Computer Science and Application 计算机科学与应用, 2022, 12(7), 1711-1718
Published Online July 2022 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2022.127171

Hans汉斯

基于MWOL编程支持批量生成文档

刘　健[1,2]，戴杨杨[2]

[1]四川开放大学，四川　成都
[2]四川华新现代职业学院，四川　成都

收稿日期：2022年6月4日；录用日期：2022年6月29日；发布日期：2022年7月6日

摘　要

本文提出在模板文档Template中加入标识字符，由进程调用MWOL替换标识字符，批量、快速生成众多目标文档的方法。项目严格按照分层思想设计，降低各层之间的耦合度，每层都可以适应项目的变化。项目将代码与模板文档Template的关系分开，单独由配置文件去协调模板文档Template内容的变化，甚至是模板文档Template的更换。这样的设计可以提高项目的鲁棒性、适应性，可以将技术人员由非技术的工作中抽身出来，由非技术人员负责制作模板文档Template。当新项目出现时，下层的设计可以在短时间内少量改动或者不改动，便可满足新项目的要求。

关键词

MWOL，模板，分层，重载

Build a Batch of Documents Based on MWOL Programming Support

Jian Liu[1,2], Yangyang Dai[2]

[1]Sichuan Open University, Chengdu Sichuan
[2]Sichuan Huaxin Modern Vocational College, Chengdu Sichuan

Received: Jun. 4th, 2022; accepted: Jun. 29th, 2022; published: Jul. 6th, 2022

Abstract

A solution on building a batch of target documents quickly through calling MWOL by process to replace marked strings which already added in the template document is presented in this paper. According the principle of layering design, it reduces the coupling between each two layers to adapt new situation in the project. It cut down the relationship between the codes and the tem-

文章引用: 刘健, 戴杨杨. 基于 MWOL 编程支持批量生成文档[J]. 计算机科学与应用, 2022, 12(7): 1711-1718.
DOI: 10.12677/csa.2022.127171

plate document by independent configuration file to suit the content changing in the template document, even the whole template document changing. Office clerks, instead of technical staffs, making the template document would promote the project's robustness and flexibility. The design of lower layer could match the new request with few amending, even not amending, while new project coming.

Keywords

MWOL, Template, Layering, Overload

1. 引言

办公软件的应用早已不再局限于计算机专业人士的领域，这项技能是办公室白领的基本技能。然而当数量众多的文档采用相同格式，少量文字变化时，由办公软件单个地人工输入处理显然不合适。

通过邮件合并方式生成的文档默认为全部内容合并为单个文档，没法满足单条数据对应生成单个文档的要求。即使在模板文档 Template 里设定特殊格式和内容，可以在生成后把单个文档分割为满足要求的众多单个文档。但因为有事先加入的特殊格式和内容，不可避免地造成目标文档内容不尽如人意。

另外，如果要求目标文档的最终文件格式不是普通办公软件的默认格式，而是其他格式，这会带来巨大困扰。数量可观的单个文档转换格式是一项耗时、耗力，又容易出错的工作。虽然网络中有众多的格式转换软件提供免费服务，但是转换出来的结果基本都自带水印，问题就是用户不会接受带有第三方广告的结果。

本文借鉴“软件即服务”[1]和综合解决方案[2]的思想及多种软件开发方法[3] [4] [5]，提出的解决方案为通过办公软件的编程支持，引用 MWOL，即 Microsoft Word Object Library，可以在非常短的时间内批量生成目标文档。

2. 项目背景

目标文档的生成有两种途径：一、由代码根据模板文档 Template 的排版来生成目标文档里的全部内容；二、代码只替换模板文档 Template 里的关键内容再生成目标文档。显然第二种方法才是上优选择。

模板文档 Template 的格式选择也是一个问题：Word 还是 PDF？经过试验，PDF 模板不能很好满足文字的排版，Word 模板在这方面有天然的优势。故本项目选择以 Word 格式制作模板文档 Template，其内容排版类似于图 1。

