# FindIt(帮我筛简历)实验报告

## Table of Contents

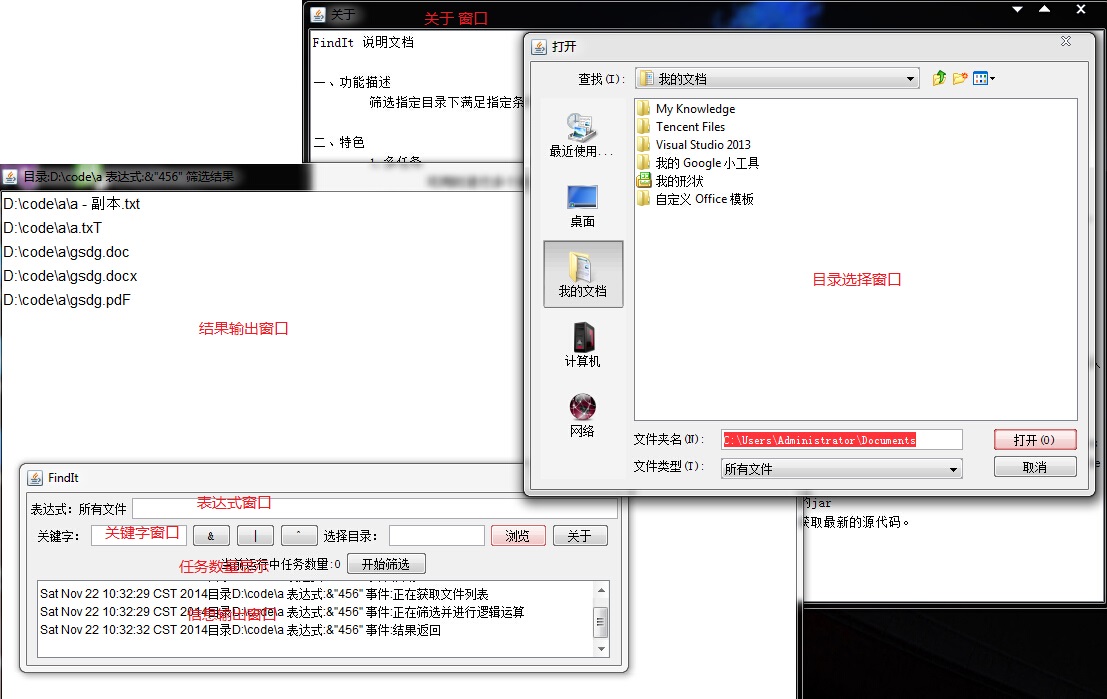
* [1 需求分析](file:///C:\Users\Administrator\Desktop\新建文件夹\report.html#sec-1)
* [2 整体设计](file:///C:\Users\Administrator\Desktop\新建文件夹\report.html#sec-2)
* [3 实现](file:///C:\Users\Administrator\Desktop\新建文件夹\report.html#sec-3)
* [4 实验测试](file:///C:\Users\Administrator\Desktop\新建文件夹\report.html#sec-4)
* [5 关于本程序](file:///C:\Users\Administrator\Desktop\新建文件夹\report.html#sec-5)
* [6 组员分工](file:///C:\Users\Administrator\Desktop\新建文件夹\report.html#sec-6)

## 1 需求分析

* 题目 人工筛取海量简历费时费力，用Java实现一个简历自动筛选软件，实现以下功能：
  + 需要有界面
  + 能够设置一个目录，软件对该目录下的所有简历进行筛选
  + 能够对word和pdf格式的简历进行筛选
  + 筛选时，从软件界面指定关键字，然后读取所有word文档和pdf文档的内容，查找其中是否存在该关键字，如果存在，则将文件路径名显示在界面上。筛选结果是一系列文件路径名组成的列表。（对word和pdf文档内容的读取，可以借助第三方的程序库实现）
  + 支持同时指定多个关键字（至少3个），并同时依据这3个关键字进行筛选。
  + 不仅支持英文关键字，还要支持中文关键字。
  + 可选功能：支持对关键字的排除，例如，能够通过“-西安电子科技大学”筛 选出所有不包含“西安电子科技大学”关键字的简历.

