setup-as.py

```
import socket,sys,java,os

# ----------------------------------------------------------------
# 初期化
# ----------------------------------------------------------------

# サーバ情報
nodeName = AdminControl.getNode()
cellName = AdminControl.getCell()
server = AdminControl.queryNames('node='+ nodeName +',type=Server,*')
serverName = AdminControl.getAttribute(server, "name")
scope = AdminConfig.getid("/Node:" + nodeName + "/Server:" + serverName + "/")

#行区切り
lineSep = java.lang.System.getProperty('line.separator')

#JDBC関連定義
jdbcProviderName = "jdbc-provider-db2"
jndiName_sessions = "jdbc/Sessions"
authDataName = "auth-db2"

# DB接続情報
dbName = "dmtdb"
dbuser = "db2user"
dbpassword = "db2user"
dbport = 50000

# JVMヒープサイズ
jvmxms = 512
jvmxmx = 2048

# Spring Frameworkライブラリ格納ディレクトリ
# springlibdir = mavenRepository + "¥¥org¥¥springframework¥¥spring-instrument¥¥4.2.3.RELEASE"
springlibdir = ""

# springframeworkバージョン
springVersion = "4.2.3.RELEASE"

# サービス統合バス名
busName = "CveBus"

# セッションパーシスタンス用スキーマ（ローカル設定では未使用）
schemaName = ""

# DB2 Javaクライアントライブラリインストールディレクトリ
db2dir = ""
db2dirList = []
db2dirList.append("C:¥¥IBM¥¥SQLLIB")
db2dirList.append("C:¥¥Program Files¥¥IBM¥¥SQLLIB")
for dir in db2dirList:
    if (os.path.exists(dir)):
        db2dir = dir
        break
    else:
        continue
    #endif
#next
if (db2dir == ""):
    print "Error : DB2 Javaクライアントのパスが参照できません、スクリプトを中止します。"
    sys.exit()
else:
    print ("ディレクトリ[" + db2dir + "]をDB2 Javaクライアントパスとして適用します。")
    pass
#endif

# JPAライブラリ格納ディレクトリ情報
jpalibdir = ""
jpadirList = []
jpadirList.append("C:¥¥IBM¥¥WebSphere¥¥AppServerD¥¥plugins")
jpadirList.append("C:¥¥IBM¥¥WebSphere¥¥AppServer¥¥plugins")
jpadirList.append("C:¥¥IBM¥¥WebSphere¥¥AppServer_D8551¥¥plugins")
for dir in jpadirList:
    if (os.path.exists(dir)):
        jpalibdir = dir
        break
    else:
        continue
    #endif
#next
if (jpalibdir == ""):
    print "Error : WebSphereのプラグインパスが参照できません、スクリプトを中止します。"
    sys.exit()
```

```
else:
    print ("ディレクトリ[" + jpalibdir + "]をWebSphereプラグインパスとして適用します。")
    pass
#endif

print "DB名に[" + dbName + "を適用します。"
print "DB2のユーザ名に[" + dbuser + "]、パスワードに[" + dbpassword + "]を適用します。"

# dbName = getInput("データベース名を入力して下さい。何も入力しない場合は[MDPDB]が適用されます。", "dmtdb")
# dbuser = getInput("DB2ユーザ名を入力して下さい。何も入力しない場合は[db2user]が適用されます。", "db2user")
# dbpassword = getInput("DB2パスワードを入力して下さい。何も入力しない場合は[db2user]が適用されます。", "db2user")
# schemaName = getInput("スキーマ名を入力して下さい。何も入力しない場合は[MDP]が適用されます。", "MDP")

# ------------------------------------------------
# fuction getInput
# ------------------------------------------------
def getInput (prompt, defaultValue):
    print prompt
    retValue = raw_input()
    if (retValue == ""):
        retValue = defaultValue
    #endIf
    return retValue
#endDef

# ------------------------------------------------
# fuction wsadminToList
# ------------------------------------------------
def wsadminToList(inStr):
    outList=[]
    if (len(inStr)>0 and inStr[0]=='[' and inStr[-1]==']'):
        inStr = inStr[1:-1]
        tmpList = inStr.split(" ")
    else:
        tmpList = inStr.split("¥n")
    for item in tmpList:
        item = item.rstrip();
        if (len(item)>0):
            outList.append(item)
    return outList
#endDef

# ------------------------------------------------
# fuction createAuthDataEntry
# ------------------------------------------------
def createAuthDataEntry(authDataName,dbuser,dbpassword):

