**《Web应用开发》课程大作业报告评阅表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 项目 | 报告写作和格式规范，无抄袭情况（30%） | 符合课程要求（20%） | 功能合理且完善，代码无抄袭情况（30%） | 具有特色或创新性（10%） | 按时提交材料，材料无遗漏（10%） | 总分（100%）  签名： |
| 得分 |  |  |  |  |  |  |
| 得分说明/评语 |  |  |  |  |  |  |

2023-2024第2学期

《Web应用开发》

课程大作业报告

大作业题目：Go语言实现单页简历解析SPA应用 （命题/自选）

所在教学班： XX班 姓名： 张三 学号： 123

1. 功能概述

这是一个基于Gin框架构建的简单web应用，其主要功能是允许用户上传PDF格式的简历文件，然后从这些文件中提取文本内容，并在服务器上生成一个HTML格式的解析结果页面。以下是程序的功能概述：

1. Web服务器启动：程序使用Gin框架创建一个web服务器，并在指定端口（默认为8080）监听HTTP请求。

2. 路由设置：程序定义了一个路由`/upload`，用于处理两种类型的HTTP请求：

GET /upload`：当用户访问这个路由时，服务器将返回一个HTML页面，允许用户上传PDF文件。

`POST /upload`：当用户提交表单上传PDF文件时，服务器将接收并处理这个请求。

3. 文件上传处理：

用户通过GET请求访问上传页面后，可以选择一个PDF文件上传。

- 用户提交表单后，服务器通过POST请求接收文件，并进行处理。

4. PDF文件解析：服务器接收到PDF文件后，使用`ledongthuc/pdf`库打开并读取文件内容。程序遍历PDF的每一页，提取文本内容。

5. 关键词高亮：程序定义了一系列关键词，如“姓名”、“专业”、“电话”等，使用正则表达式在提取的文本中匹配这些关键词，并将它们包装在`<b>`标签中以实现高亮显示。

6. 内容转换与保存：提取并处理的文本内容被转换成HTML格式，然后保存到服务器上的一个新文件中，文件名默认为`save.html`。

7. 错误处理：在文件上传和解析过程中，如果遇到任何错误（如文件打开失败、PDF解析失败、文件保存失败等），服务器将返回相应的错误信息。

8. 结果展示：解析成功后，服务器将生成的HTML文件名传递给`result.html`模板，渲染并展示解析结果。

9. 静态文件服务：程序还配置了静态文件服务，用于提供静态资源，如CSS样式表、JavaScript脚本等。

10. 日志记录：程序使用标准库`log`记录关键操作和错误信息。

这个程序是一个基本的示例，展示了如何结合Gin框架和PDF处理库来创建一个处理文件上传和解析的web应用。它可以根据需要进行扩展和定制，以满足更复杂的业务需求。

二、实现原理和方法

技术栈

1. Go语言：作为服务器端编程语言。

2. Gin框架：Go语言的Web框架，用于快速搭建Web服务器和处理HTTP请求。

3. pdf库：用于处理PDF文件，提取文本内容。

4. regexp库：用于在文本中匹配正则表达式，实现关键词高亮。

5. html/template库：用于渲染HTML模板。

6.os库：用于文件操作，如创建和删除临时文件。

7. io库：用于执行底层的I/O操作，如文件复制。

8. log库：用于记录日志信息。

实现过程:

服务器初始化:使用Gin框架创建Web服务器，并设置监听端口。定义`/upload`路由，处理GET和POST请求。GET请求处理，用户访问`/upload`时，服务器渲染并返回一个上传页面（`upload.html`）。POST请求处理用户提交表单上传PDF文件时，触发`/upload`路由的POST处理程序。文件上传处理解析multipart表单，获取上传的文件。检查是否有文件上传，如果没有，则返回错误。创建临时文件为上传的文件创建一个临时存储文件。使用`pdf`库读取PDF文件的每一页。提取每一页的文本内容，并累加到一个字符串构建器中。关键词高亮定义一组关键词。使用正则表达式匹配文本中的关键词，并用`<b>`标签包裹以实现高亮。结果保存将处理后的HTML内容保存到一个文件（例`resume\_output.html`）。错误处理在文件上传、解析和保存过程中，如果遇到错误，返回相应的错误信息。渲染一个结果页面（`result.html`），展示解析后的简历内容。