## 2 整体设计

* 各部分设计
  + GUI



* + 总体设计



* 难点及解决方案
  + 读取PDF及Doc（Docx）格式的文档
    - 通过互联网搜索，引入外部包（POI，PDFBOX），已解决。
  + 逻辑查找
    - 设计一个语法分析器和一个结果组合器，解析并执行表达式中的逻辑符 号
  + 表达式中是否可以出现括号
    - 这个问题困扰了我好久，后来决定使用正则匹配来替代之。（使用正则 匹配后理论来说连关键字都可以省略，但是考虑到并非所有用户都可以 写出正确的正则表达式，故保留了关键字的与或非运算）

## 3 实现

* 设计时采用自上而下的设计方案，先从总体把握设计，然后设计细节，但在 实现过程用，我使用自下而上的实现方式，每实现一部分都进行测试，降低 后期维护成本。以下为测试代码：
* 1: package findIt;
* 2:
* 3: import java.lang.reflect.InvocationTargetException;
* 4: import javax.swing.SwingUtilities;
* 5: import javax.swing.UIManager;
* 6: import javax.swing.UnsupportedLookAndFeelException;
* 7:
* 8:
* 9: /\*\*
* 10: \* 测试grammerAnalysis.keyWordExtraction()
* 11: \* @author SkyFire
* 12: \*
* 13: \*/
* 14: */\**
* 15: *public class test222{*
* 16:  *public static void main(String [] args){*
* 17:  *ArrayList<String> arrl=new ArrayList<String>();*
* 18:  *Scanner in=new Scanner(System.in);*
* 19:  *String tempStr=in.next();*
* 20:  *in.close();*
* 21:  *tempStr=grammerAnalysis.keyWordExtraction(tempStr,arrl);*
* 22:  *for(String st:arrl){*
* 23:  *System.out.println(st);*
* 24:  *}*
* 25:  *System.out.println(tempStr);*
* 26:  *}*
* 27: *}*
* 28: *\*/*
* 29:
* 30:
* 31: /\*\*
* 32: \* 测试FindItFileList.getFileList()
* 33: \* @author SkyFire
* 34: \*
* 35: \*/
* 36: */\**
* 37: *public class test222{*
* 38:  *public static void main(String [] args){*
* 39:  *ArrayList<String> arrl=new ArrayList<String>();*
* 40:  *Scanner in=new Scanner(System.in);*
* 41:  *String tempStr=in.next();*
* 42:  *in.close();*
* 43:  *arrl=FindItFileList.getFileList(tempStr);*
* 44:  *for(String st:arrl){*
* 45:  *System.out.println(st);*
* 46:  *}*
* 47:  *}*
* 48: *}*
* 49: *\*/*
* 50:
* 51: /\*\*
* 52: \* 读取pdf测试
* 53: \* @author SkyFire
* 54: \*
* 55: \*/
* 56:
* 57: */\**
* 58: *public class test222{*
* 59:  *public static void main(String [] args){*
* 60:  *Scanner in=new Scanner(System.in);*
* 61:  *String tempStr=in.next();*
* 62:  *in.close();*
* 63:  *String arrl=readFile.readPDF(tempStr);*
* 64:  *System.out.println(arrl);*
* 65:  *}*
* 66: *}*
* 67:
* 68: *\*/*
* 69:
* 70: /\*\*
* 71: \* 读word
* 72: \* @author SkyFire
* 73: \*
* 74: \*/
* 75: */\**
* 76:
* 77: *public class test222{*
* 78:  *public static void main(String [] args){*
* 79:  *Scanner in=new Scanner(System.in);*
* 80:  *String tempStr=in.next();*
* 81:  *in.close();*
* 82:  *String arrl=null;*
* 83:  *try{*
* 84:  *arrl=readFile.readOffice(tempStr);*
* 85:  *}catch(XmlException e){*
* 86:  *}catch(OpenXML4JException e){*
* 87:  *}catch(FileNotFoundException e){*
* 88:  *}catch(Exception e){*
* 89:  *}*
* 90:  *System.out.println(arrl);*
* 91:  *}*
* 92: *}*
* 93: *\*/*
* 94:
* 95: /\*\*
* 96: \* 测试readText
* 97: \* @author SkyFire
* 98: \*
* 99: \*/
* 100: */\**
* 101: *public class test222{*
* 102:  *public static void main(String [] args){*
* 103:  *Scanner in=new Scanner(System.in);*
* 104:  *String tempStr=in.next();*
* 105:  *in.close();*
* 106:  *String arrl=null;*
* 107:  *try{*
* 108:  *arrl=readFile.readText(tempStr);*
* 109:  *}catch(FileNotFoundException e){*
* 110:  *}catch(Exception e){*
* 111:  *}*
* 112:  *System.out.println(arrl);*
* 113:  *}*
* 114: *}*
* 115: *\*/*
* 116:
* 117: /\*\*
* 118: \* 主类
* 119: \* @author SkyFire
* 120: \* @version 1.0
* 121: \*/
* 122: public class FindItMain{
* 123: public static void main(String [] args) throws InvocationTargetException, InterruptedException{
* 124: SwingUtilities.invokeAndWait(new Runnable() {
* 125: @Override
* 126: public void run() {
* 127: try {
* 128: UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
* 129: } catch (ClassNotFoundException e) {
* 130: e.printStackTrace();
* 131: } catch (InstantiationException e) {
* 132: e.printStackTrace();
* 133: } catch (IllegalAccessException e) {
* 134: e.printStackTrace();
* 135: } catch (UnsupportedLookAndFeelException e) {
* 136: e.printStackTrace();
* 137: }
* 138: }
* 139: });
* 140: FindItGui fgui=new FindItGui();
* 141: }
* 142: }