从图 1 可以看出，在需要文字替换的地方打上标签。不论后期是需要插入普通字符串，还是图片格式，都是同样类型的标识字符，区别只在于替换时的方法不同。

3. 各层介绍

项目严格按照分层思想来进行设计，在基础数据之上还有三层，分别为数据访问、业务逻辑和业务。

XXXX 单位
XXXX 年 XXXX 证

证号	ticket			image1
姓名	name	性别	gender	
身份证号	identity			
报名号	cand			
部门	depart			
类别	type	日期	XXXX 年 X 月 X 日	

科目	时间	场地	
科目一	9:00-11:30	site1（场地：number1，顺序：serial1）	image2
科目二	13:30-17:30	site2（场地：number2，顺序：serial2）	

Figure 1. Template document
图 1. 模板文档 Template

3.1. 基础数据

项目的基础数据有三个特点：来源惟一、结果完整、用途单纯。来源惟一是指数据全部由负责部门统一提供，而不是由多个部门分散提供。结果完整是指提供的数据中，已经包含全部的字段，不需要从多个来源处组合生成完整数据。用途单纯是指项目的数据只用于本项目，用完即止，不会用于其他项目。

基于上述三个特点，数据的存在形式选项很多，可以是办公软件的电子表格 Excel，也可以是半专业的数据库 Access，还可以是专业领域的数据库。考虑到数据的安全性和完整性，存于数据库是首选。

数据库中表的字段可以与来源数据的字段一一对应，只是需要增加字段“时间戳”。因为数据的内容不会一成不变，会因为各种各样的原因产生变化。当数据内容变化后，可以依据“时间戳”来访问产生变化的记录，而不需要访问全部的数据。

第二个增加的字段是“标识”，此字段的作用是用于最终文档的文件名。生成原理是选择具有惟一性的字段，不一定是主键，在内容中间插入随机字符，最后生成 MD5 结果。即使字段的内容具有连续性，如此生成后，也没法用反推算法猜出相邻文档的文件名。

3.2. 数据访问

本层可以分为两块，一块存在于数据库内部，一块存在于数据库外部。

3.2.1. 数据库内部

存储过程相比嵌入式的 SQL 代码具有高效性与安全性，由存储过程访问底层数据可以比在代码中访问底层数据具有更快的运行速度，也可以防止 SQL 注入等攻击。

由存储过程访问底层数据还有一个优点是 SQL 代码独立于项目的程序代码，降低了相互间的耦合性。当数据库中的结构发生变化时，只要存储过程与程序代码的接口不变，那么只需要变动存储过程的 SQL 代码，而不需要变动程序代码，也就不需要再次编译生成可执行文件和再次部署。

同时，存储过程可以在提取基础数据时直接过滤，只留下符合条件的数据，可以防止大量无效数据进出数据库，降低数据库服务器与业务服务器之间的流量。

3.2.2. 数据库外部

数据库外部主要由三个类组成，第一个类与数据库联系紧密，第二个类用于二维码图像的生成，第

三个类负责文档的读写。

数据库类的方法是读和写两方面。读数据库的方法有 5 个重载，根据参数的不同，可以返回 int 和 DataTable 类型的值。各个重载方法的相同参数都为连接字符和存储过程名称，不同的参数可以为单个字段与单个值，也可以为字段数组与值数组，还可以为字段与值共同组成的键值对 KeyValuePair。写数据库的方法有 6 个重载，也可以有各个不同参数，返回值基本都为逻辑型。相同参数同样为连接字符和存储过程名称，不同参数可以根据需求增加键值对、DataTable 及时间戳等。返回值都用于判断数据库写入操作成功与否。

二维码图像类需要先引入第三方的类库 QRCoder，再生成 3 个重载方法。同样道理，其中一个相当于是主方法，其他方法通过各种变形与主方法相适应，并调用主方法以实现最终目的。其中主要的参数为二维码代表的信息、二维码中间的小图标、尺寸等。生成的结果可以是磁盘中的图像文件，也可以是内存的图像流。