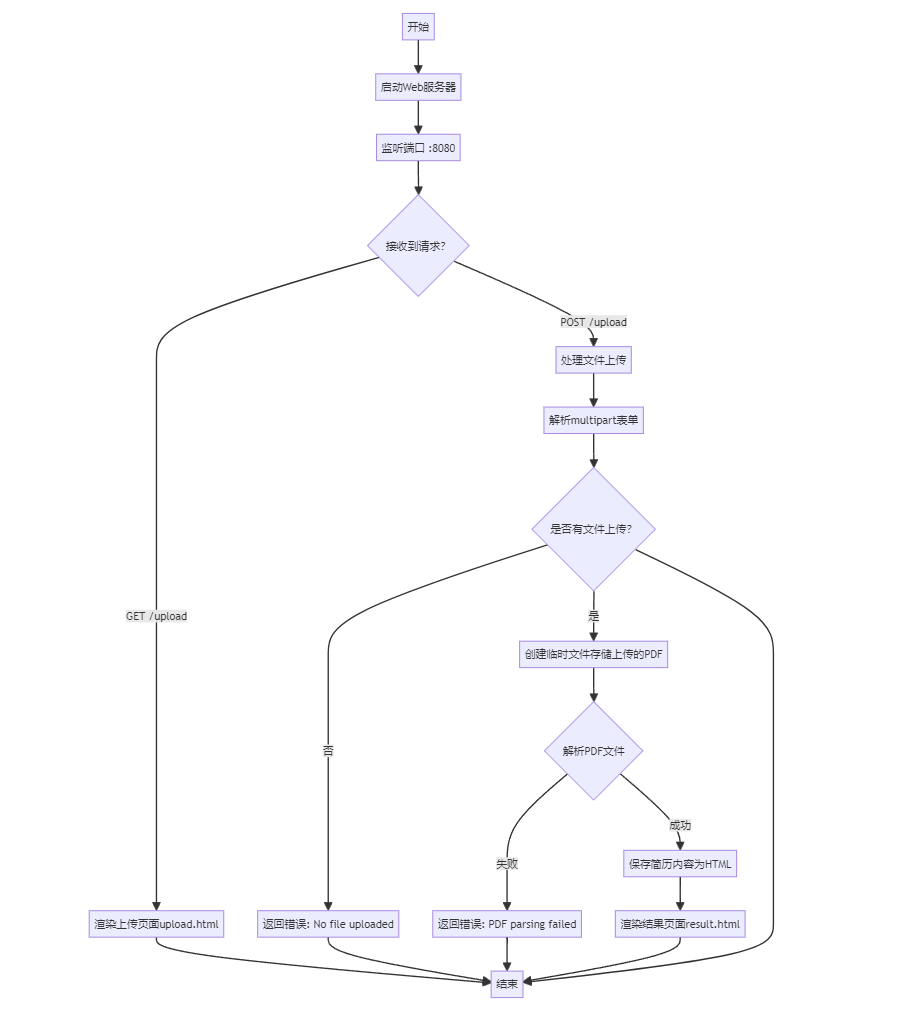
这个程序是一个端到端的示例，实现了从接收用户上传的文件到展示处理结果的完整流程。

图1 简历解析系统UML图

1. 实验和测试环境

GoLand

Go 1.22.3



图2生产环境以及配置图

四、运行和测试结果

描述一下结果，附带效果截图

简历上传功能

|  |
| --- |
| 图3 简历上传功能运行效果    图4 简历解析功能运行效果 |

五、结论和总结

功能性：此应用程序成功实现了其核心功能，即允许用户上传PDF简历，提取文本内容，并在服务器上生成HTML格式的解析结果页面。用户交互：通过简洁的Web界面，用户可以轻松上传文件并获取处理结果，提供了良好的用户体验。技术实现：Go语言结合Gin框架为创建高效、高性能的Web服务提供了优秀的支持。ledongthuc/pdf库有效地处理了PDF文件的读取和文本提取。可扩展性：代码结构清晰，易于扩展和维护。未来可以添加更多功能，如更复杂的文本分析、用户认证、文件存储解决方案等。错误处理：程序中包含了基本的错误处理机制，能够对可能出现的问题给出反馈，但可能需要更详细的用户提示和日志记录以便于问题诊断。Gin框架因其性能和易用性成为构建Web应用的良好选择。ledongthuc/pdf库为PDF文件处理提供了有效的支持。这个应用程序是一个实用的工具，可以作为更大型项目的基础，或者作为一个学习示例，展示如何使用Go语言和相关库来处理Web应用中的文件上传和解析任务。

附：关键代码（不超过100行）

// HighlightKeywords函数用于在简历文本中高亮关键词

func HighlightKeywords(text string) string {

// 定义一组关键词

keywords := []string{"姓名", "专业", "电话", "邮箱", "教育背景", "个人获奖情况", "感兴趣的研究方向", "项目经历"}

// 编译正则表达式，用于匹配关键词

re := regexp.MustCompile(`\b(` + strings.Join(keywords, "|") + `)\b`)

// 替换文本中的关键词为高亮格式(HTML bold标签)

text = re.ReplaceAllString(text, `<b>$1</b>`)

