Sheeter

以go (https://go.dev/dl/)做成的excel轉換工具
將指定格式的excel轉換為json資料檔案, 以及產生用於讀取資料檔案所需的cs/go程式碼
前身是sheet (https://github.com/yinweli/Sheet)

系統需求

- go (https://go.dev/dl/)1.19以上

安裝說明

- 安裝go (https://go.dev/dl/)
- 安裝sheeter (https://github.com/yinweli/sheeter), 在終端執行以下命令

```
go install github.com/yinweli/Sheeter/v2/cmd/sheeter@latest
```

- 如果想要安裝舊版的sheeter (https://github.com/yinweli/sheeter), 則可以在終端執行以下命令

```
go install github.com/yinweli/Sheeter/cmd/sheeter@latest
```

- 請注意! v2與v1的excel格式不相容, 彼此間無法讀取對方的excel檔案

如何使用

- 建立資料表單

- 建立設定檔案
- 在終端執行建置命令

```
sheeter build --config 設定檔案
```

- 執行後會產生以下檔案
    - 用於操作表格的程式碼檔案(.cs或是.go)
    - 符合json格式的文字檔案(.json)

範例

範例在專案中的`support/example`目錄下, 範例目錄說明如下

目錄/檔案	說明
verify.xlsx	範例的excel檔案
verify.yaml	範例的設定檔案
build.bat	執行轉換的批次檔
example.cs	cs版本的表格操作程式碼範例
example.go	go版本的表格操作程式碼範例
codeCs	產生的cs版本程式碼
codeGo	產生的go版本程式碼
json	產生的資料檔案

命令說明

以下描述了sheeter (https://github.com/yinweli/sheeter)提供的命令與旗標

build命令

用於建置資料檔案與程式碼

```
sheeter build [flags]
```

例如

```
sheeter build --config setting.yaml
sheeter build --config setting.yaml --lineOfField 1 --lineOfData 2
```

請注意目前sheeter (https://github.com/yinweli/sheeter)最多只能開啟999999個excel檔案, 每個excel檔案只能容納999999個表單
(不過在開啟這麼多檔案之前, 記憶體就用光了吧)

設定檔

```
source:                                 # 輸入列表
  - path1                               # 轉換path1目錄底下符合規格的excel檔案
  - path2                               # 轉換path2目錄底下符合規格的excel檔案
  - path/excel.xlsx                     # 轉換指定的excel檔案內符合規格的表單
merge:                                  # 合併列表
  - merge1$excel1#sheet&excel2#sheet # 將excel1#sheet, excel2#sheet合併為
merge1
  - merge2$excel3#sheet&excel4#sheet
exclude:                                # 排除列表
  - excel2.xlsx#sheet                   # 排除列表填寫excel的檔名以及sheet名稱
output: path/output                     # 輸出路徑
tag:          cs                        # 標籤字串
autoKey:      true                      # 自動選取索引
lineOfTag:    5                         # 標籤行號(1為起始行)
lineOfName:   1                         # 名稱行號(1為起始行)
lineOfNote:   2                         # 註解行號(1為起始行)
lineOfField:  3                         # 欄位行號(1為起始行)
lineOfData:   6                         # 資料行號(1為起始行)
```

命令旗標

旗標	參數	說明
--config	路徑與檔名; 例如: path/seeting.yaml	設定檔案路徑
--source	路徑, 檔案名稱, 路徑/檔案名稱...	輸入列表
--merge	name$excel.xlsx#sheet&excel.xlsx#sheet...	合併列表
--exclude	excel.xlsx#sheet, excel.xlsx#sheet...	排除列表
--output	路徑	輸出路徑
--tag	標籤列表	指定那些標籤的欄位要輸出
--autoKey	true/false	是否啟用自動選取索引

--lineOfTag	行號(1為起始行)	標籤行號
--lineOfName	行號(1為起始行)	名稱行號
--lineOfNote	行號(1為起始行)	註解行號
--lineOfField	行號(1為起始行)	欄位行號
--lineOfData	行號(1為起始行)	資料行號

