Imports System.Data

Imports System.Data.SqlClient

Imports System.Configuration.ConfigurationSettings

Imports WMS\_STD\_Formula.W\_Module

Public Class DBType\_SQLServer

#Region " Constructor "

#End Region

Public Shared gComputerName As String = "" ' System.Windows.Forms.SystemInformation.ComputerName

'\*\*\* Enum Execute Type

Enum EXEC

Reader

NonQuery

Scalar

End Enum

Enum DBConfigPlace

RegisTry

IniFile

End Enum

Enum enuCommandType

StoreProcedure

Text

End Enum

' \*\*\* Variable

Private fldSQLString As String = ""

Private fldExecuteType As EXEC = EXEC.Reader

Private fldDataReader As SqlDataReader

Private fldDataTable As DataTable = New DataTable

Private fldScalarOutput As String

Private fldCommandType As enuCommandType = enuCommandType.Text

Private fldConnectionTimeout As Integer = 0

'\*\*\* DB Operation Variable

Public Connection As New SqlConnection(WV\_ConnectionString)

Public DataAdapter As New SqlDataAdapter

Public SQLServerCommand As SqlCommand = New SqlCommand

Public DS As New DataSet

#Region " Properties "

' \*\*\* Properties

Public WriteOnly Property SetEXEC\_TYPE() As EXEC

Set(ByVal Value As EXEC)

fldExecuteType = Value

End Set

End Property

Public WriteOnly Property SetSQLString() As String

Set(ByVal Value As String)

fldSQLString = Value

End Set

End Property

Public ReadOnly Property GetDataTable() As DataTable

Get

Return fldDataTable

End Get

End Property

Public ReadOnly Property GetDataReader() As SqlDataReader

Get

Return fldDataReader

End Get

End Property

Public ReadOnly Property GetScalarOutput() As String

Get

Return fldScalarOutput

End Get

End Property

Public WriteOnly Property SetCommandType() As enuCommandType

Set(ByVal Value As enuCommandType)

fldCommandType = Value

End Set

End Property

Public WriteOnly Property SetConnectionTimeout() As Integer

Set(ByVal Value As Integer)

fldConnectionTimeout = Value

End Set

End Property

#End Region

Public Sub EXEC\_Command()

'If fldSQLString = "" Then

' MessageBox.Show("ClsDBUtil [EXEC\_Command] : Please define SQL Command ")

'End If

Try

With SQLServerCommand

If fldCommandType = enuCommandType.Text Then

.CommandType = CommandType.Text

ElseIf fldCommandType = enuCommandType.StoreProcedure Then

.CommandType = CommandType.StoredProcedure

End If

.CommandText = fldSQLString

.Connection = Connection

.CommandTimeout = fldConnectionTimeout

Select Case fldExecuteType

Case EXEC.Reader : fldDataReader = .ExecuteReader

Case EXEC.NonQuery : .ExecuteNonQuery()

Case EXEC.Scalar : fldScalarOutput = IIf(IsDBNull(.ExecuteScalar()), "", .ExecuteScalar)

End Select

End With

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Sub

Public Overloads Sub EXEC\_DataAdapter()

Try

' If Connection.State = ConnectionState.Open Then Connection.Close()

' connectDB()

DataAdapter = New SqlDataAdapter(fldSQLString, Connection)

DataAdapter.SelectCommand.CommandTimeout = 0

fldDataTable = New DataTable

DataAdapter.Fill(fldDataTable)

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Sub

Public Overloads Sub EXEC\_DAdapterName(ByVal DtName As DataTable)

Try

If Connection.State = ConnectionState.Open Then Connection.Close()

connectDB()

DataAdapter = New SqlDataAdapter(fldSQLString, Connection)

DataAdapter.Fill(DtName)

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Sub

Public Overloads Sub EXEC\_DataAdapter\_By\_Command()

Try

If Connection.State = ConnectionState.Open Then Connection.Close()

connectDB()

DataAdapter = New SqlDataAdapter(fldSQLString, Connection)

DataAdapter.Fill(fldDataTable)

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Sub

Public Function connectDB() As Boolean

Try

If Connection.State = ConnectionState.Closed Then

Connection.Open()

Return True

End If

Return False

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Function

Public Sub disconnectDB()

Try

If Connection.State = ConnectionState.Open Then

Connection.Close()

End If

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Sub

'add\_by weerapol

'add\_date 2013-01-23

'credit by big (yuttajak)