后来将此代码直接用于完成版本

* 关键代码注释
  + 语法分析器
  + 1: package findIt;
  + 2: import java.io.File;
  + 3: import java.io.IOException;
  + 4: import java.util.ArrayList;
  + 5:
  + 6: import org.apache.poi.openxml4j.exceptions.OpenXML4JException;
  + 7: import org.apache.xmlbeans.XmlException;
  + 8:
  + 9: /\*\*
  + 10: \* 语法分析器
  + 11: \* @author SkyFire
  + 12: \*/
  + 13:
  + 14: class FindItAnalysis{
  + 15: private String expression;
  + 16: private String dir;
  + 17: private String op;
  + 18: private ArrayList<String> keyWordVec=new ArrayList<String>();
  + 19: private ArrayList<String> fileList;
  + 20: private ArrayList<String> result;
  + 21: /\*\*
  + 22: \* @param gui GUI实例
  + 23: \*/
  + 24: void startAnalysis(FindItGui gui){
  + 25: */\**
  + 26:  *\* 获取逻辑表达式褐和目录*
  + 27:  *\*/*
  + 28: expression=new String(gui.getExpression());
  + 29: dir=new String(gui.getDir());
  + 30: */\**
  + 31:  *\* 测试目录是否存在*
  + 32:  *\*/*
  + 33: File file =new File(dir);
  + 34: if (!(file.exists()&&file.isDirectory())){
  + 35: gui.errorOut(expression,dir,"目录不存在！",1);
  + 36: return;
  + 37: }
  + 38: */\**
  + 39:  *\* 检验表达式中的引号*
  + 40:  *\*/*
  + 41: if(grammerAnalysis.quotesNum(expression)%2==1){
  + 42: gui.errorOut(expression,dir,"逻辑表达式中引号数量有误！",1);
  + 43: return;
  + 44: }
  + 45: */\**
  + 46:  *\* 提取关键字和逻辑操作符*
  + 47:  *\*/*
  + 48: op=grammerAnalysis.keyWordExtraction(expression,keyWordVec);
  + 49: */\**
  + 50:  *\* 测试逻辑操作符数量*
  + 51:  *\*/*
  + 52: if(op.length()!=keyWordVec.size()){
  + 53: gui.errorOut(expression,dir,"逻辑表达式中操作符数量有误！",1);
  + 54: return;
  + 55: }
  + 56: */\**
  + 57:  *\* 测试逻辑操作符合法性*
  + 58:  *\*/*
  + 59: if(!grammerAnalysis.opTest(op)){
  + 60: gui.errorOut(expression,dir,"逻辑表达式中操作符有误！",1);
  + 61: return;
  + 62: }
  + 63: */\**
  + 64:  *\* 获取文件列表*
  + 65:  *\*/*
  + 66: gui.receMsg(expression, dir, "正在获取文件列表");
  + 67: fileList=FindItFileList.getFileList(dir);
  + 68: */\**
  + 69:  *\* 初始结果为所有文件*
  + 70:  *\*/*
  + 71: result=new ArrayList<String>(fileList);
  + 72: */\**
  + 73:  *\* 结果生成*
  + 74:  *\*/*
  + 75: gui.receMsg(expression, dir, "正在筛选并进行逻辑运算");
  + 76: for(int i=0;i<op.length();++i){
  + 77: try{
  + 78: */\**
  + 79:  *\* 获取并组合匹配结果*
  + 80:  *\*/*
  + 81: result=mergeIt.merge(result, FindItMatch.getMatch(fileList,(String) keyWordVec.toArray()[i]),op.charAt(i));
  + 82: }catch(IOException e){
  + 83: gui.errorOut(expression,dir,e.toString(),0);
  + 84: }catch(OpenXML4JException e){
  + 85: gui.errorOut(expression,dir,e.toString(),0);
  + 86: }catch(XmlException e){