文档读写类其实是关于 Word 文档和 Text 文档的两个类，合并于同一个代码文件中，在代码文件首部引入 Microsoft Office Object Library 和 Microsoft Word Object Library 两个类。Word 文档类的方法也有 3 个重载，主体思想为先创造两个实例，即 Word.Application 实例和 Word.Document 实例，然后将 Word.Application 实例打开目标文档的返回值赋予 Word.Document 实例，此后便可在 Word.Document 实例中进行相应的书签替换操作。此时，对模板文档 Template 中的文本书签逐一判断存在与否，若存在方法参数中对应的文本书签，便使用 get_Item 方法获取文本对象用于替换书签位置的文本内容。然后同样对目标文档中的图像书签逐一判断存在性，若存在方法参数中的对应的图像书签，也使用 get_Item 方法获取图像书签对象并选中，用新生成的 Word.InlineShape 图像对象去替换对应的图像书签。如果 Word.InlineShape 图像尺寸不合规格，还需要调整高度 Height 和宽度 Width。

若目标要求为 PDF 格式，则有两个类的方法提供格式转换，其一为在生成新的 Word 文档后，再次用 Word.Document 实例的 ExportAsFixedFormat 方法完成转换；其二为将这两步功能合并到同一个方法中，直接输出 PDF 文档，不需要两次打开 Word 文档两次，可以节省 CPU 和磁盘的时间。

文档读写类文件后半部分的 Text 文档类只有写功能，没有读功能，因为这个类只是为上层提供写入日志的功能。这个类里的方法是 4 个重载，主要功能就是记录以参数形式传递来的、DataRow 形成的字符串，并在字符串前面打上时间戳，最后写入 CSV 文件。当然，参数类型不会局限于 DataRow，也可以是 DataTable、字符串数组，或者是这些参数的组合形式。

上述三个类的共同特点是全部涉及磁盘 I/O 的操作。不管是文件的 I/O，还是数据库的 I/O，这些都是高频率出错的操作，所以在这些类的方法里都统一采用 try…catch…finally 块来处理异常，预防因为磁盘文件出错或数据库访问出错而造成程序崩溃。

3.3. 业务逻辑

在数据访问层的各个类都是各司其职，有点各自为政的意思。在这一层，就要把数据访问层的类组合使用，以达到中间结果的目的。此层主要由两个类组成：二维码标识类和文档类。

二维码标识类的构成主要是 3 重载的二维码标识方法，可以接受以 DataRow 类型为主体的参数，也可以接受以连接字符串和存储过程名为主体的参数，其他参数是关于二维码图像的配置参数，诸如路径、名称、像素、尺寸等。当主体参数为 DataRow 类型时，是把 DataRow 里的信息直接注入到二维码图像里，此时只调用数据访问层的二维码图像类。当主体参数为连接字符串和存储过程名时，需要同时调用数据访问层的二维码图像类和数据库。

文档类的组成方法比较多，有文档创建方法、PDF 文档创建方法、数据库读取方法、日志记录方法。

文档创建方法是 3 个重载，其功能有三个方面。第一方面为调用数据访问层的数据库读取方法，把数据从数据库提取出来，在内存中形成 DataTable 格式数据。第二方面为将模板文件 Template 拷贝一份，以数据库内的“标识”字段内容命名。第三方面为调用数据访问层的文档读取类的写入方法，将 DataTable 格式数据替换以“标识”字段内容命名文档里的相应书签内容。