Public Function \_ExecQuery\_StoredProc\_Parameter(ByVal str\_storeName As [String], ByVal str\_param As [Object]()) As DataTable

Try

Dim dt\_detail As New DataTable("dt\_result")

DataAdapter = New SqlDataAdapter()

'Step 1 : Connecttion

If Connection.State = ConnectionState.Open Then Connection.Close()

connectDB()

With SQLServerCommand

.CommandText = str\_storeName

.CommandType = CommandType.StoredProcedure

.Connection = Connection

.CommandTimeout = fldConnectionTimeout

'Step 2 : Add Parameter

Dim j As Integer = 0

While j < str\_param.Length

SQLServerCommand.Parameters.AddWithValue(DirectCast(str\_param(j), [String]), str\_param(j + 1))

j += 2

End While

End With

'Step 3 : Add Command and Datable

DataAdapter.SelectCommand = SQLServerCommand

DataAdapter.Fill(dt\_detail)

Return dt\_detail

Catch ex As Exception

Throw ex

Finally

disconnectDB()

End Try

End Function

'add\_by top

'add\_date 2014-01-02

Public Function EXEC\_Scalar(ByVal strSQL As String) As Object

Try

' Dim objConn As System.Data.SqlClient.SqlConnection

Dim objCmd As System.Data.SqlClient.SqlCommand

' Dim strConnString As String

Dim myScalar As Object

If Connection.State = ConnectionState.Open Then Connection.Close()

connectDB()

' strSQL = "SELECT MAX(Used) FROM customer"

objCmd = New System.Data.SqlClient.SqlCommand()

With objCmd

.Connection = Connection

.CommandType = CommandType.Text

.CommandText = strSQL

End With

myScalar = objCmd.ExecuteScalar()

objCmd = Nothing

Return myScalar

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Function

Public Function EXEC\_Scalar(ByVal strSQL As String, ByVal Conn As SqlConnection, ByVal connTran As SqlTransaction) As Object

Try

' Dim objConn As System.Data.SqlClient.SqlConnection

Dim objCmd As System.Data.SqlClient.SqlCommand

' Dim strConnString As String

Dim myScalar As Object

'If Connection.State = ConnectionState.Open Then Connection.Close()

' connectDB()

' strSQL = "SELECT MAX(Used) FROM customer"

objCmd = New System.Data.SqlClient.SqlCommand()

With objCmd

.Connection = Conn

.Transaction = connTran

.CommandType = CommandType.Text

.CommandText = strSQL

End With

myScalar = objCmd.ExecuteScalar()

objCmd = Nothing

Return myScalar

Catch ex As SqlException

Throw New Exception("Database Exception " & vbNewLine & ex.Message)

Catch ex As Exception

Throw ex

End Try

End Function

'add\_by top

'add\_date 2014-01-02

'credit by NEUNG

#Region "[VARIABLE]"

'Public connectionStringName As String = ConfigurationManager.AppSettings.Get("ConnectionStringName")

'Public connectionString As String = ConfigurationManager.ConnectionStrings(connectionStringName).ConnectionString

'Public Connection As New SqlConnection(connectionString)

' Public dmssql\_Command As New SqlCommand

Private dmssql\_DataReader As SqlDataReader

Private dmssql\_DataAdapter As New SqlDataAdapter

Private dmssql\_ScalarResult As String = ""

Enum eData

DataReader

DataAdapter

End Enum

Enum eCommandType

Text

StoredProcedure

End Enum

#End Region

Public Sub DBconnect()

If Connection.State = ConnectionState.Open Then Connection.Close()

Connection.Open()

End Sub

Public Sub DBdisconnect()

Connection.Close()

End Sub

Public Function DBExeQuery(ByVal StrSQL As String, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal edata As eData = eData.DataReader, Optional ByVal eCommandTimeout As Integer = 0) As DataTable

Dim dt As New DataTable

Try

With SQLServerCommand

.CommandText = StrSQL

.Connection = Connection

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

'IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

End With

Select Case edata

Case edata.DataAdapter

dmssql\_DataAdapter.SelectCommand = SQLServerCommand

dmssql\_DataAdapter.Fill(dt)

Case Else

DBconnect()

dmssql\_DataReader = SQLServerCommand.ExecuteReader()

dt.Load(dmssql\_DataReader)

dmssql\_DataReader.Close()

DBdisconnect()

End Select

Return dt

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeQuery(ByVal StrSQL As String, ByVal Connection As SqlConnection, ByVal Transaction As SqlTransaction, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal edata As eData = eData.DataReader, Optional ByVal eCommandTimeout As Integer = 0) As DataTable

