oneM2M Extract Attribute tool 분석

라이브러리

```
from __future__ import annotations
import argparse, json, csv, os
from dataclasses import dataclass
from pathlib import Path
from typing import Dict, Set, Union, List, Tuple
from docx import Document
from docx.table import Table
import docx.opc.exceptions
from unidecode import unidecode
from rich.console import Console
from rich.progress import Progress, TextColumn, BarColumn
from rich.console import Console
from rich.table import Table
```

__future__: 상위 버전 기능 사용 가능

argparse: 명령행 인자 파싱

dataclasses: 데이터 클래스 사용 가능

pathlib: 파일 경로를 객체로 처리

typing: Dic, Set, Tuple, List 등과 같은 타입 사용 가능

docx: docx 파일 작성, 관리

rich: 텍스트 출력 시 색상, 밑줄, 마크다운, 테이블 등 변경 가능

데이터 클래스

```
@dataclass
class AttributeTable:
  headers:list
  attribute:int
  shortname:int
  occursIn:int
  filename:str
  category:str
```

```
@dataclass
class Attribute:
  """ Datastruture for an attribute, including shortname, longname, category etc.
  """
  shortname: str      # Lower case variant of the short name
  shortnameOrig: str
  attribute: str
  occurences:int
  occursIn:Set
  categories:Set
  documents:Set

  def asDict(self) -> dict:
    """ Return this dataclass as a dictionary.
    """
    return  { 'shortname' : self.shortnameOrig,
              'attribute' : self.attribute,
              'occursIn'  : sorted([ v for v in self.occursIn ]),
              'categories': sorted([ v for v in self.categories ]),
              'documents' : sorted([ v for v in self.documents ])
            }
```

AttributeTable: 문서에서 shortname tables을 찾기 위해 정의한 데이터 클래스

Attribute: 데이터 구조를 정의한 데이터 클래스

asDict 함수는 데이터 클래스를 딕셔너리 형식으로 return

변수

```
Attributes     = Dict[str, Attribute]
AttributesSN   = Dict[str, List[str]]
```

Atrributes: str을 key, Arritbute를 value로 하는 딕셔너리

AttributesSN: 잘 모르겠음

```
attributeTables:list[AttributeTable] = [

  # TS-0004
  AttributeTable(headers=['Parameter Name', 'XSD long name', 'Occurs in', 'Short Name'],  a
  AttributeTable(headers=['Root Element Name', 'Occurs in', 'Short Name'],        attribute
  AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name'],           attribute
  AttributeTable(headers=['Resource Type Name', 'Short Name'],              attribute=0, sh
  AttributeTable(headers=['Member Name', 'Occurs in', 'Short Name'],            attribute=0
  AttributeTable(headers=['Member Name', 'Short Name'],                 attribute=0, shortn

  # TS-0022
  AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name', 'Notes'],      attri
  AttributeTable(headers=['Member Name', 'Occurs in', 'Short Name', 'Notes'],       attribu
  AttributeTable(headers=['ResourceType Name', 'Short Name'],               attribute=0, sh

  # TS-0023
  AttributeTable(headers=['Resource Type Name', 'Short Name'],              attribute=0, sh
  AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name'],           attribute
  AttributeTable(headers=['Argument Name', 'Occurs in', 'Short Name'],          attribute=0

  # TS-0032
  AttributeTable(headers=['Attribute Name', 'Short Name'],               attribute=0, shor
  AttributeTable(headers=['Attribute Name', 'Occurs in', 'Short Name', 'Notes'],      attri
  AttributeTable(headers=['Member Name', 'Occurs in', 'Short Name', 'Notes'],       attribu

]
```

attributeTables: 문서별로 구조를 List 형식으로 정의한 테이블. 새 사양의 문서를 추가하려면 여기에 추가해야 함

함수

```python
def findAttributeTable(table:Table, filename:str) -> Union[AttributeTable, None]:
  """ Search and return a fitting AttributeTable for the given document table.
    Return `None` if no fitting entry could be found.
  """
  try:
    fn = filename.lower()
    row0 = table.rows[0]
    for snt in attributeTables:
      if len(snt.headers) != len(row0.cells):
        continue
      idx = 0
      isMatch = True
      while isMatch and idx < len(snt.headers):
        isMatch = row0.cells[idx].text == snt.headers[idx]
        idx += 1
      isMatch = isMatch and fn.startswith(snt.filename)
      if isMatch:
        return snt
  except:
    pass
  return None
```

findAttributeTable: 입력 문서가 attributeTables에 정의한 AttributeTable인지 검사하는 함수

1. 헤더가 일치하는 테이블 탐색
2. 한 줄씩 읽으면서 각 AttributeTable 정의와 같은지 검사
3. 적합한 AttributeTable을 반환. 적합한 항목이 없다면 None 리턴