--config

從設定檔讀取參數, 設定檔中的參數都可以被其他的旗標值替代

```
sheeter build --config setting.yaml --lineOfName 5
```

像是這種情況, 設定檔中的`lineOfName`的值就會被`--lineOfName`的值替代

--source

輸入列表, 可用以下幾種格式組合, 每個項目以 , 分隔
程式只會讀取副檔名為xlsx的檔案, 以及路徑中需使用 / 而非 \

- 路徑名稱
  path, path/, path/path...
- 檔案名稱
  example.xlsx, path/example.xlsx...

--merge

合併列表, 指定哪幾個表格可以額外合併成一個新的表格讀取器
指定的表格必須存在於輸入列表中, 其表格格式必須完全一致, 並且新的表格名稱不能與原本的的表格名稱重複, 否則輸出的程式碼會有錯誤
要注意的是, 合併列表不會取代原本個別的表格資料, 而是另外建立新的表格讀取器
每個合併項目可以寫成如下的模式

- name1$excel1#sheet&excel2#sheet
  將建立一個新的表格讀取器, 名稱為name1, 其中包含excel1#sheet與excel2#sheet的內容
- name2$excel1.xlsx#sheet&excel2.xlsx#sheet
  將建立一個新的表格讀取器, 名稱為name2, 其中包含excel1#sheet與excel2#sheet的內容

--exclude

排除列表, 在此列表中的表單不會被輸出, 每個項目以 , 分隔
排除名稱以excel檔名(不能有路徑以及副檔名), 跟sheet名稱用 # 組合
例如: excel#sheet, item#data, magic#cost

--output

輸出路徑, 決定產生的檔案要輸出到哪邊去

--tag

標籤字串, 用於控制那些欄位要輸出, 參考標籤行

--autoKey

是否啟用自動選取索引

- 未啟用
  將以表格中的 `pkey/lkey/skey` 欄位作為主要索引
- 啟用
  將以表格中第2欄作為主要索引, 但是如果第2欄的類型不是 `int/long/string` 的情況下會回報錯誤

version命令

用於顯示版本資訊

```
sheeter version
```

help命令

用於顯示命令說明

```
sheeter help [command]
```

資料表單說明

檔案名稱

檔案名稱必須符合以下規則

- 名稱中若是包含 `ignore` (不分大小寫)的檔案會被忽略
- 不是空字串
- 不能以數字開頭
- 只能使用以下符號: 英文字母, 數字, 空格, 底線

表單名稱

表單名稱必須符合以下規則

- 名稱中若是包含 `ignore` (不分大小寫)的表單會被忽略
- 不是空字串
- 只能使用以下符號: 英文字母, 數字, 空格, 底線

產生的檔案名稱規則

- excel名稱: 首字大寫(cs) 或是 首字小寫(go), 移除底線或空格, 底線或空格後首字必為大寫
- sheet名稱: 首字必為大寫, 移除底線或空格, 底線或空格後首字必為大寫
- 若excel名稱與sheet名稱相同, 則只留下excel名稱

標籤行

標籤行用來控制該欄位的資料是否要輸出到資料檔案與程式碼中
當設定檔中的標籤字串與欄位的標籤符合時, 該欄位就會被輸出
欄位若是沒有設定標籤, 則不會輸出到資料檔案
欄位的標籤若是包含 `ignore` (不分大小寫), 則不會輸出到資料檔案與程式碼
標籤為空格表示表格結尾

標籤字串	欄位標籤	是否輸出	原因
CS	C	是	C符合
CS	S	是	S符合
CS	X	否	無符合
A	AB	是	A符合
B	AB	是	B符合
X	AB	否	無符合
ABC	ADG	是	A符合
BEH	ADG	否	無符合