PDF 文档创建方法的功能与文档创建方法非常类似，区别在于最后生成的文档格式不同，其可以直接生成 PDF 格式的文档。

数据库读取方法为调用数据访问层的数据库读取方法，形成 DataTable 格式数据以供其他类方法使用。

日志记录方法有 3 个重载，也是调用下层文本文档类写入方法，将时间戳与 DataRow 数据以循环的方式形成一条字符串往下层传，然后形成日志文件的一行内容。

此层每个方法里加入大量条件判断语句，目的是提高程序的鲁棒性。首先是对数量众多的输入参数进行逐一的判断，防止无效参数参与后期的运算。其次对文件的拷贝操作提供可数范围内的、循环的线程睡眠等待，防止模板文件 Template 拷贝失败造成内容替换失败。还有是对“标识”字段命名的文档内容替换后返回的布尔值进行判断，可以及时了解下层操作的成功与否。

另外，此层的类方法都要提供大量、以自然语言表述的错误信息。当上层方法调用这些方法时，可以把这些错误信息逐层向上反馈给最终用户。因为是以自然语言表述，最终用户没有阅读障碍，同时还可以屏蔽下层及本层出错的原始信息，可以提高使用者的用户体验与评价。

3.4. 业务

业务层的任务是建立与用户沟通的人机界面，同时要完成基础参数的设置。

基础参数的设置有两种途径：一是硬编码的参数跟着代码走；二是非硬编码的参数在代码之外。显然，放在代码之外才是最优选择。这里可以设置的参数没有数量限制，最关键的参数是时间戳和模板文件 Template 里的书签，因为这是两个可以变化的因素。基础数据在变动后会更新时间戳，如果只需要生成此时间点以后的文档，此时基础数据的时间戳与基础参数里的时间戳可以成为数据选取的分界线。模板文件 Template 里的书签数量不定，不宜单一设置，应该合并设置，即全部的书签合并为一个比较长的字符串，相邻书签之间以特殊字符分隔。在使用时，以分隔符为界分离出每个书签，再逐一替换。鉴于基础参数数量较少的原因，可以将其存入应用配置文件，不需要另外开辟磁盘空间用于存储文本性质的配置文件。优点在于一则应用配置文件本质上也是文本文件，二则可以省去运行时文件读写的开销，三则可以减小程序的体积。

人机界面的主方法先使用大量代码通过 ConfigurationManager 提取应用配置文件里的各项基础参数，基本都转换为字符型，少量转换为整型和布尔型。其次使用业务逻辑层的文档类提取数据库的数据转换成 DataTable 格式，以循环方式每条数据逐一生成二维码。再将二维码与其他数据共同去替换模板文件 Template 里的书签。运行的过程中，有详细的运行步骤提示，可以告知用户每个文档的开始时间、文档的名称、成功与否等信息。当缺少这些提示时，虽然程序能正常运行，但用户没有直观感受，也不知道文档的生成情况，会降低用户评分。

当运行完毕后，根据数据库里数据生成的若干文档全部生成，然后存放于云端，由文档使用者自行下载使用。

4. 关键技术

4.1. 文档读写类

文档读写类中关于 Word 文档的操作手段是调用自身接口，针对模板文档 Template 里的不同类型书

签采用不同的替换方法，实现的步骤如下。

```
1 if(参数非法) return
4  创建 Word.Application 实例 word_app
5  创建 Word.Document 实例 word_doc
6 try
7 {
8       word_doc=word_app 打开 template
9if(bookmarks!=null)
10     for(i=0 to bookmarks.length-1)
11       if(word_doc 存在(bookmarks[i]))
12             word_doc.get_item(bookmark).文本=数据行[bookmark]
13     if(image_bookmarks!=null&&images!=null)
14for(i=0 to image_bookmarks.length-1)
15           if(word_doc 存在(image_bookmarks[i]&&exist image[i])
16           {
17 选中 word_doc.get_item(image_bookmarks[i])
18              shape=Word.InlineShape 加入 image[i]
19              调整 shape’s height or width
20           }
21 }
22 catch {return false}
23  存为 PDF
24 finally
25 {
26   关闭 word_doc
27   退出 word_app
28 }
```

