libutils

general utilities library for all `xl2roefact` components and modules.

Identification:

- code-name: `libutils`
- copyright: (c) 2023, 2024 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Components:

- `complete_sexe_file() -> bool`: Rename and move resulted exe file (called from `build_sexe` script)
- `dict_sum_by_key(dict, str) -> float`: Sum a dictionary for a given key at all depth levels
- `find_str_in_list(list, list) -> int`: Search more strings (ie, a list) in list of strings
- `hier_get_data_file(file_name: str) -> Path`: Get `Path(file_name)` from hierarchy of locations
- `invoice_taxes_summary(list[dict]) -> dict`: Calculates invoice taxes summary as required by ROefact requirements
- `isnumber(str) -> bool`: Test a string if it could be used as number (int or float)

hier_get_data_file

```
def hier_get_data_file(file_name: str) -> Path | None
```

Get `Path(file_name)` from hierarchy of locations: (1) current directory, (2) package `data/` directory, (3) `None` is file does not exists in 1 or 2 locations.

Arguments:

- `file_name` - the name of the file to be returned as full path

Returns:

- `Path` - path of file if was found in (1) or (2) locations or `None` if not found

complete_sexe_file

```
def complete_sexe_file(drop_source: bool = True) -> bool
```

Rename and move resulted exe file. This function is dedicated only to development phase, so various objects are hard coded.

Specs:

- `file to process` .../dist_sexe/xl2roefact_to_update_name.exe `-->` .../dist/xl2roefact-version-win64.exe
- Note 1: all function code suppose that current directory is root of `xl2roefact`, ie where is located `pyproject.toml` of package

Arguments:

- `drop_source` - indicate to delete source file after copying, ie make a "move" operation, otherwise make a copy keeping the source file. Default behaviour is to delete source.

Returns:

- `bool` - True if file was found, renamed and moved with no error

invoice_taxes_summary

```
def invoice_taxes_summary(invoice_lines: list[dict]) -> list
```

Calculates invoice taxes summary as required by ROefact requirements.

Arguments:

- `invoice_lines` - section with item lines from 'big' invoice dictionary

Returns:

- `list` - usable for "cac_TaxSubtotal" key

dict_sum_by_key

```
def dict_sum_by_key(search_dict: dict | list[dict], sum_key: str) -> float
```

Sum all dictionary (or list off dictionaries) items, at all levels, for a given key.

Arguments:

- `search_dict` - dictionary to be searched for
- `sum_key` - key to be searched

Returns:

- `float` - with required sum

isnumber

```
def isnumber(a_string: str) -> bool
```

test if a string is valid as any kind of number.

Arguments:

- `a_string` - input string.

Returns:

- `True` - if input string is valid as any kind of number, orherwise `False`.

find_str_in_list

```
def find_str_in_list(list_of_str_to_find: list, list_to_search: list) -> int
```

find a substring from `list_of_str_to_find` in elements of `list_to_search`.

Arguments:

- `list_of_str_to_find` - list of strings to search for.
- `list_to_search` - liste where to search for substrings.

Returns:

- `index` - the index of list item which contains `str_to_find` (first found) or `None` if not found.

ldxml

ldxml: modul de incarcare a facturii in sistemul ANAF E-Factura

Identification:

- code-name: `ldxml`
- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Specifications:

- document cerinte initiale: `110-SRE-api_to_roefact_requirements.md` section `Componenta xl2roefact`
- INTRARI: fisier `f-XML`
- IESIRI: raport cu validarea si identificatorul incarcarii

sys_settings

System database and parameters.

This module acts as an "ORM" between `xl2roefact` system and different data objects. It contains:

- tinny physical data objects (Section 1.)

- logical data objects (Section 2.)
- interfaces to external data objects as files or other specialized systems (Section 2.)

Notes:

- "Sections 1, sl , ..." organization of code even is just a pure visual one, is recommended to be respected and followed it being intended to increase code readability and latter maintainability.
- IMPORTANT to keep in mind: This module IS NOT intended to be modified by end users or administrators. Only development stuff can alter this database because application code must be updated accordingly.
- for updaters remark: because dependencies, code sections should follow strict enumerated order in comments

References:

- copyright: (c) 2024 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

InvoiceTypes

Section 2. INTERFACES & LOGICAL data

rdinv

rdinv: modul de procesare a fisierului Excel ce contine factura si colectare a datelor aferente.

Formatul acceptat fisier Excel este `XLSX`.