    # ------------------------------------------------------------
    # J2C認証データの存在チェックを行い、既に存在する場合は削除する
    # ------------------------------------------------------------
    parms = ["-alias", nodeName + "/" + authDataName]
    try:
        authDataEntry = AdminTask.getAuthDataEntry(parms)
        print "作成済みのJAAS - J2C 認証データ[" + authDataName + "]を削除します"
        AdminTask.deleteAuthDataEntry(parms)
    except:
        parms = []
    #endTry

    # ------------------------------------------------------------
    # J2C認証データを作成する
    # ------------------------------------------------------------
    try:
        parms = ["-alias", authDataName,"-user", dbuser,"-password", dbpassword, "-description"]
        AdminTask.createAuthDataEntry(parms)
        print "JAAS - J2C 認証データ[" + authDataName + "]を作成します"
        AdminConfig.save()
    except:
        print "Error : JAAS - J2C 認証データ[" + authDataName + "]の作成に失敗しました"
    #endTry

#endDef

# ------------------------------------------------
# fuction createJDBCProvider
# ------------------------------------------------
def createJDBCProvider():

    jdbc = AdminConfig.getid("/Node:" + nodeName + "/Server:" + serverName + "/JDBCProvider:" + jdbcProviderName + "/")
    if (len(jdbc) == 0):
        print "JDBCプロバイダ"+ jdbcProviderName +"を作成します"
```

setup-as.py

```
        classpath = ["${DB2_JCC_DRIVER_PATH}/db2jcc.jar", "${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar", "${DB2_JCC_DRIVER_PATH}/
db2jcc_license_cisuz.jar", "${PUREQUERY_PATH}/pdq.jar", "${PUREQUERY_PATH}/pdqmgmt.jar"]
        nativepath = ["${DB2_JCC_DRIVER_NATIVEPATH}"]

        parms = ["-scope", "Node=" + nodeName + ",Server=" + serverName]
        parms.extend(["-databaseType", "DB2", "-providerType", "DB2 Universal JDBC Driver Provider"])
        parms.extend(["-implementationType", "XA データ・ソース", "-implementationClassName", "com.ibm.db2.jcc.DB2XADataSource"])
        parms.extend(["-name", jdbcProviderName, "-description",""])
        parms.extend(["-classpath", classpath, "-nativePath", nativepath])
        AdminTask.createJDBCProvider(parms)
        AdminConfig.save()

        # ------------------------------------------------
        # createVariableSubstitutionEntry
        # ------------------------------------------------
        varSubstitutions = AdminConfig.list("VariableSubstitutionEntry",scope).split(lineSep)

        for varSubst in varSubstitutions:
            getVarName = AdminConfig.showAttribute(varSubst, "symbolicName")
            if (getVarName == "DB2_JCC_DRIVER_PATH"):
                print "WebSphere変数 DB2_JCC_DRIVER_PATHを削除します"
                AdminConfig.remove(varSubst)
                AdminConfig.save()
            #endIf
        #endFor

        print "WebSphere変数 DB2_JCC_DRIVER_PATHを作成します"
        varName = "(cells/" + cellName + "/nodes/" + nodeName + "/servers/" + serverName + "|variables.xml#VariableMap_1)"
        db2lib = db2dir + "¥java"
        varValue = [["symbolicName", "DB2_JCC_DRIVER_PATH"], ["description", ""], ["value", db2lib.replace('¥¥', '/')]]
        AdminConfig.create('VariableSubstitutionEntry', varName, varValue)
        AdminConfig.save()

    else:
        print "JDBCプロバイダ" + jdbcProviderName + "は作成済です"
    #endIf
    print AdminTask.listJDBCProviders()
#endDef