  + 87: gui.errorOut(expression,dir,e.toString(),0);
  + 88: }
  + 89: }
  + 90: */\**
  + 91:  *\* 输出结果*
  + 92:  *\*/*
  + 93: gui.resultOut(expression,dir,result);
  + 94: }
  + 95: }
  + 目录获取器
  + 1: package findIt;
  + 2: import java.io.File;
  + 3: import java.util.ArrayList;
  + 4: /\*\*
  + 5: \* 获取目录列表
  + 6: \* @author SkyFire
  + 7: \*
  + 8: \*/
  + 9:
  + 10: class FindItFileList{
  + 11: /\*\*
  + 12: \* 获取目录列表
  + 13: \* @param dir 目录
  + 14: \* @return 所有文件列表
  + 15: \*/
  + 16: static ArrayList<String> getFileList(String dir){
  + 17: ArrayList<String> fileList=new ArrayList<String>();
  + 18: File file=new File(dir);
  + 19: if (file.isDirectory()) {
  + 20: File[] dirFile = file.listFiles();
  + 21: for (File f : dirFile) {
  + 22: */\**
  + 23:  *\* 如果是目录，则递归获取，加入当前列表*
  + 24:  *\*/*
  + 25: if (f.isDirectory()) {
  + 26: fileList.addAll(getFileList(f.getAbsolutePath()));
  + 27: } else {
  + 28: fileList.add(f.getAbsolutePath());
  + 29: }
  + 30: }
  + 31: }
  + 32: return fileList;
  + 33: }
  + 34: }
  + 结果组合器
  + 1: package findIt;
  + 2: import java.util.ArrayList;
  + 3:
  + 4: /\*\*
  + 5: \* 结果组合器
  + 6: \* @author SkyFire
  + 7: \*
  + 8: \*/
  + 9: class mergeIt{
  + 10: /\*\*
  + 11: \* 逻辑运算
  + 12: \* @param ls1 列表1
  + 13: \* @param ls2 列表2
  + 14: \* @param op 运算符
  + 15: \* @return 逻辑运算结果
  + 16: \*/
  + 17: static ArrayList<String> merge(ArrayList<String> ls1,ArrayList<String> ls2,char op){
  + 18: ArrayList<String> tempArr=new ArrayList<String>();
  + 19: switch (op){
  + 20: case '&':
  + 21: */\**
  + 22:  *\* 查找相同的，加入*
  + 23:  *\*/*
  + 24: tempArr=new ArrayList<String>();
  + 25: outer:
  + 26: for(String t1:ls1){
  + 27: for(String t2:ls2){
  + 28: if(t1.equals(t2)){
  + 29: tempArr.add(t1);
  + 30: continue outer;
  + 31: }
  + 32: }
  + 33: }
  + 34: break;
  + 35: case '|':
  + 36: */\**
  + 37:  *\* 加入第一个列表，然后查找第一个列表中没有的，加入*
  + 38:  *\*/*
  + 39: tempArr=new ArrayList<String>(ls1);
  + 40: outer:
  + 41: for(String t2:ls2){
  + 42: for(String t1:ls1){
  + 43: if(t1.equals(t2)){
  + 44: continue outer;
  + 45: }
  + 46: }
  + 47: tempArr.add(t2);
  + 48: }
  + 49: break;
  + 50: case '^':
  + 51: */\**
  + 52:  *\* 查找第二个列表中没有的，加入*
  + 53:  *\*/*
  + 54: tempArr=new ArrayList<String>();
  + 55: outer:
  + 56: for(String t1:ls1){
  + 57: for(String t2:ls2){
  + 58: if(t1.equals(t2)){
  + 59: continue outer;
  + 60: }
  + 61: }
  + 62: tempArr.add(t1);
  + 63: }
  + 64: break;
  + 65: }
  + 66: return tempArr;
  + 67: }
  + 68: }