Identification:

- code-name: `rdinv`
- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Specifications:

- document cerinte initiale: `110-SRE-api_to_roefact_requirements.md` section `Componenta xl2roefact`
- INTRARI: fisier format XLSX ce contine factura emisa (cod: `f-XLSX`)
- IESIRI: fisier format JSON imagine a datelor facturii (cod: `f-JSON`)

rdinv

```
def rdinv(file_to_process: str,
          invoice_worksheet_name: str = None,
          *,
          invoice_type_code: InvoiceTypesEnum = InvoiceTypesEnum.NORMALA,
          debug_info: list[str] = None,
          owner_datafile: Path = None) -> dict
```

read Excel file for invoice data.

Produce a dictionary structure + JSON file with all data regarding read invoice: canonical KV data, meta data, map to convert to XML and original Excel data.

Arguments:

- `file_to_process` - the invoice file (exact file with path).
- `invoice_worksheet_name` - the worksheet containing invoice, optional, defaults to first found worksheet.
- `invoice_type_code` - code of invoice type, for example "380" for regular.
- `debug_info` - list with *index[0]* containing all print messages issued by function. If None or nothing sent will print on `stdout`. List is required because a mutable object is needed to be able to write in.
- `owner_datafile` - specify a file to read supplier data from, default `None` meaning to read supplier data from Excel file.

Returns:

- `dict` - the invoice extracted information from Excel file as `dict(Invoice: dict, meta_info: dict, excel_original_data: dict)`

Notes:

- `db: pylightxl object`: EXCEL object with invoice (as a whole)
- `ws: pylightxl object`: WORKSHEET object with invoice

get_excel_data_at_label

```
def get_excel_data_at_label(pattern_to_search_for: list[str],
                            worksheet: xl.Database.ws,
                            area_to_scan: list[list[int]] = None,
                            targeted_type: Callable = str,
                            down_search_try: bool = True) -> dict
```

get "one key Excel values", like invoice number or invoice issue date.

Arguments:

- `pattern_to_search_for` - for example for inv number, will pass the `PATTERN_FOR_INVOICE_NUMBER_LABEL`.
- `worksheet` - the worksheet containing invoice (as object of `pyxllight` library).
- `area_to_scan` - area of cells to be searched, default whole worksheet.
- `targeted_type` - what type expect (will try to convert to, if cannot will return str), default `str`.
- `down_search_try` - establish if DOWN search method is tried, default `True`.

Returns:

`None` if not found OR `dictionary` containing: * `"value": int | float | str` - the value found covenrted to requested `targeted_type` if possible or `str` otherwise; if "out of space" then returns `None` * `"location": (row, col)` - adrees of cell where found value

Notes:

- normal scan order is 1.RIGHT, 2.DOWN (if allowed), 3.IN-LABEL only in given area and pattern.

mk_kv_invoice_items_area

```
def mk_kv_invoice_items_area(invoice_items_area_xl_format) -> dict
```

transform `invoice_items_area` in "canonical JSON format" (as kv pairs).

Arguments:

- `invoice_items_area_xl_format` - invoice items area in Excel format (ie, DataFrame with row, col, data).

Returns:

- `invoice_items_area_xl_format` - dictionary with invoice items in Excel format (ie, rows, columns).

Notes:

- for ROefact XML model (& plan) see `invoice_files/__model_test_factura_generat_anaf.xml`.

get_invoice_items_area

```
def get_invoice_items_area(worksheet, invoice_items_area_marker,
                           wks_name) -> dict
```

get invoice for `invoice_items_area`, process it and return its Excel format.

Process steps & notes:

- find invoice items subtable.

- clean invoice items subtable.
- extract relevenat data.
- NOTE: all Excel cell addresses are in `(row, col)` format (ie, Not Excel format like "A:26, C:42, ...")

Arguments:

- `worksheet` - the worksheet containing invoice (as object of `pyxllight` library).
- `invoice_items_area_marker` - string with exact marker of invoice items table.
- `NOTE` - this is the UPPER-LEFT corner and is determined before calling this procedure.
- `wks_name` - the wroksheet name (string) of the `worksheet` object.

Returns:

- `invoice_items_area` - dictionary with invoice items in Excel format (ie, rows, columns).

get_merged_cells_tobe_changed

```
def get_merged_cells_tobe_changed(file_to_scan,
                                  invoice_worksheet_name,
                                  keep_cells_of_items_ssd_marker=None) -> list
```

scan Excel file to detect all merged ranges.

Arguments:

- `file_to_scan` - the excel file to be scanned.
- `invoice_worksheet_name` - the worksheet to be scanned.
- `keep_cells_of_items_ssd_marker` - tuple with cells that will be marked IN ANY CASE to be preserved:
- use case: to keep all potential invoice items ssd rows.
- format: `tuple(row, col, val)` where row & col are relevant here
- default: `None`

Returns:

- `cells_to_be_changed` - list with cells that need to be chaged in format `(row,col)`.