processDocuments (진행바 관련 내용 제외)

```python
def processDocuments(documents:list[str], outDirectory:str, csvOut:bool) -> Tuple[Attributes, AttributesSN]:

  docs            = {}
  ptasks          = {}
  attributes:Attributes   = {}    # Mapping short name -> Attribute definition
  attributesSN:AttributesSN = {}    # Mapping attribute name -> List of short names
```

```python
    for d in documents:
      if not (dp := Path(d)).exists():
        stopProgress(f'[red]Input document "{d}" does not esist')
        return None, None
      if not dp.is_file():
        stopProgress(f'[red]Input document "{d}" is not a file')
        return None, None
      try:
        docs[d] = Document(d)
        ptasks[d] = progress.add_task(f'Processing {d} ...', total = 1000)
        progress.update(readTask, advance=1)
      except docx.opc.exceptions.PackageNotFoundError as e:
        stopProgress(f'[red]Input document "{d}" is not a .docx file')
        return None, None
      except Exception as e:
        stopProgress(f'[red]Error reading file "{d}"')
        console.print_exception()
        return None, None
```

documents:list[str]의 각 요소에 대해 탐색

1. 파일이 존재하는가? 아니라면 리턴 None, None
2. 경로가 파일인가? 아니라면 리턴 None, None
3. docs에 docx의 Document 개체들 저장
4. 3 과정에서 파일이 docx 문서가 아니라면 리턴 None, None

```
for docName, doc in docs.items():
```

```
for table in doc.tables: # Process the tables
  progress.update(processTask, advance=1)
  if (snt := findAttributeTable(table, docName)) is None: # Check if the table i
    continue
  headersLen = len(snt.headers) # Get the number of headers
  for r in table.rows[1:]: # Process the rows
    cells = r.cells # Get the cells
    if cells[0].text.lower().startswith('note:') or len(r.cells) != headersLen:
      continue
```

docs의 각 document에 대해

1. findArrtibuteTabel로 테이블이 정의돼있는지 검사
2. 테이블이 note:로 시작하거나 헤더 길이가 맞지 않는다면 스킵

```
  # Extract names and do a bit of transformations
  attributeName = unidecode(cells[snt.attribute].text).strip()
  shortnameOrig = unidecode(cells[snt.shortname].text.replace('*', '').strip())
  shortname     = shortnameOrig.lower()
  occursIn      = map(str.strip, unidecode(cells[snt.occursIn].text).split(',')) if snt.occursIn > -1 else ['n/a']

  # Don't process empty shortnames
  if not shortname:
    continue

  # Create or update entry for shortname
  if shortname in attributes: # Check if the shortname already exists
    entry = attributes[shortname]
    for v in occursIn: # Add the occursIn values
      entry.occursIn.add(v)
    entry.categories.add(snt.category) # Add the category
    entry.documents.add(docName) # Add the document
    entry.occurences += 1 # Increase the occurence counter
  else: # Create a new entry
    entry = Attribute(  shortname = shortname,
                shortnameOrig = shortnameOrig,
                attribute = attributeName,
                occurences = 1,
                occursIn = set([ v for v in occursIn ]),
                categories = set([ snt.category ]),
                documents = set([ docName ])
              )

  attributes[shortname] = entry
```

3. attributeName과 shortname을 추출한다. occursIn이 -1 이하면 'n/a'를 입력한다.
4. shortname이 없다면 스킵
5. shortname이 attributes()에 있다면 occursIn, category, document를 업데이트하고 shortname이 없다면 새 Arritbute를 생성한 후 attributes[shortname]에 저장한다.

```
    # Add the entry to the mapping list between attributes and short names. This is a list!
    if (al := attributesSN.get(entry.attribute)):
      # only add the entry to the mapping attributes -> entry if the shortname is different
      if len([ sn for sn in al if sn == entry.shortname ]) == 0:
        al.append(entry.shortname)
    else:
      attributesSN[entry.attribute] = [ entry.shortname ]
  continue
```

1. attribute가 이미 있는지 확인하고
2. 있다면 shortname이 있는지 체크하고 없으면 shortname 추가
3. 없다면 attribute 추가

```
with open(f'{outDirectory}{os.sep}attributes.json', 'w') as jsonFile:
  json.dump([ v.asDict() for v in attributes.values()], jsonFile, indent=4)
```

JSON 파일에 attributes 작성

Csv 파일로 output을 설정했다면 attribute와 shortname을 정렬 후 csv 파일에 작성

```
if csvOut: # Check if CSV output is enabled
  for docName, doc in docs.items():     # Individually for each input file
    progress.update(writeTask, advance=1)
    # write a sorted list of attribute / shortnames to a csv file
    with open(f'{outDirectory}{os.sep}{docName.rsplit(".", 1)[0] + ".csv"}', 'w') as csvFile:
      writer = csv.writer(csvFile) # Create a CSV writer
      writer.writerow(['Attribute', 'Short Name']) # Write the header
      writer.writerows(  # Write the data
        sorted( # Sort the data
          [ [attr.attribute, attr.shortnameOrig] for attr in attributes.values() if docName in attr.documents ], # Get the data
          key=lambda x: x[0].lower() )) # type: ignore [index] # Sort by attribute name
```

printAttributeTables 함수는 발견한 attribute를 출력. 중복된 shortname도 출력 가능

printAttributeCsv 함수는 발견한 attribute를 csv 파일에 작성

printDuplicateCsv 함수는 중복된 attribute를 csv 파일에 작성. 중복된 shortname을 csv 파일에 작성