## 4 实验测试

* 安全及稳定性：在设计过程中，已经进行各功能模块的测试，完成后自己写了几个文件进行 测试，目前（提交前）没有安全及稳定性问题。
* 效率：在效率方面确实存在一些问题，筛选每个关键字都需要对所有文件进 行重新读入，使用第三方库读取PDF及Doc文件效率也略低。
  + 解决方案及其可行性 开发过程中有着尝试将读文件任务并发处理，但是那样会需要更加庞大的 内存空间，会影响到程序的稳定性，所以不可行，正在寻找一种折中的方 案。
* 程序的可移植性以及可扩充性
  + 可移植性可以先不谈，因为java本来就是一种可移植编程语言
  + 可扩充性 这个程序的功能扩充主要是对支持文件类型的种类进行扩充，增加文件类 型需要在readFile.java中添加读取某种类型文件的方法，以及在 findItMatch.java中添加对文件类型判断的代码

## 5 关于本程序

* + - 程序已经实现指定的基础功能及可选功能，并做了一些扩充

FindIt 说明文档

一、功能描述

筛选指定目录下满足指定条件的文档

二、特色

1.多任务

可同时进行多个筛选任务

2.逻辑筛选

可以将关键字通过‘&’、“|”、‘^’组合起来进行逻辑筛选，如查找同时含有“西安”、“大学”,但是不包括“外国语”的文档，则可以输入表达式：&"西安"&"大学"^"外国语"

3.支持多种文件格式筛选

目前支持.txt，.pdf，.doc，.docx，.ppt，.pptx，.xls，.xlsx文件格式的筛选

4.支持正则筛选

关键字可以为正则表达式，如：&"西安.\*大学" 可以筛选出含有“西安(任意字符)大学” 的所有结果

5.操作简单

对于一般关键字可以在关键字框直接输入关键字，点击任意逻辑符号按钮，将在表达式框里自动生成表达式，如：输入关键字“西安”，点击 & 按钮，会在表达式框内生成&"西安"

三、关于版权

此项目中引入的扩展JAR：pdfbox-app-1.8.7.jar、xmlbeans-2.3.0.jar、poi-3.11-beta3-20141111.jar、poi-examples-3.11-beta3-2.141111.jar、poi-excelant-3.11-beta3-20141111.jar、poi-ooxml-3.11-beta3-20141111.jar、poi-ooxml-schemas-3.11-beta3-20141111.jar、poi-scratchpad-3.11-beta3-20141111.jar。

以上包的版权属于原作者，在此向开发者们致敬，感谢你们开发出如此强大的jar

此项目为开源项目，可随时在<http://github.com/skyfireitdiy/FindIt>获取最新的源代码。

--SkyFire

## 6 小组分工

## 小组名称：Java研发小组-代号J

## 小组成员：王茂斌（03121303），张世欣（），杨林春（03121304），纪志鹏（），王玉龙（），刘易斯（03121039）

## 工作模式：通过QQ讨论组方式讨论并分配任务

## 分工：

## 王茂斌：整体设计，撰写文档，语法分析器编码，驱动编码，整体测试

## 张世欣：GUI设计，GUI编码

## 杨林春：文件匹配器编码（包括文件读取）

## 刘易斯：文件列表获取器编码

## 王玉龙：查找pdf，doc等格式读取方法，提供外部包及其使用方法

## 纪志鹏：各组件分析，测试