Notes:

- function is intended to be used ONLY internal in this module.
- use `openpyxl` library to do its job.

build_meta_info_key

```
def build_meta_info_key(excel_file_to_process: str,
                        invoice_worksheet_name: str, ws_size: list,
                        keyword_for_items_table_marker: str,
                        found_cell: list) -> dict
```

build meta_info key to preserve processed Excel file meta information: start address, size.

Notes:

- (1.) all cell addresses are in format (row, col) and are absolute (ie, valid for whole Excel file).
- (2.) this function is designed to be used internally by current module (using outside it is not guaranteed for information 'quality').

Arguments:

- `excel_file_to_process` - name of file to process as would appear in `meta_info` key.
- `invoice_worksheet_name` - the worksheet name as would appear in `meta_info` key.
- `ws_size` - worksheet size as would appear in `meta_info` key (index 0 max rows, index 1 max columns).
- `keyword_for_items_table_marker` - the content of cell used as start of invoice items subtable as would appear in `meta_info`.
- `found_cell` - position of cell used as start of invoice items subtable as would appear in `meta_info` key (index 0 row, index 1 column).

Returns:

- `meta_info` - dictionary built with meta information to be incorpoarted in final invoice dict

get_partner_data

```
def get_partner_data(partner_type: str,
                     *,
                     wks,
                     param_invoice_header_area: dict,
                     supplier_datafile: Path = None) -> None
```

Get invoice partener data from Excel.

Notes:

- *for developers*: function works by generating side effects and must be located in `rdinv.py`
- *side effects*: this function works by directly modifying `param_invoice_header_area` sent parameter
- *supplier_datafile exception*: if file is not found or cannot be read, this function will force complete application termination (`sys.exit`)

Arguments:

- `partner_type` - one of "CUSTOMER", "SUPPLIER" or "OWNER" to specify for what kind of parner get data. The value "OWNER" is designed to get data from an outside database / file (master data)
- `wks` - current work-on `pylightxl Worksheet` object
- `param_invoice_header_area` - *mode IN-OUT*, outside `param_invoice_header_area` as used and needed in `rdinv()`. This function will write back in this variable
- `supplier_datafile` - for `partner_type = "CUSTOMER"` here is expected the file where to get supplier data

Returns:

- `None` - all data is produced directly in parameters as side effect

chkisld

chkisld: modul de verificare a starii de incarcare a unei facturi emise

Identification:

- code-name: `chkisld`
- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Specifications:

- document cerinte initiale: `110-SRE-api_to_roefact_requirements.md` section `Componenta xl2roefact`
- INTRARI: fisier `f-XLSX` sau numarul / cheia / codul facturii
- IESIRI: valoarea echivalent `TRUE` daca factura a fost deja incarcata sau valoare echivalent `FALSE` daca factura nu a fost incarcata

__init__

__version__

default conversion takes place over xl2roefact actual version

wrxml

wrxml: modul de generare a fisierului format XML

Identification:

- code-name: `wrxml`
- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Specifications:

- document cerinte initiale: `110-SRE-api_to_roefact_requirements.md` section `Componenta xl2roefact`
- INTRARI: fisier `f-JSON`
- IESIRI: fisier format XML conform cerintelor si sistemului ANAF E-Factura (cod: `f-XML`)

__main__

xl2roefact.main: Python package standard file to assure run as `python -m xl2roefact`.

Identification:

- code-name: `__main__`
- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Deployments:

- Windows: MSI installer with EXE application.
- Linux: `xl2roefact` executable shell as wrapper for `xl2roefact.py`.

Specifications:

- command general format: `python -m xl2roefact [OPTIONS] COMMAND [ARGS]...`.
- help: `python -m xl2roefact --help`.

config_settings

Configuration and setting parameters.

Regulile recomandate se gasessc in documentul (recommended rules are in document `xl2roefact/data/README_app_config_rules.md`)

Public objects:

- `rules_content`: contains the rules text (rendered)

Info:

- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

DEFAULT_DUE_DATE_DAYS

NOTE: "pattern-uri" (sabloane) de identificare si regasire a datelor folositi de

__ comanda `xl2json` reprezentind functionalitatea de extragere a datelor din Excel si exportul lor in formatul JSON (modulul `rdinv)__

---

rules_content

`rules_content` public variable to be use as "mini help" by `settings -r` command of application

load

```
def load()
```

Read and load settings from external data file.

commands

Layer 2 commands API implementation.

