```
'''
Created on 12 Dec 2012

@author: gohariba
'''

from __future__ import with_statement
import sys
import os

# Java Imports
from java.io import FileOutputStream
from java.util import Date
from java.lang import System


from utils import loadJars, getProperties
####
##   Global Variable conatining the Application properties
####
props = getProperties()

if 'LIBDIR' not in props:
    sys.exit ("LIBDIR not defined \n")
loadJars(props['LIBDIR'])                    # Adds all the jars into sys.path

# Java Libraries import
from org.apache.poi.ss.usermodel import Cell, Row, Sheet
from org.apache.poi.ss.util import CellRangeAddress, SheetUtil
from org.apache.poi.xssf.usermodel import *
from org.apache.poi.xssf.streaming import SXSSFWorkbook, SXSSFSheet;

# Help functions to access Sybase Database
from jsybase import *


EXCEL_FLUSH_LIMIT = 1000

def readFile(file):
    """ Reads a file and returns its contents
        params : file -> String
        return : data -> Sting
    """
    with open(file, 'rb') as f:
        return f.read()

class MyWorkBook(SXSSFWorkbook):
    """ Subclass Streaming WorkBook provided by Apache POI  """

    def writeBeanstoSheet(self, beans, sheetname):
        """  Takes in a list of named Tuples ( a.k.a beans ) and the sheetname
             and pastes them into a tab with the name as the sheetname
             params : beans -> list of named tuples
                      sheetname  ->  String
             returns:  None
```

```
        """

        sh = self.createSheet(sheetname)
        rownum = 0
        ## Logic to populate the header and the first row
        row = sh.createRow(rownum)
        firstRow = beans.next()
        headers = firstRow._fields
        for index, item in enumerate(headers):
            cell = row.createCell(index)
            cell.setCellValue(item)
        rownum+=1

        row = sh.createRow(rownum)
        for index, item in enumerate(firstRow):
            cell = row.createCell(index)
            cell.setCellValue(item)
        rownum+=1

        ## Logic to populate the rest of the rows
        for bean in beans:
            row = sh.createRow(rownum)
            for index, item in enumerate(bean):
                cell = row.createCell(index)
                cell.setCellValue(item)
            rownum+=1
        print " %s rows - Completed " % rownum

    def writeRowstoSheet(self, datarows, sheetname):
        sh = self.createSheet(sheetname)
        rownum = 0
        for datarow in datarows:
            row = sh.createRow(rownum)
            for index, item in enumerate(datarow):
                cell = row.createCell(index)
                cell.setCellValue(item)
            rownum+=1
            if rownum % 1000 == 0:
                print " %s rows - Completed " % rownum
        print " %s rows - Completed " % rownum


def dumpQueryDatatoExcel():
    """ Takes query from System property 'QUERY'
        Runs the sql against the database
        Creates an excel file of the name specified by 'OUTPUTXLSX'
        Pastes the test results data into worksheets

        Note: Ensure that the following properties are defined when running this script
        by commandline
              1. LIBDIR     -> the lib directory where all the required jars reside
              2. QUERY   ->  the query that need to be run
              3. OUTPUTXLSX -> the xlsx file name to be created with the test results
              4. SERVERNAME -> the Sybase database server name
              5. DBNAME     -> the Sybase database name
```

```
            6. USERNAME   -> the Sybase database user name
            7. PASSWORD   -> the Sybase database password
            8. PORTNUMBER -> the Sybase database server's port number
    """

    # Output excel 2007
    wb = MyWorkBook(EXCEL_FLUSH_LIMIT)
    datarows = executeQuery1(props['QUERY'])
    wb.writeRowstoSheet(datarows, 'output')
    out = FileOutputStream(props['OUTPUTXLSX'])
    wb.write(out)
    out.close()
    del wb

def main():
    dumpQueryDatatoExcel()

if __name__ == '__main__':
    main()
    #import profile
    #profile.run('main()', 'test_ba6prof')
```

```
'''
Created on 17 Apr 2012

@author: Gokulnath Haribabu
'''
from __future__ import with_statement
import sys
import os

from com.ziclix.python.sql import zxJDBC
from collections import namedtuple

from java.lang import System
from java.util import Properties

from utils import ClassPathHacker, getProperties


# Get the Java Commandline Properties
props = getProperties()

# Load the jconn4 jar
try :
    jarLoad = ClassPathHacker()
    if 'LIBDIR' not in props:
        sys.exit ("LIBDIR not defined \n")

    jarLoad.addFile(os.path.join(props['LIBDIR'], 'jconn4.jar'))
except :
    sys.exit ("Loading jconn4.jar failed \n%s" % (str(sys.exc_info())))


params = { 'serverName' : props['SERVERNAME'],
           'databaseName' : props['DBNAME'],
           'user' : props['USERNAME'],
           'password' : props['PASSWORD'],
           'portNumber' : int(props['PORTNUMBER']) }


# get the connection
conn = apply(zxJDBC.connectx, ("com.sybase.jdbc4.jdbc.SybConnectionPoolDataSource",),
params)

