README.md 2024-12-12

README

安装: 使用python uv 安装所有依赖

```
uv sync
```

配置:

1. setting.toml 文件: REWORKSPACE 这个配置到本地目录 2. 码运行主要在: test_standard_process.py 这个文件

代码运行步骤:

1. test_download_all_bonds 这个函数用来下载pdf文件 修改年份范围和省份范围,可以指定某个省某个年份范围下载PDF,下载完成之后的文件目录结构: {REWORKSPACE}/<省份>/<年份>

```
def test_download_all_bonds():
   for year in range(2024, 2014, -1):
       # provinces = get_province_list(["广东省", "黑龙江省","山东省","福建省","辽宁省","云南省","吉林省","四川
      # DONE PROVINCE =_[""]
      ##一'西藏自治区"一海南省"、一青海省"、一宁夏回族自治区"、二江苏省"一院西省
      # BBNE provinces = ["吉林省" ["云南省" ""四川省" "肉蒙古自治区"]
      #=provinces== ["宁夏回族自治区"_"安徽省", "山东省" _"山西省"]
      #50 PD PATTORES -- PLE 的复数自治区产 -- "无 苏治" -- "红 西治" -- "可以治" --
      #-----
      # provinces = ["河南省", "浙江省", "海南省", "湖北省"]
      # provinces = ["湖南省", "甘肃省", "福建省", "西藏自治区"]
      # provinces = ["贵州省", "辽宁省", "陕西省", "青海省"]
      provinces = ["陕西省", "安徽省"]
      ## 河南省/山东省/福建省
      for province in provinces:
          set_parameters(str(year), province)
          pd_ext.filter_table_file(Path(REWORKSPACE + "/overall.csv"), col_filters,
                               REWORKSPACE + "/" + filtered_file_name)
          os.makedirs(download_dir, exist_ok=True)
          download_bonds(source_file_path, download_dir)
```

2test_unzip_all_files 这个用来unzip下载的所有zip包里面的文件到指定目录all_files目录,修改省份和年份就可以

README.md 2024-12-12

```
def test_import_project_detail_data():
   1. manual check and collect first raw data
   2. cleanup data for csv project
    :return:
   set_parameters( year: "2015", province: "宁夏回族自治区")
   cleanup_source_file_path = cleanuped_file
   cleaned_df = pd_ext.read_to_df(cleanup_source_file_path)
   models = pd_ext.read_to_models_by_field_mapping(cleaned_df, model_class=BondProjectDetail
                                                    , field_mappings=FIELD_MAPPINGS)
   for model in models:
       trv:
           value = csv_ext.replace_dots_except_last(model.amount)
           # if float(value)>1000:
           model.amount = float(value) / 10000
        except Exception as e:
           model.amount = model.amount
       db.save(model)
```

3test_import_project_detail_data 用来将csv数据数据清洗,结构化,导入数据句,修改年份和省份,从 **<年份>-<省份名>.csv** 文件中读取数据,清洗结果的文件是 *cleanup-<年份>-<省份名>.csv*

```
def test_import_project_detail_data():
   1. manual check and collect first raw data
   2. cleanup data for csv project
   :return:
   set_parameters( year: "2015", province: "宁夏回族自治区")
   cleanup_source_file_path = cleanuped_file
   cleaned_df = pd_ext.read_to_df(cleanup_source_file_path)
   models = pd_ext.read_to_models_by_field_mapping(cleaned_df, model_class=BondProjectDetail
                                                    , field_mappings=FIELD_MAPPINGS)
   for model in models:
       try:
           value = csv_ext.replace_dots_except_last(model.amount)
           # if float(value)>1000:
           model.amount = float(value) / 10000
       except Exception as e:
           model.amount = model.amount
       db.save(model)

   至 2023-四川省.csv

2023-四川省.xlsx

   □ cleanup-2020-四川省.csv
```

4.```test_cleanup_import`` 直接从清洗过的文件中导入数据库,使用基本同第三步骤,主要发现有些原始文件保存数据库出错,然后修改一下清洗文件之后,直接导入

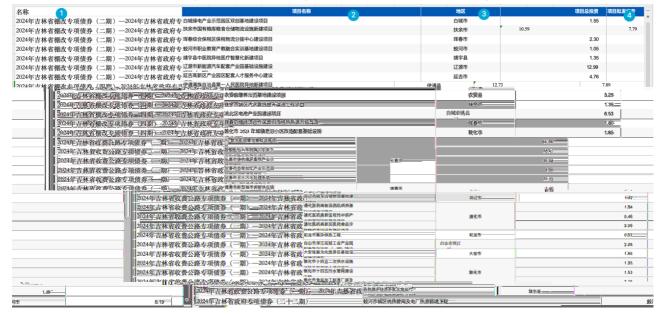
数据从哪里来

pdf文件下载完成之后,

1. 通过WPS 的 PDF转换成EXCEL功能 批量转换PDF成EXCEL文件

README.md 2024-12-12

- 2. 从EXCEL文件中获取到相关的债券 息表格,复制到<年份>-<省份名>.csv 中间中
- 3. 复制过程中尽可能的对齐数据,有偏差问题不大,但是取的值的列数需要对上,比如



1, 2, 3, 4是需要的数据,每行不对齐没有问题,但是每行有数据的1, 2, 3, 4顺序的值需要是正确对应的,中间有多少空行是可以在清洗时候处理的 具体就是每行的:第1有数 的列 表 债券名称,第2个有值的列 表 项目名称,第4个有值的列 表 发行金额,1和2, 3和4中间有空列,每行都有不一样数量的空列都问题不大可以清洗