Objectives:

- create an environment where a xl2roefact can be run in *session or interactivelly mode*
- session parameters: persist commands run parameters in user profile (directory of `os.%userprofile%` or Linux `~/.profile`)
- group all layer 2 commands for:
  - `xl2roefactd` (aka server)
  - `xl2roefact-client` (aka console client)
  - `web2roefact` (aka web client UI front end)* components

Identification:

- code-name: `commands`
- Copyright: (c) 2024 RENware Software Systema
- Author: Petre Iordanescu (petre.iordanescu@gmail.com)

CommandResult Objects

```
@dataclass
class CommandResult()
```

Define the result of execution. This structure contains information collected in methods execution. After each method execution all "prints" and status information stated by method in its execution (ie, which was saved) will be contained in.

Fields: * `status_code`: `int` Status code as used by HTTP standard returned codes (200 for success) * `status_timestamp`: `str` Timestamp of information in UTC ISO format * `status_text`: `str` Short text of this result set. Normally used to display a brief message note associated to code (for example "404 Not found") * `result`: `Any` The effective result information returned by method as core result of execution. Depending on method, this is a *Python specific structure*, scalar, basic or complex one * `stdout_text`: `str` Collected console "prints" output in text format. Normally a standard print() of this value will reproduce the exact console output if method would be "raw executed" in development mode * `stdout_html`: `str` the same as `stdout_text` but in HTML format ready to be sent "as is" to a browser (its a COMPLETE and FULL HTML doc). Used if pages together with other elements it is recommended to isolate it with distinct `div` and `iframe` tags

SessionDataType Objects

```
@dataclass
class SessionDataType()
```

Define session data used in class `Commands`. These data objects mainly represents almost all parameters encountered in console application and that are suspect to be reusable when "chain more commands" iin the same session and is desirable to keep last values.

Also in web applications is normal to not ask for parameters already entered by an end user and to preserve last entered values at least as default ones.

Commands Objects

```
class Commands()
```

xl2roefact commands layer implementation. Descriere generala layer si componenta..

__init__

```
def __init__()
```

Init session data variables with default values.

session_data_set

```
def session_data_set(*,
                     file_name: str = ...,
                     invoice_type: InvoiceTypesEnum = ...,
                     files_directory: Path = ...,
                     owner_datafile: Path = ...,
                     verbose: bool = ...) -> bool
```

Set session data.

Rules: * session data is kept as class-instance variables. This will use for "interactive" or "web" editions * if a parameter is not sent at call, then it is left unchanged * any other sent value is saved as instance variable * elipsis as default parametrs values help to make difference between a sent parameter (even with None) and a not sent one

Arguments:

- `all_item` - more instances = all data items required to be kept as reusable session data

Returns:

- `bool` - any change was made

session_data_reset

```
def session_data_reset()
```

Resset session data to defaults.

version

```
@classmethod
def version(cls) -> str
```

return the version of `xl2roefact` used by this class

xl2json

```
def xl2json(invoice_type: InvoiceTypesEnum = ...,
            file_name: str = ...,
            files_directory: Path = ...,
            owner_datafile: Path = ...,
            verbose: bool = ...) -> bool
```

read excel invoice and generate a JSON file with invoice data, miscellaneous meta and original Excel found data

Arguments:

- `invoice_type_code` - invoice type (for example regular invoice or storno) as this info is not usually subject of Excel file. Default to `380` (regular / usual invoice)
- `file_name` - files to process (wildcards allowed).
- `files_directory` - directory to be used to look for Excel files. Defaults to `invoice_files/`. NOTE: if default directory does not exists will consider current directory instead
- `owner_datafile` - File to read invoice supplier (owner) data instead Excel.
- `verbose` - show detailed processing messages". Defaults to `False`.

Returns:

- `bool` - True if command executed without errors. If return in False, the kasr result should be inspected to see error status and text (method `results_stack_pop()`)

response_out

```
def response_out(*,
                 status_code=200,
                 status_text="undefined",
                 status_result="undefined")
```

Prepare and enque a response. This is designed to be in-class used and not for public interface. Arguments are min of what to be enqueued. Other ibformation are constructed local.

get_last_result

```
def get_last_result() -> dict[CommandResult]
```

Get last result dictionary from stack WITHOUT drooping it.

Returns:

- `CommandResult` - last result as dictionary

pop_session_results

```
def pop_session_results() -> list[CommandResult]
```

Get all session results as dictionary.

Returns:

- `CommandResult` - list with all session results as dictionary

get_var_name

```
@classmethod
def get_var_name(cls, var)
```

Return a variable defined in class as string of its name.

app_cli

app_cli: the command line application for all xl2roefact functionalities.