Dim dt As New DataTable

Try

With SQLServerCommand

.CommandText = StrSQL

.Connection = Connection

.Transaction = Transaction

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

'IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

End With

Select Case edata

Case edata.DataAdapter

dmssql\_DataAdapter.SelectCommand = SQLServerCommand

dmssql\_DataAdapter.Fill(dt)

Case Else

'dmssql\_DataReader = SQLServerCommand.ExecuteReader()

'dt.Load(dmssql\_DataReader)

'dmssql\_DataReader.Close()

dmssql\_DataReader = SQLServerCommand.ExecuteReader()

dt = GetDrToDTManuel(dmssql\_DataReader)

dmssql\_DataReader.Close()

End Select

Return dt

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeQuery(ByVal StrSQL As String, ByVal Connection As SqlConnection, ByVal Transaction As SqlTransaction, ByVal pSQLServerCommand As SqlCommand, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal edata As eData = eData.DataReader, Optional ByVal eCommandTimeout As Integer = 0) As DataTable

Dim dt As New DataTable

Try

With pSQLServerCommand

.CommandText = StrSQL

.Connection = Connection

.Transaction = Transaction

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

' IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

End With

Select Case edata

Case edata.DataAdapter

DataAdapter.SelectCommand = pSQLServerCommand

DataAdapter.Fill(dt)

Case Else

'fldDataReader = pSQLServerCommand.ExecuteReader()

'dt = GetDrToDTManuel(fldDataReader)

'fldDataReader.Close()

dmssql\_DataReader = pSQLServerCommand.ExecuteReader()

dt = GetDrToDTManuel(dmssql\_DataReader)

dmssql\_DataReader.Close()

End Select

Return dt

Catch ex As Exception

Throw ex

End Try

End Function

Public Overridable Function GetDrToDTManuel(ByVal dr As SqlDataReader) As DataTable

Dim dt = New DataTable

GC.Collect()

' dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)

Dim dtSchema As DataTable = dr.GetSchemaTable()

' You can also use an ArrayList instead of List<>

Dim listCols As List(Of DataColumn) = New List(Of DataColumn)()

If Not dtSchema Is Nothing Then

For Each drow As DataRow In dtSchema.Rows

Dim columnName As String = System.Convert.ToString(drow("ColumnName"))

Dim column As DataColumn = New DataColumn(columnName, CType(drow("DataType"), Type))

column.ReadOnly = False

column.Unique = CBool(drow("IsUnique"))

column.AllowDBNull = CBool(drow("AllowDBNull"))

column.AutoIncrement = CBool(drow("IsAutoIncrement"))

listCols.Add(column)

Dim a = column.ToString

dt.Columns.Add(column)

Next drow

End If

Do While dr.Read()

Dim dataRow As DataRow = dt.NewRow()

For i As Integer = 0 To listCols.Count - 1

dataRow((CType(listCols(i), DataColumn))) = dr(i)

Next i

dt.Rows.Add(dataRow)

Loop

Return dt

End Function

Public Function DBExeQuery\_Scalar(ByVal StrSQL As String, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal eCommandTimeout As Integer = 0) As String

Try

DBconnect()

With SQLServerCommand

.CommandText = StrSQL

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

' IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

.Connection = Connection

dmssql\_ScalarResult = .ExecuteScalar()

End With

DBdisconnect()

Return dmssql\_ScalarResult

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeQuery\_Scalar(ByVal StrSQL As String, ByVal Connection As SqlConnection, ByVal Transaction As SqlTransaction, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal eCommandTimeout As Integer = 0) As String

Try

With SQLServerCommand

.CommandText = StrSQL

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

'IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

.Connection = Connection

.Transaction = Transaction

dmssql\_ScalarResult = .ExecuteScalar()

End With

Return dmssql\_ScalarResult

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeNonQuery(ByVal StrSQL As String, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal eCommandTimeout As Integer = 0) As Integer

Try

DBconnect()

With SQLServerCommand

.CommandText = StrSQL

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

'IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

.Connection = Connection

Return .ExecuteNonQuery()

End With

DBdisconnect()

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeNonQuery(ByVal StrSQL As String, ByVal Connection As SqlConnection, ByVal Transaction As SqlTransaction, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal eCommandTimeout As Integer = 0) As Integer

