table-data-format-transform-app TUTORIAL

Main Contributor	Contact E-mail	Renew Date	GitHub
JamesHanZhang	`jameshanzhang@foxmail.com`	2023-11-08	LINK

项目说明

本程序需要和参数表搭配执行, 由参数表确定基本参数, 由互动界面确定常需修改的参数
基于上述参数, 本程序可以实现如下功能:

相同表结构的数据源的转换处理

1. 通过`if_circular=True`可无视大小读取excel, csv, md文档为数据源;
2. 数据源的读取过程中, 会自动将脏数据抽离出来形成`_error_lines`文件保存在导入路径下, 并仅导入可正确读取的记录;
3. csv导入过程中, 支持多字符作为分隔符, 会在导入路径下自动生成`_repl_sep.csv`为后缀名的单字符分隔符的文件, 在进行读取;
4. 可将数据源转换成excel, csv, md文档;
5. 可将数据源转换成`sql`建表语句(自动爬取数据结构)及插入语句; 其中oracle插入语句穿插`commit;`命令不对系统造成负担;
6. sql建表及插入语句的创建支持5种数据库, 分别是: Oracle, MySql, PostgreSql, SqlServer, GBase;
7. 可以文件夹的形式批量读取文件夹下所有的excel, csv, md(包含子文件夹下的数据)为数据源, 并可指定单一数据类型进行读取;
8. 仅支持同数据结构的文件(可以不同数据格式, 例如同结构的xlsx, csv文件)批量读取, 批量读取的文件最终会统合形成单一文件, 或通过`if_sep=True`参数拆分成多个文件;
9. 可基于chunksize的大小设定将数据源拆分成多个记录数上限为chunksize的小文件;
10. 支持基本的数据处理方法: 修改列名, 修改数据类型, 选择特定列输出, 数据脱敏;
11. CSV数据导入过程中, 会自动筛选出脏数据并保存为后缀名为`_error_lines.csv`的数据; 支持多分隔符导入;
12. 导入导出皆会保存数据记录条数到`.json`参数表内; 导出SQL的时候会将各列最长的长度记录到表结构中;

不同表结构的不同数据源的处理

1. 如果转换参数一致, 可使用参数表复用模式, 转换过程复用同一个参数表, 导入的文件名会自动转为导出的文件名及表名;
2. 如果转换参数不一致, 可采用批量新建参数表模式,先批量新建参数表, 再执行新建成功的复数参数表, 实现不同文件的转换;

主程序说明

- `file_trans_on_extends_app`：主要用于快捷转换单文件数据格式，支持excel, csv, markdown；直接使用即可，根据提示输入参数运行；
- `table_data_format_transform_app`: 项目所有完整功能开放的应用，需调整参数后再执行，根据提示执行即可；
  - `import_params_setting.py`: 调整导入参数，选择数据导入方式；
  - `basic_process_params_setting.py`: 调整过程参数，选择激活的基本处理过程功能；
  - `output_params_setting.py`: 调整导出参数，选择数据导出方式；

附属程序 - file_trans_on_extends_app使用说明

中文名: **根据拓展名快速格式转换APP**

1. 登录程序，根据提示数据导入导出文件的名称（注意，必须包含拓展名，例如`test.xlsx`）；
2. 并输入一些其他的参数，根据拓展名会自动导出相应的文件；
3. 仅支持excel, csv, markdown之间的或许转换；
4. 支持大数据单文件转换；csv支持多字符分隔符；
5. 会自动校验csv, markdown的导入数据，自动识别脏数据；
6. 支持将大文件数据拆分成多个小文件数据；

主程序 - table_data_format_transform_app使用说明

中文名: **多功能数据格式转换APP**

1. 第一次执行的时候，必须选择参数覆盖模式，可以半路关闭，仅自动为程序搭建路径下的各个文件路径；
2. 一般需要先调整好`_params_setting.py`文件内的参数，再打开文件执行，因参数过多，仅支持根据参数表执行；
3. 第一次执行之后，也可以采用通过修改resources里已经有的基本参数表`BASE_PARAMS_SET`来确定新的参数表的模式，就不需要必须在执行程序前就调整好参数了；`BASE_PARAMS_SET`的参数含义和`_params_setting.py`文件内的说明一致；
4. 具备附属程序的所有功能；

