机中，要指明一个文件，仅文件名是不够的，而是需要完整的路径才能唯一的定位它。但此处为什么可以只写一个程序名字？只是因为java安装目录下面的bin目录被加到了系统的path环境变量中，而这个添加操作通常是在安装JRE时自动完成的。如果bin目录没有被添加到path环境变量中（这种情况可能出现在随便的手动拷贝jre目录的时候），则此处就要写完整了，例如：C:\Program Files (x86)\Java\jre1.8.0_121\bin\java -cp ……

2.-cp选项：用于罗列类路径的位置，它是-classpath的缩写。

类路径的作用是告诉java虚拟机从哪些地方查找类。

下面重点讲一下类路径的要点：

1.类路径中可以指定三种位置：文件夹、jar文件、zip文件；

2.-cp或者-classpath可以指定多个位置，在windows上是用分号隔开，在linux上是用冒号隔开。例如在linux上：-cp dir1:dir2:dir3，此处指定了3个目录作为类查找路径。

3.如果没有明确的指定类路径，则默认是当前工作路径，注意当前工作路径是一个文件夹，因此如果当前工作路径下面有个jar文件，jar文件中的类是不会被找到的。记住文件夹与jar各是各。

4.如果通过-cp或者-classpath选项指定了类路径，则当前工作路径就不会再包含进类路径中了。此时如果仍然需要将当前工作路径纳入类路径，需要通过点号再加回来。例如：-cp .:dir1，此处表示在linux上当前工作目录和dir1目录都是类路径；

5.指定jar文件的示例：-cp main.jar。指定zip文件也可以，此处不再演示。

6.关于通配符的使用：

a.通配符只是用来匹配jar的，不能用来匹配类文件或者目录。

b.通配符不递归匹配；

c.如果目录dir下面有6个jar都要作为类路径，传统的可以写成：-cp test1.jar:test2.jar:test3.jar:test4.jar:test5.jar:test6.jar，有没有发现很麻烦，其实用通配符写起来简单多了：-cp *，此时如果当前目录也要作为类路径，可写成：-cp .:*，冒号是linux上的路径分隔符；

d.有没有发现，通配符只是减少了在指定类路径时罗列jar麻烦。

7.还可以通过CLASSPATH环境变量指定类路径，但是不到万不得已，不要这样用。显然的，如果配置到环境变量中去了，则系统中所有的java程序都会相互影响，那就不妙了。