import aiohttp

import asyncio

import logging

logging.basicConfig(level=logging.INFO,format='%(asctime)s - %(levelname)s ：%(message)s')

INDEX\_URL = 'https://spa5.scrape.center/api/book/?limit=18&offset={offset}'

DETAIL\_URL = 'https://spa5.scrape.center/api/book/{id}'

PAGE\_SIZE = 18

PAGE\_NUMBER = 100

CONCURRENCY = 5

semaphore = asyncio.Semaphore(CONCURRENCY)

session = None

async def scrape\_api(url):

async with semaphore:

try:

logging.info('scraping %s', url)

async with session.get(url) as response:

return await response.json()

except aiohttp.ClientError:

logging.error('error occured while scraping %s', url, exc\_info=True)

async def scrape\_index(page):

url = INDEX\_URL.format(offset=PAGE\_SIZE \* (page - 1))

return await scrape\_api(url)

import json

async def main():

global session

session = aiohttp.ClientSession()

scrape\_index\_tasks = [asyncio.ensure\_future(scrape\_index(page)) for page in range(1, PAGE\_NUMBER + 1)]

results = await asyncio.gather(\*scrape\_index\_tasks)

logging.info('results: %s', json.dumps(results, ensure\_ascii=False, indent=2))

if \_\_name\_\_ == '\_\_main\_\_':

asyncio.get\_event\_loop().run\_until\_complete(main())

ids = []

for index\_data in results:

if not index\_data: continue

for item in index\_data.get('results'):

ids.append(item.get('id'))

from motor.motor\_asyncio import AsyncIOMotorClient

MONGO\_CONNECTION\_STRING = 'mongodb://localhost:27017'

MONGO\_DB\_NAME = 'bookS'

MONGO\_COLLECTTON\_NAME = 'books'

client = AsyncIOMotorClient(MONGO\_CONNECTION\_STRING)

db = client[MONGO\_DB\_NAME]

collection = db[MONGO\_COLLECTTON\_NAME]

async def save\_data(data):

logging.info('saving data %s', data)

if data:

return await collection.update\_one({'id': data.get('id')}, {'$set': data}, upsert=True)

async def scrape\_detail(id):

url = DETAIL\_URL.format(id=id)

data = await scrape\_api(url)

await save\_data(data)

scrape\_detail\_tasks = [asyncio.ensure\_future(scrape\_detail(id)) for id in ids]

await asyncio.wait(scrape\_detail\_tasks)

await session.close()