执行程序的4种基本模式

1. <**表结构一致模式**> 单参数表处理模式, **常用模式**, 数据源的表结构一致时采用该模式处理数据转换;
2. <**结构不一致参数一致模式**> 不同表结构数据源转换, 批量模式, 将文件夹下的文件基于同一个参数表分别进行转换; 导入的文件名主体默认为导出的文件名主体以及导出的SQL表名;
3. <**皆不一致模式**> 不同表结构数据源采用不同的参数表, 需首先挨个搭建各个数据源对应的参数表, 再批量执行; 搭建参数表基于`BASE_PARAMS_SET.json`表进行修改, 其各个属性的含义与`_params_setting.py`一致;
4. <**多参数表执行模式**> 输入已存在的不同的参数表名, 直接执行; 如该参数表不存在则报错;

选择模式进入后，过程数据处理是固定采用`activation`激活模式的，但是导入参数的模式，以及导出数据的模式有不同：

导入参数的模式

1. 通过`_params_setting.py`来设定参数, 会自动覆盖同名参数表;
    - 第一次执行程序的时候需要选择这个模式，且必须在程序未启动时调整好参数；其实是开发者模式；
2. 通过已经存在的`resources`里的参数文件来确定参数, 如该参数文件不存在则报错;
    - 方便已经转换过一次的数据重新转换的时候不需要再调整参数；
3. 通过修改`resources`里已经有的基本参数表`BASE_PARAMS_SET`来确定新的参数表;
    - 选择这个模式，允许启动程序后再调整参数；

导出数据的模式

1. <根据激活导出模式>: 采用'output_params_setting.py'中的激活功能判断是否导出对应格式的数据;
    - 该模式激活几个功能就导出几个文件(激活拆分功能则更多);
    - 该模式在导出`.sql`文件时, 采用参数'table_name'确定导出的表名及文件名;
2. <根据拓展名导出模式>采用导出文件output_file的拓展名(例如'test.xlsx')判断是导出哪种格式数据;
    - 模式一次仅激活一个导出功能;
    - 该模式在导出`.sql`文件时, 默认将临时表的表名设置为导出文件的主体名(去掉拓展名);
3. <自动模式>, 根据导出的文件名'output_file'是否有拓展名(例如'test.xlsx')来判断采取以上两种模式的哪种模式;
    - 如导出的文件名没有拓展名, 即只有纯粹的文件名(例如'test'), 则激活<根据激活导出模式>;
    - 如导出的文件名有拓展名(例如'test.xlsx'), 则激活<根据拓展名导出模式>;

在选择对应的模式后，即可根据提示输入不同的参数，主要还是根据参数表的内置参数执行程序；

主程序参数说明

- `import_params_setting.py`: 调整导入参数，选择数据导入方式；
- `basic_process_params_setting.py`: 调整过程参数，选择激活的基本处理过程功能；
- `output_params_setting.py`: 调整导出参数，选择数据导出方式；

三个参数文件内部都有文字提示，在此不予赘叙，仅就核心参数做进一步说明:

`import_params_setting.py`参数说明

- `batch_import_params`: 同结构批量文件导入功能
    - `if_batch`: 判断是否开启批量导入，多文件会合并为一个文件，也可以重新进行拆分；
    - `import_type`: 根据拓展名，例如`.xlsx`判断批量筛选某一类型的批量导入文件；
- `if_circular`: 是否循环读取数据，循环读取模式是专门用来应对大数据的，可以小批量多迭代的方式逐步导入数据；
- `chunksize`： 循环导入时每次读取的记录条数，以及`if_sep=True`时进行拆分的数据条数；

`basic_process_params_setting.py`参数说明

- 通过`activation`参数判断是否激活该功能；
- `change_names_previously`：在程序开启前修改列名；
- `change_names_finally`：在导出的最后修改列名；
- `change_types_opt`：修改列类型，需要采用`pandas`所用的列类型，可以参考默认参数的提示；
    - 提示位置：`analysis_modules/default_properties/func_tuner_properties.py:63`
- `pick_columns_opt`：通过列表来仅选择特定的列导出；
- `data_masking_opt`：数据脱敏功能；
- `basic_processing_order`用来控制各个功能的执行顺序；

`output_params_setting.py`参数说明

- `overwrite`：决定了导出的文件是覆盖新建还是追加到同名文件的末尾；
- `if_sep`：决定了是否按照导入参数的`chunksize`的记录条数来将数据拆分成多个子文件；常用来保证太大的数据源能拆分成excel能承载的极限的数据条数；
- `only_one_chunk`：拆分的时候，是否需要仅拆分出第一个子文件数据；通常用来处理数据很大的时候，仅想正常观看数据结构的时候使用；