

```
import camelot
import pandas as pd

from extractors.direct_downloads import extract_with_request
from models.registry import Registry
from utils.helpers import clean_str, date_from_pdf

class MDA(Registry):
    def __init__(self, **kwargs):
        """
        Child container for extracting, transforming, and loading registry data for Moldova

        :param kwargs: see models.registry for details
        """

        # assign constant values for final registry
        self.country_code = "MD"
        self.country_name = "Moldova"
        self.country_code_alpha3 = "MDA"
        self.country_code_alpha2 = "MD"
        self.country_code_alpha1 = "MD"
        self.country_code_alpha0 = "MD"
        self.country_code_alpha10 = "MD"
        self.country_code_alpha11 = "MD"
        self.country_code_alpha12 = "MD"
        self.country_code_alpha13 = "MD"
        self.country_code_alpha14 = "MD"
        self.country_code_alpha15 = "MD"
        self.country_code_alpha16 = "MD"
        self.country_code_alpha17 = "MD"
        self.country_code_alpha18 = "MD"
        self.country_code_alpha19 = "MD"
        self.country_code_alpha20 = "MD"
        self.country_code_alpha21 = "MD"
        self.country_code_alpha22 = "MD"
        self.country_code_alpha23 = "MD"
        self.country_code_alpha24 = "MD"
        self.country_code_alpha25 = "MD"
        self.country_code_alpha26 = "MD"
        self.country_code_alpha27 = "MD"
        self.country_code_alpha28 = "MD"
        self.country_code_alpha29 = "MD"
        self.country_code_alpha30 = "MD"
        self.country_code_alpha31 = "MD"
        self.country_code_alpha32 = "MD"
        self.country_code_alpha33 = "MD"
        self.country_code_alpha34 = "MD"
        self.country_code_alpha35 = "MD"
        self.country_code_alpha36 = "MD"
        self.country_code_alpha37 = "MD"
        self.country_code_alpha38 = "MD"
        self.country_code_alpha39 = "MD"
        self.country_code_alpha40 = "MD"
        self.country_code_alpha41 = "MD"
        self.country_code_alpha42 = "MD"
        self.country_code_alpha43 = "MD"
        self.country_code_alpha44 = "MD"
        self.country_code_alpha45 = "MD"
        self.country_code_alpha46 = "MD"
        self.country_code_alpha47 = "MD"
        self.country_code_alpha48 = "MD"
        self.country_code_alpha49 = "MD"
        self.country_code_alpha50 = "MD"
        self.country_code_alpha51 = "MD"
        self.country_code_alpha52 = "MD"
        self.country_code_alpha53 = "MD"
        self.country_code_alpha54 = "MD"
        self.country_code_alpha55 = "MD"
        self.country_code_alpha56 = "MD"
        self.country_code_alpha57 = "MD"
        self.country_code_alpha58 = "MD"
        self.country_code_alpha59 = "MD"
        self.country_code_alpha60 = "MD"
        self.country_code_alpha61 = "MD"
        self.country_code_alpha62 = "MD"
        self.country_code_alpha63 = "MD"
        self.country_code_alpha64 = "MD"
        self.country_code_alpha65 = "MD"
        self.country_code_alpha66 = "MD"
        self.country_code_alpha67 = "MD"
        self.country_code_alpha68 = "MD"
        self.country_code_alpha69 = "MD"
        self.country_code_alpha70 = "MD"
        self.country_code_alpha71 = "MD"
        self.country_code_alpha72 = "MD"
        self.country_code_alpha73 = "MD"
        self.country_code_alpha74 = "MD"
        self.country_code_alpha75 = "MD"
        self.country_code_alpha76 = "MD"
        self.country_code_alpha77 = "MD"
        self.country_code_alpha78 = "MD"
        self.country_code_alpha79 = "MD"
        self.country_code_alpha80 = "MD"
        self.country_code_alpha81 = "MD"
        self.country_code_alpha82 = "MD"
        self.country_code_alpha83 = "MD"
        self.country_code_alpha84 = "MD"
        self.country_code_alpha85 = "MD"
        self.country_code_alpha86 = "MD"
        self.country_code_alpha87 = "MD"
        self.country_code_alpha88 = "MD"
        self.country_code_alpha89 = "MD"
        self.country_code_alpha90 = "MD"
        self.country_code_alpha91 = "MD"
        self.country_code_alpha92 = "MD"
        self.country_code_alpha93 = "MD"
        self.country_code_alpha94 = "MD"
        self.country_code_alpha95 = "MD"
        self.country_code_alpha96 = "MD"
        self.country_code_alpha97 = "MD"
        self.country_code_alpha98 = "MD"
        self.country_code_alpha99 = "MD"

        # assign registry input data source and filename
        self.registry_input_dir = "registry_input_dir"
        self.registry_input_filename = "registry_input_filename"

        # assign registry input data extraction method
        self.registry_input_data_extraction_method = "registry_input_data_extraction_method"