def executeQuery(sql, params=None):
    with conn.cursor() as c:
        c.execute(sql, params)
        headers = [ x[0] for x in c.description ]
        bean = namedtuple('bean', headers)
        for row in c:
            yield bean._make(row)

def executeQuery1(sql, params=None):
    with conn.cursor() as c:
        c.execute(sql, params)
        headers = [ x[0] for x in c.description ]
```

```
        yield headers
        for row in c:
            yield row


#def saveData(data,table_name):
#    header = data[0]

#    c =conn.cursor()
#    try:
#        for row in data[1:]:
#            record = [ x for x in zip(header, row) if x[1] is not None ]
#            cols = [ x[0] for x in record ]
#            qs = [ '?' for x in cols ]
#            values = [ x[1] for x in record ]
#
#            cols_str = ','.join(cols)
#            qs_str = ','.join(qs)
#
#            insert_stmt = 'INSERT INTO %s ( %s ) VALUES ( %s )' % ( table_name,
cols_str, qs_str )
#
#            c.execute(insert_stmt, values)
#    except Exception, e:
#        print str(e)
#    finally:
#        c.close()
#    conn.commit()

# Test
```

```
'''
Created on 17 Apr 2012

@author: gohariba
'''
from java.util.concurrent import TimeUnit
from java.util.concurrent import Executors, ExecutorCompletionService
from java.util import Properties
from java.lang import System
from java.io import FileInputStream, BufferedInputStream
import sys
import os
import glob


def loadProperties (propertiesFilePath):
    """ Load a Java properties file into a Dictionary. """
    result = {}
    source = FileInputStream(propertiesFilePath)
    bis = BufferedInputStream(source)
    jprops = Properties()
    jprops.load(bis)
    bis.close()
    for key in jprops.keySet().iterator():
        result[key] = jprops.get(key)
    return result

#propertiesFilePath = sys.argv[1]
#props = loadProperties(propertiesFilePath)
props = System.getProperties()

def getProperties():
    return props

class ClassPathHacker :
##########################################################
# from http://forum.java.sun.com/thread.jspa?threadID=300557
#
# Author: SG Langer Jan 2007 translated the above Java to this
#       Jython class
# Purpose: Allow runtime additions of new Class/jars either from
#       local files or URL
#########################################################
    import java.lang.reflect.Method
    import java.io.File
    import java.net.URL
    import java.net.URLClassLoader

    def addFile (self, s):

        ###############################################
        # Purpose: If adding a file/jar call this first
        #       with s = path_to_jar
        ###############################################
```

```
        # make a URL out of 's'
        f = self.java.io.File (s)
        u = f.toURL ()
        a = self.addURL (u)

        return a

    def addURL (self, u):
        ##################################
        # Purpose: Call this with u= URL for
        #       the new Class/jar to be loaded
        ##################################

        parameters = [self.java.net.URL]
        sysloader =  self.java.lang.ClassLoader.getSystemClassLoader()
        sysclass = self.java.net.URLClassLoader
        method = sysclass.getDeclaredMethod("addURL", parameters)
        a = method.setAccessible(1)
        b = method.invoke(sysloader, [u])
        return u


def loadJars(libdir):

    for jar in glob.glob(os.path.join(libdir, r'*.jar')):
        sys.path.append(jar)

def loadJarsDynamically():

    jarLoad = ClassPathHacker()
    for jar in glob.glob(r'./lib/*.jar'):
        try :
            jarLoad.addFile(jar)
            print " Jar - %s - Loaded successfully " % jar
        except :
            sys.exit ("Loading jar - %s failed \n%s" % (jar, sys.exc_info()))


def shutdown_and_await_termination(pool, timeout):
    pool.shutdown()
    try:
        if not pool.awaitTermination(timeout, TimeUnit.SECONDS):
            pool.shutdownNow()
            if (not pool.awaitTermination(timeout, TimeUnit.SECONDS)):
                print >> sys.stderr, "Pool did not terminate"
    except InterruptedException, ex:
        # (Re-)Cancel if current thread also interrupted
        pool.shutdownNow()
        # Preserve interrupt status
        Thread.currentThread().interrupt()



def parallelMap(items, CallableClass):
```

```
MAX_CONCURRENT = 4
pool = Executors.newFixedThreadPool(MAX_CONCURRENT)
pool.prestartAllCoreThreads()
ecs = ExecutorCompletionService(pool)

el_time = time.time() - st_time
print "Pool up in %s secs " % el_time

submitted = 0
for item in items:
    ecs.submit(CallableClass(item))
    submitted += 1

el_time = time.time() - st_time
print "%s Tasks submitted successfully in %s secs " % (submitted, el_time)

while submitted > 0:
    tg = ecs.take().get()
    submitted -= 1
    if submitted % 10000 == 0:
            el_time = time.time() - st_time
            print "%s Tasks Remaining. Time Elapsed - %s secs " % (submitted,
            el_time)

print "shutting pool down..."
shutdown_and_await_termination(pool, 5)
el_time = time.time() - st_time
print "Completed in %s secs " % el_time
```