Try

With SQLServerCommand

.CommandText = StrSQL

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

'IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

.Connection = Connection

.Transaction = Transaction

Return .ExecuteNonQuery()

End With

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeNonQuery(ByVal StrSQL As String, ByVal Connection As SqlConnection, ByVal Transaction As SqlTransaction, ByVal pSQLServerCommand As SqlCommand, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal eCommandTimeout As Integer = 0) As Boolean

Try

With pSQLServerCommand

.CommandText = StrSQL

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

' IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

.Connection = Connection

.Transaction = Transaction

Return .ExecuteNonQuery()

End With

Catch ex As Exception

Throw ex

End Try

End Function

Public Function DBExeQuery\_Scalar(ByVal StrSQL As String, ByVal Connection As SqlConnection, ByVal Transaction As SqlTransaction, ByVal pSQLServerCommand As SqlCommand, Optional ByVal eCommandType As eCommandType = eCommandType.Text, Optional ByVal eCommandTimeout As Integer = 0) As String

Try

With pSQLServerCommand

.CommandText = StrSQL

If eCommandTimeout = 0 Then

.CommandTimeout = fldConnectionTimeout

Else

.CommandTimeout = eCommandTimeout

End If

'IIf(eCommandType = eCommandType.StoredProcedure, .CommandType = CommandType.StoredProcedure, .CommandType = CommandType.Text)

If eCommandType = eCommandType.StoredProcedure Then

.CommandType = CommandType.StoredProcedure

Else

.CommandType = CommandType.Text

End If

.Connection = Connection

.Transaction = Transaction

dmssql\_ScalarResult = .ExecuteScalar()

End With

Return dmssql\_ScalarResult

Catch ex As Exception

Throw ex

End Try

End Function

Public Function GetDataTableFromDB(ByVal TableName As String, ByVal FieldSelect As String, ByVal strWhere As String) As DataTable

Try

Dim dt As New DataTable

Dim strSQL As String = ""

strSQL = "SELECT " & FieldSelect & " FROM " & TableName & " WHERE " & strWhere & " "

SetSQLString = strSQL

connectDB()

EXEC\_DataAdapter()

dt = GetDataTable

Return dt

Catch ex As Exception

Throw ex

Finally

disconnectDB()

End Try

End Function

Public Function GetScalarFromDB(ByVal TableName As String, ByVal FieldSelect As String, ByVal strWhere As String) As Object

Try

Dim strSQL As String = ""

strSQL = "SELECT " & FieldSelect & " FROM " & TableName & " WHERE " & strWhere & " "

SetSQLString = strSQL

SetCommandType = DBType\_SQLServer.enuCommandType.Text

SetEXEC\_TYPE = DBType\_SQLServer.EXEC.Scalar

connectDB()

EXEC\_Command()

Return GetScalarOutput

Catch ex As Exception

Throw ex

Finally

disconnectDB()

End Try

End Function

#Region " Varidate Data"

Public Shared Function ValidateisNull\_txt(ByVal ObjData As Object)

If String.IsNullOrEmpty(ObjData) Then Return ""

If ObjData Is Nothing Then Return ""

If IsDBNull(ObjData) Then Return ""

Return ObjData

End Function

Public Shared Function ValidateisNull\_bit(ByVal ObjData As Object)

If ObjData Is Nothing Then Return False

If IsDBNull(ObjData) Then Return False

Return ObjData

End Function

Public Shared Function ValidateisNull\_date(ByVal ObjData As Object)

If ObjData Is Nothing Then Return Nothing

If IsDBNull(ObjData) Then Return Nothing

Return ObjData

End Function

Public Shared Function ValidateisNull\_num(ByVal ObjData As Object)

If ObjData Is Nothing Then Return 0

If IsDBNull(ObjData) Then Return 0

Return ObjData

End Function

Public Shared Function ValidateIsNum(ByVal ObjData As String) As Boolean

Dim intRegexPattern As New System.Text.RegularExpressions.Regex("^[0-9]\*$")

If Not intRegexPattern.IsMatch(ObjData) Then

Return False

End If

Return True

End Function

Public Shared Function ValidateIsEmail(ByVal ObjData As String) As Boolean

Dim pattern As String = "^[a-zA-Z][\w\.-]\*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]\*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.]\*[a-zA-Z]$"

Dim emailAddressMatch As Text.RegularExpressions.Match = System.Text.RegularExpressions.Regex.Match(ObjData, pattern)

If emailAddressMatch.Success Then

Return True

Else

Return False

End If

End Function

#End Region

End Class