# ------------------------------------------------
# fuction createDataSource
# ------------------------------------------------
def createDataSource(dName,jName,authDataName,dbName,schemaName,dbServer,dbport):

    datasourceName  = dName
    jndiName        = "jdbc/" + jName
    templateId      = "DB2 Universal JDBC Driver DataSource(templates/system|jdbc-resource-provider-templates.xml#DataSource_DB2_UNI_1)"

    # ------------------------------------------------------------------
    # データソースの存在チェックを行い、既に存在する場合は削除する
    # ------------------------------------------------------------------
    datasource = AdminConfig.getid("/Node:" + nodeName + "/Server:" + serverName + "/JDBCProvider:" + jdbcProviderName + "/DataSource:" +
datasourceName + "/")
    if (len(datasource) != 0):
        datasourceId = datasource[datasource.find("(")+1:datasource.find(")")]
        AdminTask.deleteDatasource("DB2 Universal JDBC Driver Provider (XA)(" + datasourceId + ")")
        print "既存のデータソース[" + datasourceId + "]を削除しました"
    else:
        pass
    #endIf

    # ------------------------------------------------------------------
    # データソースを作成する
    # ------------------------------------------------------------------
    print "データソース[" + jndiName + "]を作成します"

    resProplist = [["name", "databaseName"], ["type", "java.lang.String"], ["value", dbName]]
    resProplist.append([["name", "driverType"], ["type", "java.lang.Integer"], ["value", 4]])
    resProplist.append([["name", "serverName"], ["type", "java.lang.String"], ["value", dbServer]])
    resProplist.append([["name", "portNumber"], ["type", "java.lang.Integer"], ["value", dbport]])
    resProplist.append([["name", "currentSchema"], ["type", "java.lang.String"], ["value", schemaName.upper()], ["required", "false"], ["
supportsDynamicUpdates", "false"]])

    if (dbName == "DMTDB" or dbName == "DMTDB1"):
        resProplist.append([["name", "resultSetHoldability"], ["type", "java.lang.Integer"], ["value", 1]])
        resProplist.append([["name", "allowNextOnExhaustedResultSet"], ["type", "java.lang.Integer"], ["value", 1]])
        resProplist.append([["name", "downgradeHoldCursorsUnderXa"], ["type", "java.lang.Boolean"], ["value", "true"]])
    else:
        pass
    #endIf

    attributes = [["jndiName", jndiName], ["authDataAlias", nodeName + "/" + authDataName], ["authMechanismPreference", "BASIC_PASSWORD"]]
```

```
    attributes.append(["propertySet",[["resourceProperties", resProplist]]])
    attributes.append(["xaRecoveryAuthAlias", nodeName + "/" + authDataName])
    attributes.append(["mapping", [["mappingConfigAlias", "DefaultPrincipalMapping"], ["authDataAlias", nodeName + "/" + authDataName]]])
    attributes.append(["connectionPool", [["connectionTimeout", "10"], ["maxConnections", "20"], ["minConnections", "5"], ["purgePolicy", "EntirePool"
]]])

    AdminJDBC.createDataSourceUsingTemplate(nodeName, serverName, jdbcProviderName, templateId, datasourceName, attributes)
    AdminConfig.save()

#endDef


# --------------------------------------------
# fuction setJVMProperties
# --------------------------------------------
def setJVMProperties():
    print ""
    print "javaagentを設定します"
    AdminTask.setJVMProperties("[-nodeName " + nodeName + " -serverName " + serverName + " -genericJvmArguments " + "¥"-Xquickstart -Dcom.ibm.xml.
xlxp.jaxb.opti.level=3 -javaagent:" + jpalibdir + "¥com.ibm.ws.jpa.jar" + " -javaagent:" + springlibdir + "¥spring-instrument-" + springVersion + ".