4.2. 应用配置文件

关于项目的各个配置都存储在应用配置文件里，采用 XML 格式存储的键/值对。提高鲁棒性和灵活性的配置是书签的键/值对，其格式如下。

key="image_bookmarks" value="image1,image2"

key="bookmarks"
value="name,identity,gender,cand,type,number1,serial1,site1,depart,ticket,site2,number2,serial2"

从中可以看出，value 的值与书签命名一一匹配即可，顺序不影响结果。各个值之间的分隔符可以是任意字符，当然经验建议使用常见字符。

5. 应用案例

项目所在单位每年固定举办大型对外性质的招考会，规模都在 2000~3000 人。选取其中巅峰的 3206 人，在多个机器上运行测试得出以下结果，见表 1。其中模板文件 Template 的体积为 32 KB，照片的体积在 5~12 KB。因为二维码的非必要性，此次测试未在目标文档中加入二维码。

Table 1. Experiment environments and results
表 1. 测试环境及结果

序号	环境	机器 1	机器 2	机器 3
1	虚拟机	VMware ESXi 6.0	VMware ESXi 6.0	VMware Workstation Pro 15.5.6
2	CPU 类型	Intel(R) Xeon(R) CPU E5-2609 v2 @ 2.50GHz	Intel(R) Xeon(R) CPU E7-8880 v4 @ 2.20GHz	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
3	CPU 数量	2	4	1
4	内存	2 G	2 G	2 G
5	操作系统	Windows Server 2003 SP2	Windows Server 2003 SP2	Windows 7 SP1
6	耗时	01:21:50	01:05:01	00:58:56
7	单个文档体积	137~146 KB	137~146 KB	138~147 KB
8	文档总体积	441 MB	441 MB	443 MB

从表 1 的 CPU 型号可以看出，机器 1 和机器 2 为两台服务器，机器 3 为一台 PC 机。机器 1 和机器 2 测试的操作系统都运行在虚拟机里。机器 3 为追求测试环境的相似性，故在宿主操作系统上运行虚拟机，在虚拟机里再次安装操作系统。三台虚拟机操作系统都是分配 2G 内存，也是为做到测试环境的相似性。

从运行结果来看，三台机器的单个目标文档体积都在 140 KB 左右，只是机器 3 的单个目标文档最小值和最大值都相应比机器 1、机器 2 大 1KB。这种微量体积差异完全处在可接受范围。全部文档的总体积在 440 MB 左右，相对目前的低廉海量存储，不需要特别寻找大量存储空间。并且三个测试结果的总体积差异也只在 2 MB 左右，也是完全自可接受范围。

三次测试的运行耗时是本项目的重点。机器 1 挂载 SCSI 接口硬盘，机器挂载存储服务器，机器 3 挂载的则是普通 SATA 接口硬盘。机器 2 的 CPU 速度和硬盘速度都是三者之优，从理论上讲，应该是机器 2 的运行耗时最短，然而实际情况是机器 3 胜出。从这里可以看出，数量众多的小文件在短时间内频繁打开与关闭，是此项目运行中的瓶颈，运行耗时都由磁盘的 I/O 来决定，磁盘本身转速的高低不是主要因素。

此外，机器 1 和机器 2 上还另外有多个其他功能的虚拟机在同时运行，会有其他程序在同时读写磁盘，抢夺资源。机器 3 在测试过程中刻意关闭宿主操作系统里的其他程序，基本保证磁盘由虚拟机独家使用。这也是机器 3 在耗时方面最少的重要原因。

从整体来看，三台机器都在 2 小时内完成任务，这对人工合成目标文档来说，根本就是不可能任务。

表 1 体现出来的结果是三次测试环境不同，其实在多年的实际运作当中，模板文档 Template 经历过多次的变更。有标识字符数量的增减，位置的变动，也出现过模板文档 Template 完全更换的情况。因为采用代码与配置分离的方式，故代码可以适应各个版本的模板文档 Template，而不需要针对每个版本开发定制的程序。