名稱行

欄位名稱, 只能是英文與數字與_的組合, 並且不能以數字開頭, 也不允許空白

註解行

欄位註解, 若為空格就輸出空註解, 若是有換行符號將會被移除

欄位行

欄位類型設置, 以下是可用的類型

類型	說明
pkey	表格主要索引, 使用32位元整數類型, 索引不可重複
lkey	表格主要索引, 使用64位元整數類型, 索引不可重複
skey	表格主要索引, 使用字串類型, 索引不可重複
bool	布林值
boolArray, []bool, bool[]	以逗號分隔的布林值陣列
int	32位元整數
intArray, []int, int[]	以逗號分隔的32位元整數陣列

long	64位元整數
longArray, []long, long[]	以逗號分隔的64位元整數陣列
float	32位元浮點數
floatArray, []float, float[]	以逗號分隔的32位元浮點數陣列
double	64位元浮點數
doubleArray, []double, double[]	以逗號分隔的64位元浮點數陣列
string	字串
stringArray, []string, string[]	以逗號分隔的字串陣列

資料行

依照類型填寫相應的內容即可, 空表格(也就是沒有任何資料行)是允許的, 遇到的第一個空行會被視為表格結束

輸出欄

表格中的第一欄被用來決定該行是否要輸出
當資料行的第一欄為 `ignore` (不分大小寫), 該資料行不會輸出到資料檔案中

其他的限制

- 表格設置
    - 表格必須有標籤行, 名稱行, 註解行, 欄位行, 但是可以不需要有資料行
    - 標籤行, 名稱行, 註解行, 欄位行必須在資料行之前
    - 設定檔中必須設定好標籤行, 名稱行, 註解行, 欄位行, 資料行的位置
    - 設定檔中行數是從1開始的
- 主索引
    - 表格必須有 `pkey/lkey/skey` 欄位
    - 表格只能有一個 `pkey/lkey/skey` 欄位
    - `pkey/lkey/skey` 欄位中的內容不能重複

產生目錄

名稱	說明

codeCs	存放產生的cs程式碼
codeGo	存放產生的go程式碼
json	存放資料檔案

格式化程式碼

sheeter (https://github.com/yinweli/sheeter)並不負責幫產生的檔案排版, 如果需要排版, 就需要自己寫.bat/.sh來執行
以下介紹cs, go的排版工具, 有需要可以自己去安裝

csharpier

用於cs的排版工具

- 安裝
  - 安裝dotnet (https://learn.microsoft.com/zh-tw/dotnet/core/sdk), 如果有安裝.net sdk, 或有安裝unity可能可以省略此步驟
  - 安裝csharpier (https://github.com/belav/csharpier), 在終端執行以下命令

```
dotnet tool install csharpier -g
```

- 使用
  - 在終端執行以下命令

```
dotnet csharpier .
```

gofmt

用於go的排版工具

- 安裝
  - 安裝go (https://go.dev/dl/)時會順便安裝
- 使用
  - 在終端執行以下命令

```
gofmt -w .
```

專案目錄說明

目錄	說明
cmd/sheeter	sheeter命令程式
cmd/sheeter/build	建置表格命令
cmd/sheeter/version	顯示版本命令
sheeter	sheeter命令程式用到的各項組件
sheeter/builds	表格轉換(用於build命令)
sheeter/excels	表格組件
sheeter/fields	欄位組件
sheeter/layouts	布局組件
sheeter/nameds	命名工具
sheeter/pipelines	管線組件
sheeter/tmpls	模板組件
sheeter/utils	協助組件
support	支援專案
support/example	範例資料
support/handmade	手製模板, 用來檢查模板是否有錯誤
support/verify	驗證專案
testdata	測試資料

Task命令說明

Task是一個運行/構建task的工具, 可以到task (https://taskfile.dev/)查看更多資訊; 可在命令列執行以下命令

- `task lint`: 進行程式碼檢查
- `task test`: 進行程式碼測試
- `task bench`: 進行效能測試

JetBrains licenses

sheeter (https://github.com/yinweli/sheeter)使用了JetBrains的Goland的免費開發許可, 在此表示感謝