jar" + " ¥" ]")
    AdminConfig.save()
#endDef

# --------------------------------------------
# fuction createSessionTable
# --------------------------------------------
def createSessionTable():
    sessionTableName = "SessionTable"
    wc =  AdminConfig.list('WebContainer', scope)
    wc_att_validationExpression =["validationExpression" ,""]
    wc_att_name =["name ",sessionTableName]
    wc_att_value = ["value" ,schemaName.upper() +".SESSIONS"]
    wc_att_required = ["required", "false"]
    wcAttrs = [wc_att_validationExpression,wc_att_name,wc_att_value,wc_att_required]

    varSubstitutions = AdminConfig.list("Property",wc).split(lineSep)

    if varSubstitutions == ['']:
        print "session tableを作成します"
        AdminConfig.create('Property',wc,wcAttrs)
    else:
        for varSubst in varSubstitutions:
            getVarName = AdminConfig.showAttribute(varSubst, "name")
            if (getVarName != sessionTableName):
                print "session tableを作成します"
                AdminConfig.create('Property',wc,wcAttrs)
            else:
                print "session tableは作成済です"
            #endIf
        #endFor
    #endIf
    AdminConfig.save()
#endDef

# --------------------------------------------
# fuction createDataSourceForSessionDB
# --------------------------------------------
def createDataSourceForSessionDB():
    datasourceName_sessions  = "Sessions"

    datasourceSessions = AdminConfig.getid("/Node:" + nodeName + "/Server:" + serverName + "/JDBCProvider:" + jdbcProviderName + "/DataSource:" +
datasourceName_sessions + "/")
    if (len(datasourceSessions) == 0):

        print "データソース" + datasourceName_sessions + "を作成します"
        dataStoreHelperClassName = "com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper"
        alias = nodeName + "/" + authDataName
        otherAttributesList = [['containerManagedPersistence', 'true'],['xaRecoveryAuthAlias',alias],['componentManagedAuthenticationAlias',alias]]
        resourceAttributesList = [['serverName', 'localhost'], ['driverType', 4], ['portNumber', dbport]]

        ds_id = AdminJDBC.createDataSourceAtScope(scope,jdbcProviderName, datasourceName_sessions, jndiName_sessions,dataStoreHelperClassName,dbName,
otherAttributesList,resourceAttributesList)

        print "コンテナー管理認証別名を設定します"
        map_att_1 = ['authDataAlias',alias]
        map_att_2 = ['mappingConfigAlias','']
        mapAttrs = [map_att_1, map_att_2]

        AdminConfig.create('MappingModule', ds_id, mapAttrs)
        AdminConfig.save()
        print "データソース" + datasourceName_sessions + "を作成しました"
```

```
    else:
        print "データソース" + datasourceName_sessions + "は作成済です"
    #endIf
#endDef

# ------------------------------------------------
# fuction setSessionPersistence
# ------------------------------------------------
def setSessionPersistence():
    print "SessionDatabasePersistenceを設定します"
    AdminConfig.modify(AdminConfig.list('SessionManager', scope),'[[sessionPersistenceMode "DATABASE"]]')

    sdbp =  AdminConfig.list('SessionDatabasePersistence', scope)
    sdbp_att_password = ['password', dbpassword]
    sdbp_att_userId = ['userId', dbuser]
    sdbp_att_tableSpaceName = ['tableSpaceName', ""]
    sdbp_att_jndiName = ['datasourceJNDIName', jndiName_sessions]
    sdbp_att_db2RowSize = ['db2RowSize', "ROW_SIZE_4KB"]
    sdbpAttrs = [sdbp_att_password,sdbp_att_userId,sdbp_att_tableSpaceName,sdbp_att_jndiName,sdbp_att_db2RowSize]

    AdminConfig.modify(sdbp, sdbpAttrs)
    AdminConfig.save()
#endDef

# ------------------------------------------------
# fuction createJvmCustomProperty
# ------------------------------------------------
def createJvmCustomProperty(node, server, propName, value):
    server1 = AdminConfig.getid("/Node:" + node + "/Server:" + server + "/")
    jvm = AdminConfig.list("JavaVirtualMachine", server1)

    props = AdminConfig.showAttribute(jvm, "systemProperties")
    props = wsadminToList(props)
    propertyName = ""
    for prop in props:
        name = AdminConfig.showAttribute(prop, "name")
        name = name.rstrip()
        if (name == propName):
            propertyName = name
            break
        #endIf
    #endFor

    if (propertyName == ""):
        print "JVMカスタムプロパティ[" + propName + "]を設定します"
        parms = [["name",propName],["value",value],["required","false"]]
        attrList = AdminConfig.create("Property", jvm, parms )
        AdminConfig.save()
    else:
        print "JVMカスタムプロパティ[" + propName + "]は設定済です"
    #endIf
#endDef

# ------------------------------------------------
# fuction createDefaultJTADataSourceJNDIName
# ------------------------------------------------
def createDefaultJTADataSourceJNDIName():
    server1 = AdminConfig.getid("/Node:" + nodeName + "/Server:" + serverName)
    serv1 = AdminConfig.list('JavaPersistenceAPIService',server1)
    if (len(serv1) == 0):
        print "新規ーdefaultJTADataSourceJNDIName"
        #varName = "(cells/" + cellName + "/nodes/" + nodeName + "/servers/" + serverName + "|server.xml#ApplicationServer_%)"
        varName = AdminConfig.list('ApplicationServer',server1)
        varValue = [["defaultJTADataSourceJNDIName", ""], ["defaultNonJTADataSourceJNDIName", ""], ["defaultPersistenceProvider", "org.apache.openjpa.
