**java 扫描指定包下类(包括jar中的java类)**

原创 2017年04月01日 21:09:50

* 标签：
* [java](http://so.csdn.net/so/search/s.do?q=java&t=blog) /
* [jar](http://so.csdn.net/so/search/s.do?q=jar&t=blog) /
* [class](http://so.csdn.net/so/search/s.do?q=class&t=blog) /
* [结构](http://so.csdn.net/so/search/s.do?q=%E7%BB%93%E6%9E%84&t=blog)
* 2079

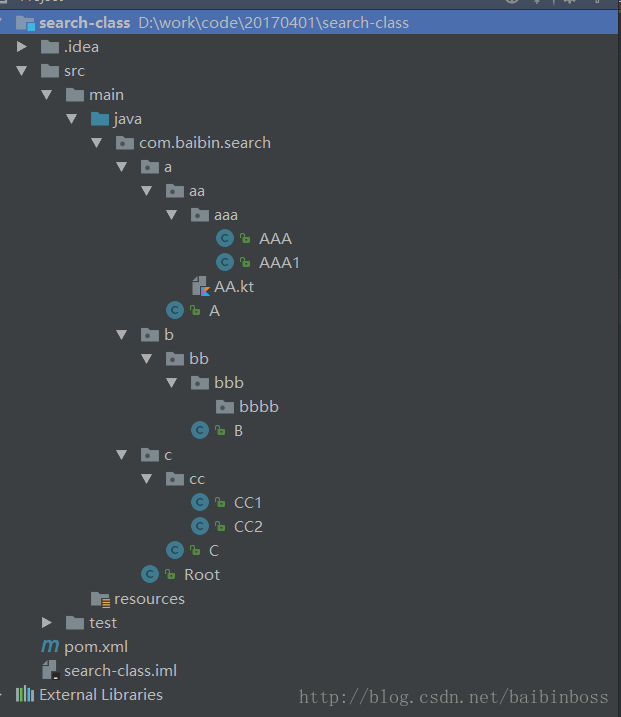
在很多的实际场景中，我们需要得到某个包名下面所有的类，包括我们自己在src里写的java类和一些第三方提供的jar包里的类，那么怎么来实现呢？今天带大家来完成这件事。

* 1
* 2

src下面的类如何获取：

首先，比较简单的是得到我们自己写的类，我们先来完成这个，项目的结构图如下：

* 1
* 2



我故意创建了这么个比较复杂的项目结构，现在我们就来获取com.baibin包下所有的类，并且打印他们，代码如下：

* 1
* 2

import org.junit.Test;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

public class Main {

List<String> classPaths = new ArrayList<String>();

@Test

public void searchClass() throws ClassNotFoundException {

//包名

String basePack = "com.baibin";

//先把包名转换为路径,首先得到项目的classpath

String classpath = Main.class.getResource("/").getPath();

//然后把我们的包名basPach转换为路径名

basePack = basePack.replace(".", File.separator);

//然后把classpath和basePack合并

String searchPath = classpath + basePack;

doPath(new File(searchPath));

//这个时候我们已经得到了指定包下所有的类的绝对路径了。我们现在利用这些绝对路径和java的反射机制得到他们的类对象

for (String s : classPaths) {

//把 D:\work\code\20170401\search-class\target\classes\com\baibin\search\a\A.class 这样的绝对路径转换为全类名com.baibin.search.a.A

s = s.replace(classpath.replace("/","\\").replaceFirst("\\\\",""),"").replace("\\",".").replace(".class","");

Class cls = Class.forName(s);

System.out.println(cls);

}

}

/\*\*

\* 该方法会得到所有的类，将类的绝对路径写入到classPaths中

\* @param file

\*/

private void doPath(File file) {

if (file.isDirectory()) {//文件夹

//文件夹我们就递归

File[] files = file.listFiles();

for (File f1 : files) {

doPath(f1);

}

} else {//标准文件

//标准文件我们就判断是否是class文件

if (file.getName().endsWith(".class")) {

//如果是class文件我们就放入我们的集合中。

classPaths.add(file.getPath());

}

}

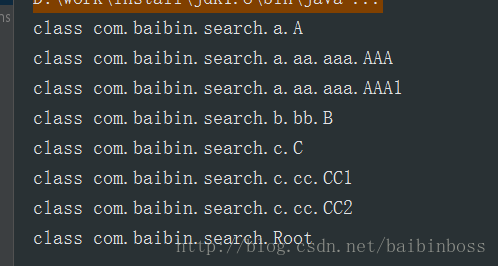
}

}

* 1

效果如下：

* 1
* 2



总结：这样的src下面的都比较容易处理，也很容易想到，但是jar包下面的就没这么简单了，但是还是有办法的。

* 1
* 2

jar中的类如何获取：

jar下的类我们可以通过JarURLConnection类来或者，代码如下：

* 1
* 2

import org.junit.Test;

import java.io.IOException;

import java.net.JarURLConnection;

import java.net.URL;

import java.util.Enumeration;

import java.util.jar.JarEntry;

import java.util.jar.JarFile;

public class JarMain {

@Test

public void searchClass() throws IOException, ClassNotFoundException {

String basePack = "org.junit";

//通过当前线程得到类加载器从而得到URL的枚举

Enumeration<URL> urlEnumeration = Thread.currentThread().getContextClassLoader().getResources(basePack.replace(".", "/"));

while (urlEnumeration.hasMoreElements()) {

URL url = urlEnumeration.nextElement();//得到的结果大概是：jar:file:/C:/Users/ibm/.m2/repository/junit/junit/4.12/junit-4.12.jar!/org/junit

String protocol = url.getProtocol();//大概是jar

if ("jar".equalsIgnoreCase(protocol)) {

//转换为JarURLConnection

JarURLConnection connection = (JarURLConnection) url.openConnection();

if (connection != null) {

JarFile jarFile = connection.getJarFile();

if (jarFile != null) {

//得到该jar文件下面的类实体

Enumeration<JarEntry> jarEntryEnumeration = jarFile.entries();

while (jarEntryEnumeration.hasMoreElements()) {

/\*entry的结果大概是这样：

org/

org/junit/

org/junit/rules/

org/junit/runners/\*/

JarEntry entry = jarEntryEnumeration.nextElement();

String jarEntryName = entry.getName();

//这里我们需要过滤不是class文件和不在basePack包名下的类

if (jarEntryName.contains(".class") && jarEntryName.replaceAll("/",".").startsWith(basePack)) {

String className = jarEntryName.substring(0, jarEntryName.lastIndexOf(".")).replace("/", ".");

Class cls = Class.forName(className);

System.out.println(cls);

}

}

}

}

}

}

}

}

* 1

通过这两种方式我们就可以得到指定包名下面所有的类了，这个还是挺有用的，比如spring中经常用来扫描指定包注解的实现等。

* 1
* 2

版权声明：本文为博主原创文章，未经博主允许不得转载。