// 将文本中的换行符替换为HTML的<br>标签

text = strings.ReplaceAll(text, "\r\n", "<br>")

text = strings.ReplaceAll(text, "\n", "<br>")

return text

}

// uploadHandler是处理文件上传的HTTP处理器函数

func uploadHandler(c \*gin.Context) {

// 检查请求方法是否为GET

if c.Request.Method == "GET" {

// 如果是GET请求，渲染upload.html页面

c.HTML(http.StatusOK, "upload.html", gin.H{})

return

}

// 如果请求方法是POST，则处理文件上传

if c.Request.Method == "POST" {

// 解析multipart表单数据

form, err := c.MultipartForm()

if err != nil {

// 如果解析表单失败，返回错误信息

c.JSON(http.StatusBadRequest, gin.H{"error": "Failed to parse form"})

return

}

// 从表单中获取文件列表

files := form.File["file"]

if len(files) == 0 {

// 如果没有文件被上传，返回错误信息

c.JSON(http.StatusBadRequest, gin.H{"error": "No file uploaded"})

return

}

// 处理第一个上传的文件

file := files[0]

// 创建一个临时文件存储上传的PDF文件

tempFile, err := os.CreateTemp("", "upload-\*.pdf")

if err != nil {

// 如果创建临时文件失败，返回错误信息

c.JSON(http.StatusInternalServerError, gin.H{"error": "Temporary file creation failed"})

return

}

defer tempFile.Close()

defer os.Remove(tempFile.Name()) // 确保在函数结束时删除临时文件

// 打开上传的文件内容

src, err := file.Open()

if err != nil {

// 如果打开文件失败，返回错误信息

c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to open file"})

return

}

defer src.Close()

// 将上传的文件内容复制到临时文件

\_, err = io.Copy(tempFile, src)

if err != nil {

// 如果文件保存失败，返回错误信息

c.JSON(http.StatusInternalServerError, gin.H{"error": "File saving failed"})

return

}

// 调用parsePDF函数解析PDF文件

resume, err := parsePDF(tempFile.Name())

if err != nil {

// 如果PDF解析失败，返回错误信息

c.JSON(http.StatusInternalServerError, gin.H{"error": "PDF parsing failed"})

return

}

// 将解析后的简历内容保存为HTML文件

savedFile := "resume\_output.html"

err = SaveContentAsHTML(resume.Content, savedFile)

if err != nil {

// 如果保存文件失败，返回错误信息

c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to save result"})

return

}

// 渲染result.html页面，并传递解析后的简历数据和保存的文件名

c.HTML(http.StatusOK, "result.html", gin.H{"resume": resume, "savedFile": savedFile})

}

}

请务必提交以下电子版文件（缺一不可）：

1.word版本报告（.doc或.docx）。除了电子版外，还需要打印并提交纸质版本。

2.将word版本报告转为PDF格式（.pdf），请确保与word版本内容一致。

3.全部源代码打包到一个文件中（.zip）。

4.关键代码（100行左右）。加注释！

5.运行效果截图（png或jpg格式），选择1张你认为效果最好的提交。更多的效果截图可以放在word版本报告中。

文件命名要求：

以上所有文件请以学号开头进行命名：

例如：

32019070299-张三.doc

32019070299-张三.pdf

32019070299-张三.zip

32019070299-张三.go

32019070299-张三.png

以上文件请分别以附件方式上传。

注意：

1.源代码中请使用相对路径，不要使用绝对路径（绝对路径会导致老师这里无法运行）。

2.请务必包含全部需要的文件，资源文件等（但不要包含没有用的多余文件），尽量不要使用CDN（有可能也会无法运行）。

3.如果你的代码中包含或依赖可能在老师这里无法运行的环境（例如有数据库等），请在word版报告中的"实验和测试环境"部分中说明清楚。

4.请在word版本报告中包含你的运行效果截图（以免老师这里无法运行导致看不到你的效果），截图请涵盖你实现的所有功能，截图可以是整个页面的布局图，也应当包含局部细节图（用来表明你实现的具体功能，局部图能更好地展现细节，同时能让老师重点关注到你想表现的内容，而不是让老师在一幅图中去猜你到底想展现什么内容）。

5.word版报告（及PDF）务必阐述清晰（你要把老师看作是完全不了解你做了什么的人，然后想办法把你做的内容说清楚，说明白，阐述请用书面语，不要过于口语化），排版规范（字体，字号，图下方要有图1.xxx，图2.xxx这样的题注）。题注是"指出现在图片下方的一段简短描述"，表格的题注要放在表格的上方。题注也会帮助老师理解你的图和表想表达的内容。

6.word版报告没有必要粘贴源代码（如有，请加注释）。

7.请提交最终版本程序（删除代码中没有用的内容，删除不需要和没用的多余文件）。