链滴

java实现在指定目录中查找字符串

作者：xh

原文链接：https://ld246.com/article/1368190290825

来源网站：链滴

 用法

<pre>Usage: java JSearch -t str [-options]
options:
    -d directory   要查找的子目录，默认是当前目录
    -f file   要查找的文件，默认所有txt和sql文件，支持正则
    -t targetStr   将要查找的目标字符串，不可空，支持正则
    -c charset   字符编码
    -r   递归查找子目录
示例:
java JSearch -t t00ls
java JSearch -t t00ls -d /root/test/txt -f .*\.txt -c gbk -r
</pre>

 代码

<pre>package pw.tly.utils;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileFilter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class JSearch {
private static String path = "";
private static String filename = ".
(txt|sql)";
private static String targetStr = "";
private static String charset = "";
private static boolean isSearchSubdirectory = false;
private static String usage = "Usage: java JSearch -t str [-options]
noptions:\n\t -d directory\t 要查找的子目录，默认是当前目录\n\t -f file\t 要查找的文件，默认所有tx
和sql文件，支持正则\n\t -t targetStr\t 将要查找的目标字符串，不可空，支持正则\n\t -c charset\t
符编码\n\t -r\t\t 递归查找子目录\n示例：\njava JSearch -t t00ls\njava JSearch -t t00ls -d /root/test/t
t -f .\.txt -c gbk -r";

```
public static void main(String[] args) {
    if(!parseArgs(args)){
        return;
    }
    List&lt;File&gt; list = getFiles(path, filename, isSearchSubdirectory);
    find(list);
}

/**
 * 在文件中查找字符串并输出
 * @param list 文件列表
 */
private static void find(List&lt;File&gt; list) {
    for (File file : list) {
        String result = find(file.getAbsolutePath());
        if(&quot;&quot;.equals(result)){
            continue;
        }
        System.out.println(&quot;**********在&quot; + file.getAbsolutePath() + &quot;中找到：*
********&quot;);
        System.out.println(result);
    }
}

/**
 * 在文件中查找字符串
 * @param filename 文件路径
 * @return 查找结果
 */
private static String find(String filename) {
    if(targetStr == null || targetStr.length() == 0) {
        throw new RuntimeException(&quot;目标字符串不能为空&quot;);
    }
    BufferedReader br = null;
    InputStreamReader isr = null;
    FileInputStream fis = null;
    String line = null;
    String result = &quot;&quot;;
    try {
        fis = new FileInputStream(filename);
        isr = &quot;&quot;.equals(charset) ? new InputStreamReader(fis) : new InputStreamRead
r(fis, charset);
        br = new BufferedReader(isr);

        Pattern pattern = Pattern.compile(targetStr);

        while((line = br.readLine()) != null){
            Matcher matcher = pattern.matcher(line);
            if(matcher.find()){
                result += line + &quot;\n&quot;;
            }
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
```

```
    } catch (UnsupportedEncodingException e) {
        System.out.println(&quot;编码无效&quot;);
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(fis != null){
            try {
                fis.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if(isr != null){
            try {
                isr.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if(br != null){
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return result;
}

/**
 * 获取某一路径下的所有符合条件的文件
 * @param path 路径
 * @param filename 文件名过滤条件
 * @param isSearchSubdirectory 是否查找子目录
 * @return 文件列表
 */
private static List&lt;File&gt; getFiles(String path, final String filename, boolean isSearchSubdi
ectory) {
    List&lt;File&gt; result = new ArrayList&lt;File&gt;();
    File file = new File(path).getAbsoluteFile();
    if(!file.exists()){
        return result;
    }
    if(file.isFile()){
        result.add(file);
        return result;
    }

    File[] files = file.listFiles(new FileFilter() {

        @Override
        public boolean accept(File file) {
```

```
            if(!file.getAbsoluteFile().isDirectory()){
                if(&amp;quot;&amp;quot;.equals(filename)){
                    return true;
                }
                if(file.getAbsoluteFile().getName().matches(filename)){
                    return true;
                }
            }
            return false;
        }

    });
    result.addAll(Arrays.asList(files));

    if(isSearchSubdirectory){
        File[] directorys = file.listFiles(new FileFilter() {

            @Override
            public boolean accept(File file) {
                if(file.getAbsoluteFile().isDirectory()){
                    return true;
                }
                return false;
            }

        });

        for (File directory : directorys) {
            List&amp;lt;File&amp;gt; list = getFiles(directory.getAbsolutePath(), filename, isSearchSubdirect
ry);
            result.addAll(list);
        }
    }

    return result;
}

/**
 * 解析参数
 * @param args 参数数组
 * @return 解析是否成功
 */
private static boolean parseArgs(String[] args) {
    if(args.length == 0){
        System.out.println(usage);
        return false;
    }
    try {
        for (int i = 0; i &amp;lt; args.length; i++) {
            if(&amp;quot;-d&amp;quot;.equals(args[i])){
                path = args[i + 1];
            }
            if(&amp;quot;-f&amp;quot;.equals(args[i])){
                filename = args[i + 1];
```

```
            }
            if(&amp;quot;-t&amp;quot;.equals(args[i])){
                targetStr = args[i + 1];
            }
            if(&amp;quot;-c&amp;quot;.equals(args[i])){
                charset = args[i + 1];
            }
            if(&amp;quot;-r&amp;quot;.equals(args[i])){
                isSearchSubdirectory = true;
            }
        }
        if(&amp;quot;&amp;quot;.equals(targetStr)){
            System.out.println(&amp;quot;请用-t参数指定要查找的字符串&amp;quot;);
            return false;
        }
        return true;
    } catch (Exception e) {
        System.out.println(usage);
        return false;
    }
}

}
```
</pre>