6. 平行类比

6.1. 开发效率

文献[6]采用 COM 技术调用 Office 对象生成数据报告，文献[7]使用第三方引擎生成文档。此两种方法因为要在大量的类库与引擎中寻找与生成文档匹配的类型，导致前期准备工作过多，耗时过长。同时

在开发的过程中，需要技术员自行编写大量代码，自行管理各个对象才能满足要求。

本文方法通过调用 Word 自带的丰富接口，在较短的时间内即可完成程序的编写。在编写的过程中，因为有接口的编程支持，技术员可较少管理文档里的各个对象，把精力投入到整体设计与其他细节中去，从而大大降低开发难度。

6.2. 文档格式

文献里的解决方案都只提供 doc 或 docx 格式的文档。本文的解决方案在此基础之上，还提供 pdf 等通用文档格式，同时还有 txt 格式的日志记录，以备后期查询纠错。

7. 总结

本项目调用 MWOL，通过编程的方式单个处理办公文档模板 Template，再通过循环的方式可以批量生成目标文档。模板文档 Template 标识出其中的替换内容作为书签，并把关于书签的配置从代码中分离出来。如此的优势在于，程序有很强的鲁棒性，可以适应模板文档 Template 内容的变化。当模板文档 Template 的书签数量及名称变化时，不需重新编写代码，只需在配置文件里修改与书签对应的字符串即可，即使更换模板文档 Template 也是如此处理。那么模板文档 Template 可以由非技术人员设计撰写，技术人员则从中得以解脱，可以专注于项目里的技术部分。

项目严格按照分层思想设计，降低各层之间的耦合度，每层都可以适应项目的变化。当项目的需求变动时，只需要变动上层设计与中层设计，几乎不用变动下层设计。甚至在另外一个项目中，下层的设计也有很强的适应性，因为全是数据库的读写，以参数形式传入，就不涉及具体的数据库或者具体的配置。

参考文献

[1] 王怀民, 毛晓光, 丁搏, 沈洁, 罗磊, 任怡. 系统软件新洞察[J]. 软件学报, 2019, 30(1): 22-32.

[2] Silva, R.F.G., Roy, C.K., Rahman, M.M., Schneider, K.A., Paixao, K. and de Almeida Maia, M. (2019) Recommending Comprehensive Solutions for Programming Tasks by Mining Crowd Knowledge. 2019 *IEEE/ACM* 27*th International Conference on Program Comprehension* (*ICPC*), Montreal, 25-26 May 2019, 358-368. https://doi.org/10.1109/ICPC.2019.00054

[3] 马晓星, 刘觳哲, 谢冰, 余萍, 张天, 卜磊, 李宣东. 软件开发方法发展回顾与展望[J]. 软件学报, 2019, 30(1): 3-21.

[4] LeClak, A., Jiang, S. and McMillan, C. (2019) A Neural Model for Generating Natural Language Summaries of Program Subroutines. 2019 *IEEE/ACM* 41*st International Conference on Software Engineering* (*ICSE*), Montreal, 25-31 May 2019, 795-806. https://doi.org/10.1109/ICSE.2019.00087

[5] Tufano, M., Pantiuchina, J., Watson, C., Bavota, G. and Poshyvanyk, D. (2019) On Learning Meaningful Code Changes via Neural Machine Translation. 2019 *IEEE/ACM* 41*st International Conference on Software Engineering* (*ICSE*), Montreal, 25-31 May 2019, 25-36. https://doi.org/10.1109/ICSE.2019.00021

[6] 孙海民, 姜学东, 计大杰, 等. 自动化技术在生成数据报告中的应用[J]. 实验室研究与探索, 2017, 36(12): 136-142.

[7] 骆蓉, 黄俊, 黎茂锋, 等. 基于 Word 模板的复杂文档快速生成方法[J]. 计算机应用与软件, 2020, 37(10): 57-63.