

```
from bs4 import BeautifulSoup
import pandas as pd

from extractors.direct_downloads import extract_with_request
from models.registry import Registry
from utils.helpers import date_to_iso

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

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

        super().__init__(**kwargs)

        # assign constant values for final registry
        self.country_code = "BGR"
        self.country_name = "Bulgaria"
        self.country_alpha3 = "BGR"
        self.country_alpha2 = "BG"
        self.country_numeric3 = "724"
        self.country_numeric2 = "72"
        self.country_numeric1 = "7"

        # assign registry input data filename
        self.registry_filename = "registry_bg.csv"

        # assign registry input data extraction method
        self.registry_extractor = "registry_bg_extractor"

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

        # assign registry input data reading method
        self.registry_reader = "registry_bg_reader"

        # assign registry data wrangling method
        self.registry_wrangler = "registry_bg_wrangler"

        # extract, transform, and load registry data
        self.extract_registry()

    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
        url = self._find_download_link()
        registry = extract_with_request(url, self.registry_extractor, registry_updated_scraper, registry_reader,
                                       self.registry_filename)

        # scrape updated-date for registry input data
        self.registry_updated = self.registry_updated_scraper(url)

        # read registry input data
        self.registry = self.registry_reader(registry)

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

        :param (str) url: download url for registry input data extracted from remote source
        :return: effect - modifies self.updated
        """

        if url is None:
            # handle when registry input data has been extracted but not in current session
            url = self._find_download_link()

        # assign dynamically created filename for registry input data
        filename = self._format_colname("registry_bg.csv")

        # capture updated-date for registry input data
        url = self._find_download_link(url, self.registry_updated_scraper, self.registry_updated_scraper)
        url = self._find_download_link(url, self.registry_updated_scraper, self.registry_updated_scraper)
        url = self._find_download_link(url, self.registry_updated_scraper, self.registry_updated_scraper)

        # assign updated-date for registry input data
        self.registry_updated = url

    def read_registry(self, registry=None):
        """
        Reads registry input data

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

        if registry is None:
            # handle when registry input data has been extracted but not in current session
            registry = self._find_download_link(self.registry_filename)

        # read registry input data
        registry = self.registry_reader(registry)

        # prep registry input data
        registry = self._format_colname(registry)

        # assign registry input data
        self.registry = registry

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

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

        # loop through raw registrations
        for reg in self.registry_wrangler(self.registry):
            # initialize storage for wrangled representation of registration in focus
            record = {}

            record["id"] = reg["id"]
            record["name"] = reg["name"]
            record["date"] = reg["date"]
            record["age"] = reg["age"]
            record["sex"] = reg["sex"]
            record["status"] = reg["status"]
            record["type"] = reg["type"]

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

    def __find_download_link(self):
        """
        Fetches the remote source html and finds the appropriate url to use when extracting registry input data

        :return: html href attribute representing the download url for registry input data
        """

        url = self._find_download_link(self.registry_updated_scraper, self.registry_updated_scraper, self.registry_updated_scraper)
        url = self._find_download_link(self.registry_updated_scraper, self.registry_updated_scraper, self.registry_updated_scraper)
        url = self._find_download_link(self.registry_updated_scraper, self.registry_updated_scraper, self.registry_updated_scraper)

        return url

    @staticmethod
    def __format_registry(registry):
        """
        Cleans styled table from Excel sheet

        :param (pandas.DataFrame) registry: registry input data to clean
        :return: cleaned copy of 'registry' argument
        """

        registry.columns = registry.columns.str.strip()
        registry.columns = registry.columns.str.lower()
        registry.columns = registry.columns.str.replace(" ", "_")

        return registry

    @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
        """

        colname = colname.replace(" ", "_").replace("-", "_").replace(".", "_").replace(":", "_").replace(":", "_")

        return colname
```