        # assign updated-date scraping method for registry input data
        self.registry_input_data_updated_date_scraping_method = "registry_input_data_updated_date_scraping_method"

        # assign registry input data reading method
        self.registry_input_data_reading_method = "registry_input_data_reading_method"

        # assign registry data wrangling method
        self.registry_data_wrangling_method = "registry_data_wrangling_method"

        # extract, transform, and load registry data
        self.registry_data_extraction_transform_load_method = "registry_data_extraction_transform_load_method"

    def extract_registry(self):
        """
        Extracts registry input data from remote source

        :return: effect - creates [DATA_DIR|TEST_DIR]/[REGISTRY_INPUT_DIR]/[self.country_code]/[self.registry_filename] file
        """

        # extract registry input data
        self.registry_input_data = self.registry_input_data_extraction_method(self.registry_input_dir, self.registry_input_filename)

        # scrape updated-date for registry input data
        self.registry_input_data_updated_date = self.registry_input_data_updated_date_scraping_method(self.registry_input_data)

    def scrape_registry_updated(self, registry=None):
        """
        Scrapes and formats updated-date for registry input data from remote source

        :param (io.BytesIO) registry: bytes stream of extracted registry input data
        :return: effect - modifies self.updated
        """

        # handle when registry input data has been extracted but not in current session
        self.registry_input_data_updated_date = self.registry_input_data_updated_date_scraping_method(self.registry_input_data)

        # assign updated-date for registry input data
        self.registry_input_data_updated_date = self.registry_input_data_updated_date_scraping_method(self.registry_input_data)

    def read_registry(self):
        """
        Reads registry input data

        :return: effect - modifies self.registry
        """

        # read registry input data
        self.registry_input_data = self.registry_input_data_extraction_method(self.registry_input_dir, self.registry_input_filename)

        # prep registry input data
        self.registry_input_data = self.registry_input_data_reading_method(self.registry_input_data)

        # assign registry input data
        self.registry_input_data = self.registry_input_data_extraction_method(self.registry_input_dir, self.registry_input_filename)

    def wrangle_registry(self):
        """
        Transforms registry data

        :return: effect - modifies self.wrangled_records
        """

        # loop through raw registrations
        self.wrangled_records = self.registry_input_data_extraction_method(self.registry_input_dir, self.registry_input_filename)

        # initialize storage for wrangled representation of registration in focus
        self.wrangled_records = self.registry_input_data_extraction_method(self.registry_input_dir, self.registry_input_filename)

        # update list of transformed values to populate final registry
        self.registry_data_wrangling_method(self.wrangled_records)

    @staticmethod
    def __format_registry(registry):
        """
        Cleans and merges tables across multiple pdf pages

        :param (camelot.core.TableList) registry: tables of registrations from registry pdf to process
        :return: cleaned and merged transformation of 'registry' argument
        """

        # initialize storage for tables on multiple pages
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        # initialize storage for column names
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        # loop through pdf pages
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        # assign table on the page in focus
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        # assign column names once using the table on the first page
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        # drop header row from each table
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        # merge tables across multiple pages together
        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

    @staticmethod
    def __format_colname(colname):
        """
        Formats column name for namedtuple compatibility

        :param (str) colname: column name from registry input data to format
        :return: formatted copy of 'colname' argument suitable for namedtuple indexing
        """

        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)

        self.registry_data_wrangling_method(self.registry_input_dir, self.registry_input_filename)
```