persistence.PersistenceProviderImpl"],["enable" ,"true"],["properties",""]]
        AdminConfig.create('JavaPersistenceAPIService', varName, varValue)
    else:
        print "更新ーdefaultJTADataSourceJNDIName"
        print AdminConfig.showall(serv1)
        AdminConfig.modify(serv1,[['defaultPersistenceProvider','org.apache.openjpa.persistence.PersistenceProviderImpl']])
    #endIf
    AdminConfig.save()
#endDef

# ------------------------------------------------
# fuction updateHeapSize
# ------------------------------------------------
def updateHeapSize():
    server1 = AdminConfig.getid("/Node:" + nodeName + "/Server:" + serverName)
    serv1 = AdminConfig.list('JavaProcessDef',server1)
    serv2 = AdminConfig.list('JavaVirtualMachine',serv1)
```

setup-as.py

```
    AdminConfig.modify(serv2,[['initialHeapSize',jvmxms]])
    AdminConfig.modify(serv2,[['maximumHeapSize',jvmxmx]])
    AdminConfig.save()
    print "Heap設定済み"
#endDef


# ------------------------------------------------
# invoke fuctions
# ------------------------------------------------

#JAAS - J2C 認証データ作成(authDataName,dbuser,dbpassword)
createAuthDataEntry("auth-db2", "db2user","db2user")

#JDBCプロバイダ作成
createJDBCProvider()

#データソース作成(datasourceName,jndiName,authDataName,dbName,schemaName,dbServer,dbport)
#createDataSource("cca",      "cca",      "auth-db2", "DMTDB", "CCAAPL1", "localhost","50000")
#createDataSource("idk",      "idk",      "auth-db2", "DMTDB", "IDKAPL1", "localhost","50000")
#createDataSource("mdm",      "mdm",      "auth-db2", "DMTDB", "MDMAPL1", "localhost","50000")
#createDataSource("mdm2",     "mdm2",     "auth-db2", "DMTDB", "MDMAPL1", "localhost","50000")
#createDataSource("master",   "master",   "auth-db2", "DMTDB", "MSTAPL1", "localhost","50000")
#createDataSource("skg",      "skg",      "auth-db2", "DMTDB", "SKGAPL1", "localhost","50000")
#createDataSource("sol_old",  "sol_old",  "auth-db2", "DMTDB", "DMTAPL1", "localhost","50000")
#createDataSource("sol_old2","sol_old2","auth-db2", "DMTDB", "DMTAPL1", "localhost","50000")
#createDataSource("db2",      "db2",      "auth-db2", "DMTDB", "DMTAPL1", "localhost","50000")
#createDataSource("sol",      "sol",      "auth-db2", "DMTDB", "DMTAPL1", "localhost","50000")
createDataSource("mail",     "mail",     "auth-db2", "DMTDB", "MAILAPL1", "localhost","50000")

createDefaultJTADataSourceJNDIName()
updateHeapSize()

#javaagent設定
#setJVMProperties()

#セッションパーシスタンス作成
#createSessionTable()
#createDataSourceForSessionDB()
#setSessionPersistence()

#JVMカスタムプロパティ設定
createJvmCustomProperty(nodeName, serverName, "java.net.preferIPv4Stack", "true")
createJvmCustomProperty(nodeName, serverName, "db2.jcc.ccsid943Mapping", "2")

# 本ツール実行後は念のためWASを再起動してください
# 再起動しない場合、過去のConnectionが残ってテスト接続がうまくいかない可能性があります。
# JCAの状態を確認（状態の確認を押下）すればConnectionが残っているか確認可能です。
```