Identification:

- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

about

```
@app_cli.command()
def about()
```

Provide a short application description.

settings

```
@app_cli.command()
def settings(rules: Annotated[
    bool,
    typer.
    Option("--rules", "-r", help="show settings recommended update rules"),
] = False)
```

Display application configuration parameters and settings that are subject to be changed by user.

Arguments:

- `rules` - show recommended rules to follow when change application configurable settings (available in both RO & EN languages). Defaults to `False`.

xl2json

```
@app_cli.command()
def xl2json(
    invoice_type: InvoiceTypesEnum = InvoiceTypesEnum.NORMALA.value,
    file_name: Annotated[
        str, typer.Argument(
            help="files to process (wildcards allowed)")] = "*.xlsx",
    files_directory: Annotated[
        Path,
        typer.Option(
            "--files-directory",
            "-d",
            exists=False,
            file_okay=False,
            dir_okay=True,
            writable=True,
            readable=True,
            resolve_path=True,
            help=
            "directory to be used to look for Excel files (if default directory does not exists
will consider current directory instead)."
        ),
    ] = "invoice_files/",
    owner_datafile: Annotated[
        Path,
        typer.Option(
            "--owner-datafile",
            "-o",
            exists=False,
            file_okay=True,
            dir_okay=False,
            writable=False,
            readable=True,
            resolve_path=False,
            help="File to read invoice supplier (owner) data instead Excel."),
    ] = None,
    verbose: Annotated[
        bool,
        typer.
        Option("--verbose", "-v", help="show detailed processing messages"),
    ] = False)
```

Extract data from an Excel file (save data to JSON format file with the same name as original file but `.json` extension).

Arguments:

- `invoice_type_code` - invoice type (for exaple regular invoice or storno) as this info is not usually subject of Excel file. Default to `380` (regular / usual invoice)
- `file_name` - files to process (wildcards allowed).
- `files_directory` - directory to be used to look for Excel files. Defaults to `invoice_files/`. NOTE: if default directory does not exists will consider current directory instead
- `owner_datafile` - File to read invoice supplier (owner) data instead Excel.

- `verbose` - show detailed processing messages". Defaults to `False`.

called_when_no_command

```
@app_cli.callback(invoke_without_command=True)
def called_when_no_command(
    ctx: typer.Context,
    version: Annotated[
        bool,
        typer.Option("--version", "-V", help="show application version"),
    ] = False)
```

Application global information (command agnostic).

run

NOTE: for `run` "reason to be" as copy of `app_cli` see iss `0.1.22b 240216piu_a`

chkxml

chkxml: modul de validare a facturii in sistemul ANAF E-Factura

Identification:

- code-name: `chkxml`
- copyright: (c) 2023 RENWare Software Systems
- author: Petre Iordanescu (petre.iordanescu@gmail.com)

Specifications:

- document cerinte initiale: `110-SRE-api_to_roefact_requirements.md` section `Componenta xl2roefact`
- INTRARI: fisier `f-XML`
- IESIRI: raport cu eventualele erori de validare

_tst_dropme2

__version__

`xl2roefact` version info.

```
######## ####              ########  ######
#  ##  # #  #              #      # ##    ##
##    ## #  #              ###  ### #  ##  #
##  ##  #  #                #  #   #  ##  #
##    ## #  #                #  #   #  ##  #
#  ##  # #  #####            #  #   #  ##  #
#  ##  # #      #            #  #   ##    ##
######## ########            ####    ######

                          ######## ########  ######   ######  ########
#######   ######          #      # #     #  ##    ## ##    ## #      #
#     ## ##    ## #######  #  ##### #  ####  #  ##  # #  ##  # ###  ###
#  ##  # #  ##  # #     #  #    #   #    #   #      # #  #####   #  #
#  ##### #  ##  # #######  #  ###   #  ###   #  ##  # #  #####   #  #
#  # #   #  ##  #          #  ##### #  #     #  ##  # #  ##  #   #  #
#  #  #  ##    ##          #      # #  #     #  ##  # ##    ##   #  #
####   #  ######           ######## ####     ########  ######    ####
```

__version__

current 0.9, previous 0.8

normalized_version

```
def normalized_version(raw_version: str = __version__) -> str
```

transform version string in canonical form.

Used in `__init__.py` to return `__version__` object as will be seen by package consumers

Arguments:

- `raw_version` - a raw version string. Defaults to package current version string.

Returns:

`str:` canonical version string

data

xl2roefact in-package designed to implement **data layer**

- Issued: 2024 March
- Author: Petre Iordanescu, petre.iordanescu@gmail.com
- (c) RENware Software Systems

Last update: June 6, 2024