"""  
The script basically generates the request\_data\_object and posts to BT.  
"""  
import json  
  
import getpass  
import logging  
from etl.core.util import uri\_post  
from etl.repo.pim\_pm.pl\_bbg\_batch import PlBbgBatchRepo  
from etl.repo.pim\_pm.pl\_bbg\_batch\_series\_vw import PlBbgBatchSeriesVwRepo  
from sqlalchemy.exc import DBAPIError  
  
BASE\_URL = 'http://ptp-dev/workshop/service/da/bbg\_transport/'  
  
  
class QueuerAgent(object):  
 """  
  
 """  
  
 def \_init\_\_(self):  
 logging.info('QueuerAgent')  
 self.USERNAME = getpass.getuser()  
  
 def get\_request(self,repo):  
 """  
  
 :param repo:  
 :return:  
 """  
 model = repo.model  
 try:  
 data = repo.query.filter(model.batch\_status\_code == 'IN\_QUEUE').all()  
 except DBAPIError:  
 logging.info('An Error has occured while connecting to the database.')  
 except:  
 logging.info('An Error has occured while fetching the data.')  
 return data  
  
  
 def get\_priority\_list(self,result):  
 """  
  
 :param result:  
 :return:  
 """  
 plist = []  
 data\_list =[]  
 history\_list=[]  
 for i in result:  
 if i.bbg\_program\_code == 'GETDATA':  
 data\_list.append(i)  
 else:  
 history\_list.append(i)  
 data\_list=self.get\_priority\_list\_by\_interface\_code(data\_list)  
 history\_list = self.get\_priority\_list\_by\_interface\_code(history\_list)  
 plist=data\_list+history\_list  
 return plist  
  
 def get\_priority\_list\_by\_interface\_code(self,result):  
 """  
  
 :param result:  
 :return:  
 """  
 plist = []  
 for i in result:  
 if i.bbg\_interface\_code == 'SAPI':  
 plist.insert(0, i)  
 else:  
 plist.append(i)  
 return plist  
  
  
 def get\_request\_object(self,objdata, result\_series):  
 """  
  
 :param objdata:  
 :param result\_series:  
 :return:  
 """  
 payload = dict()  
 payload["requestor\_code"] = "DA.PIMCOLIVE.DEV"  
 payload["request\_description"] = "TEST"  
 payload["program\_code"] = objdata.bbg\_program\_code  
 payload["interface\_code"] = objdata.bbg\_interface\_code  
 payload["response\_format\_code"] = "VERTICAL"  
 items\_list = []  
 request\_fields\_list = []  
 for i in result\_series:  
 element = dict()  
 element["cusip"] = ""  
 element["yellow\_key"] = i.bbg\_yellow  
 element["bb\_id"] = ""  
 element["tag"] = i.pl\_series\_code  
 element["isin"] = ""  
 element["bbg\_query"] = ""  
 element["ticker"] = i.bbg\_ticker  
 items\_list.append(element)  
 request\_fields\_list.append(i.bbg\_mnemonic)  
 payload["request\_data\_items"] = items\_list  
 payload["request\_fields"] = list(set(request\_fields\_list))  
 req\_options\_item = dict()  
 req\_options\_item["option\_name"] = "DATERANGE"  
 # if objdata.bbg\_program\_code == "GETDATA":  
 # payload["request\_options"] = []  
 # # req\_options\_item["option\_value"] = str(objdata.asof\_end\_date\_key)  
 # # time.strftime('%Y%m%d')  
 # elif objdata.bbg\_program\_code == "GETHISTORY":  
 # req\_options\_item["option\_value"] = str(objdata.asof\_start\_date\_key) + \  
 # "|" + str(objdata.asof\_end\_date\_key)  
 # req\_options = list()  
 # req\_options.append(req\_options\_item)  
 # payload["request\_options"] = req\_options  
 # payload = json.dumps(payload)  
  
 if objdata.bbg\_program\_code == "GETDATA":  
 req\_options\_item["option\_value"] = str(objdata.asof\_end\_date\_key)  
  
 elif objdata.bbg\_program\_code == "GETHISTORY":  
 req\_options\_item["option\_value"] = str(objdata.asof\_start\_date\_key) + \  
 "|" + str(objdata.asof\_end\_date\_key)  
  
 req\_options = list()  
 req\_options.append(req\_options\_item)  
 payload["request\_options"] = req\_options  
 payload = json.dumps(payload)  
 return payload  
  
  
 def post\_to\_bt(self,payload):  
 """  
  
 :param payload:  
 :return:  
 """  
 response = uri\_post(BASE\_URL + 'request\_data', payload)  
 return response  
  
  
 def update\_request(self,batch\_id, bt\_request\_id, progression\_url,  
 bt\_status\_code, request\_obj, batch\_status\_code, repo):  
  
 """  
  
 :param batch\_id:  
 :param bt\_request\_id:  
 :param progression\_url:  
 :param bt\_status\_code:  
 :param request\_obj:  
 :param batch\_status\_code:  
 :param repo:  
 :return:  
 """  
 model = repo.model  
 try:  
 update\_row = repo.query.filter(model.batch\_status\_code == 'IN\_QUEUE',  
 model.batch\_id == batch\_id).all()  
 except DBAPIError:  
 logging.info('An Error has occured while connecting to the database.')  
 except:  
 logging.info('An Error has occured while fetching the data.')  
 update\_row[0].batch\_status\_code = batch\_status\_code  
 update\_row[0].bt\_request\_id = bt\_request\_id  
 update\_row[0].bt\_status\_code = bt\_status\_code  
 update\_row[0].bt\_request\_payload = request\_obj  
 update\_row[0].bt\_response\_file\_path = progression\_url  
 repo.save(update\_row)  
  
 def run(self):  
 """  
  
 :return:  
 """  
 result = self.get\_request(PlBbgBatchRepo())  
 priority\_list = self.get\_priority\_list(result)  
 for i in priority\_list:  
 repo = PlBbgBatchSeriesVwRepo()  
 model = repo.model  
 try:  
 result\_batch = repo.query.filter(model.batch\_id == i.batch\_id).all()  
 except DBAPIError:  
 logging.info('An Error has occured while connecting to the database.')  
 except:  
 logging.info('An Error has occured while fetching the data.')  
 obj = self.get\_request\_object(i, result\_batch)  
 print obj  
 response = self.post\_to\_bt(obj)  
 self.update\_request(i.batch\_id, response['request\_id'], response['progression\_url'],  
 str(response['request\_status']),  
 str(obj), 'SENT\_TO\_BT', PlBbgBatchRepo())  
  
  
if \_name\_\_ == '\_\_main\_\_':  
 agent = QueuerAgent()  
 agent.run()