MicroLayout微雕生成器设计文档

系统功能设计

功能需求

大学的部分课程的final test会以开卷考试进行，但是可能会要求只能把资料记录在一张A4纸上。同学们不论是自己手写资料、还是使用word进行排版都十分麻烦，容易耽误复习时间。然而，使用markdown语言编排复习资料会比较容易。  
  
基于上述几点，MicroLayout微雕生成器被设计用于将markdown文件自动排版并整合于一个不超过两页的docx文件，可以帮助在校大学生快速制作期末复习资料，节约复习时间。

后端行为结构

实现微雕生成器的主要工作在于后端设计，总体框架很简单，分为读取文件、处理文件、保存生成的文件三部分。其主体同时也是难点的在于处理文件。  
  
![image-20210416154504442](./picture/后端1.png)  
  
处理文件的流程如下。首先读取文件内容的长度，然后根据长度设置生成docx文件的字体大小以及栏目数。例如，根据反复的测试发现，文件长度在15000到22500区间内时，字体大小设置为5，分成4栏会取得比较好的观感。接着调用python-docx、PyPDF2等外部包以及文件生成模块生成目标docx、pdf文件。然而，先前对字体和栏目的大致设置并不能保证目标文件页数不超过两页，这里需要再次判断，如果页数超过两页则需要设当调小字体重新生成文件，直到页数降到两页或字体已经无法减小（后面的情况说明输入文件长度特别大，已经无法生成微雕）。  
  
之所以需要生成pdf文件，一方面是pdf文件更易阅读、打印；另一方面，由于docx文件对象不支持页码查询，需要先将docx文件转化为pdf文件，才能从pdf文件中读取文件页数，以便后续的操作。  
  
![image-20210416160216641](./picture/后端2.png)  
  
生成docx及pdf文件的设计流程如下。模块输入读取的文件位置、分栏数、字体大小等参数，首先创建docx文件对象，进行文件的一些参数设置。接着，将读取的文件内容按行分割，通过正则表达式判断内容是否为标题，若是，则根据匹配的标题级数填入docx文件中并生成一个新的段落，若不是则说明该行为正文内容，直接填入已有的段落中。  
  
当所有行处理完毕后即可保存得到的docx文件，接着根据docx文件生成pdf文件即可。最后读取生成的pdf文件页数将其返回文件处理模块。  
  
![image-20210416165521845](./picture/后端3.png)

前端行为结构

本项目的前端用户行为十分简单。用户打开网页就可以直接上传需要转换的markdown文件，上传成功后进入下载页面，用户可以选择下载docx文件、下载pdf文件或者返回主页重新上传。  
  
为了美化界面，前端的html使用了bootstrap网站提供的css模板。  
  
![image-20210416172142298](./picture/前端.png)

前后端交互模式

由于项目采用了django框架，主要以request对象、render方法进行前后端的交互。  
  
![image-20210416180309986](./picture/交互.png)

设计展望

功能添加

如今项目已经能够正确的将markdown文件的所有内容转化为不超过两页的docx文件和pdf文件，并且将markdown文件的各级标题转化为docx文件的对应的标题。但是仍然有一些可行的功能完善：  
  
- 加粗、倾斜等字体的转换：由于文件处理是按行读取的，markdown的加粗（两对\*\*之间）可能会涉及多行。要想实现需要将多行的内容进行关联，还需要暂时储存文件内容作为中间量  
- 文件批量转换：当一次上传多个文件时，需要对request对象进行更深入的处理，下载模块也需要连带改变  
- 数学公式转换：由于输入文件为复习资料的形式，可能会出现数学公式。如果想要转换到docx文件的数学公式将会是一个极为繁琐的工程。开源python包pandoc能够实现将md文件转docx文件，或许能够查阅其源代码进行参考

性能优化

经过测试，程序主要的运行时间在于pdf文件的生成，生成一个pdf文件大约需要3至5秒。倘若生成文件的页数超过2页，则需要从头开始生成一个新的pdf文件，在最坏情况下程序可能需要运行接近一分钟。针对于此，可进行如下策略调整：  
  
- 不生成pdf文件。只生成docx文件的速度非常快，用户几乎不需要等待时间就可以处理完毕。但是docx文件并不能直接读取页数，也就无法判断生成文件的页数是否大于2。  
- 降低字体的作用长度下限，比如现在内容长度大于15000会使用5级字体，可以调整为长度大于12500就使用5级字体，以此类推。缺点是会造成生成文件留有大量未使用空间  
- 通过机器学习算法尽可能减小生成pdf文件的次数，例如综合分析原文件的行数、中英文字符所占比例、生成文件超出部分的大小等等；但是实现起来较为困难。