ADHERENCE DAL

Add reference to Oracle.DataAccess.dll

Web config

<add name="CommandConnection\_q" connectionString="User Id=cmdctr\_qa;Password=qacc2014;Data Source=DASHD.WORLD;Connection Lifetime=10;Connection Timeout=15;Pooling=true;Persist Security Info=false;Statement Cache Size=1;"/>

using System;

using System.Data;

using System.Text;

using System.Configuration;

using Oracle.DataAccess.Client;

using Oracle.DataAccess.Types;

namespace Convergys.CommandCenter.DataAccess

{

public sealed class Database

{

private string \_connectionString;

public Database(string configConnectionName)

{

\_connectionString = ConfigurationManager.ConnectionStrings[configConnectionName].ConnectionString;

}

public OracleConnection CreateConnection()

{

return new OracleConnection(\_connectionString);

}

public OracleConnection CreateConnection(string connectionString)

{

\_connectionString = connectionString;

return CreateConnection();

}

public void CloseConnection(OracleConnection connection)

{

if (connection != null)

{

if (connection.State == ConnectionState.Open)

{

connection.Close();

}

}

}

/// <summary>

/// Closes and disposes oracle connection and transaction

/// </summary>

/// <param name="connection"></param>

/// <param name="transaction"></param>

public void CloseConnection(OracleConnection connection, OracleTransaction transaction, bool isCommitable)

{

if (transaction != null)

{

if (isCommitable)

{

transaction.Commit();

}

else

{

transaction.Rollback();

}

transaction.Dispose();

}

if (connection != null)

{

if (connection.State == ConnectionState.Open)

{

connection.Close();

}

}

}

/// <summary>

/// Returns oracle command for SQL query

/// </summary>

/// <param name="sqlQuery"></param>

/// <returns></returns>

public OracleCommand GetSqlStringCommand(string sqlQuery)

{

OracleCommand command = new OracleCommand();

command.CommandType = CommandType.Text;

command.CommandText = sqlQuery;

command.BindByName = true;

return command;

}

/// <summary>

/// Returns oracle command for PL/SQL stored procedure

/// </summary>

/// <param name="storedProcedureName"></param>

/// <returns></returns>

public OracleCommand GetStoredProcCommand(string storedProcedureName)

{

OracleCommand command = new OracleCommand();

command.CommandType = CommandType.StoredProcedure;

command.CommandText = storedProcedureName;

command.BindByName = true;

return command;

}

/// <summary>

/// Adds a oracle input parameter to command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="dbType"></param>

/// <param name="value"></param>

public void AddInParameter(OracleCommand command, string paramterName, OracleDbType dbType, object value)

{

command.Parameters.Add(new OracleParameter(paramterName, dbType, value, ParameterDirection.Input));

}

/// <summary>

/// Adds a oracle input parameter to command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="dbType"></param>

/// <param name="value"></param>

//Added by PRAVEEN for manage reward\_type incentives

public void AddInOutParameter(OracleCommand command, string paramterName, OracleDbType dbType, object value, int size)

{

command.Parameters.Add(new OracleParameter(paramterName, dbType,size,value, ParameterDirection.InputOutput));

}

/// <summary>

/// Adds a oracle input parameter to command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="dbType"></param>

/// <param name="value"></param>

//Added by PRAVEEN for manage reward\_type incentives

public void AddInOutParameter(OracleCommand command, string paramterName, OracleDbType dbType, object value)

{

command.Parameters.Add(new OracleParameter(paramterName, dbType,value, ParameterDirection.InputOutput));

}

/// <summary>

/// Adds a oracle output parameter to command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="dbType"></param>

/// <param name="size"></param>

public void AddOutParameter(OracleCommand command, string paramterName, OracleDbType dbType, int size)

{

command.Parameters.Add(new OracleParameter(paramterName, dbType, size, ParameterDirection.Output, false, 0, 0, "", DataRowVersion.Default, null));

}

/// <summary>

/// Adds a oracle output parameter with WithArrayBindSize property to command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="dbType"></param>

/// <param name="size"></param>

public void AddOutParameter(OracleCommand command, string paramterName, OracleDbType dbType, int size, int arrayBindSize)

{

OracleParameter outParameter = new OracleParameter(paramterName, dbType, size);

outParameter.Direction = ParameterDirection.Output;

outParameter.ArrayBindSize = new int[arrayBindSize];

for (int index = 0; index < outParameter.ArrayBindSize.Length; index++)

{

outParameter.ArrayBindSize[index] = size;

}

command.Parameters.Add(outParameter);

}

/// <summary>

/// Adds a oracle refcursor parameter to command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

public void AddCursorOutParameter(OracleCommand command, string paramterName)

{

command.Parameters.Add(new OracleParameter(paramterName, OracleDbType.RefCursor, 0, ParameterDirection.Output, false, 0, 0, "", DataRowVersion.Default, null));

}

/// <summary>

/// Adds a parameter object to oracle command

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="dbType"></param>

/// <param name="size"></param>

/// <param name="direction"></param>

/// <param name="precision"></param>

/// <param name="scale"></param>

/// <param name="value"></param>

public void AddParameter(OracleCommand command, string paramterName, OracleDbType dbType, int size, ParameterDirection direction, byte precision, byte scale, object value)

{

command.Parameters.Add(new OracleParameter(paramterName, dbType, size, direction, false, precision, scale, "", DataRowVersion.Default, value));

}

public void AddParameter(OracleCommand command, OracleParameter param)

{

command.Parameters.Add(param);

}

/// <summary>

/// Gets the out parameters value

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <returns></returns>

public object GetParameterValue(OracleCommand command, string paramterName)

{

if (command.Parameters[paramterName].Value == null || command.Parameters[paramterName].Value == DBNull.Value)

{

if(command.Parameters[paramterName].OracleDbType == OracleDbType.Varchar2)

return string.Empty;

else

return 0;

}

else

{

if (command.Parameters[paramterName].OracleDbType == OracleDbType.Varchar2

&& command.Parameters[paramterName].Value is OracleString &&

((OracleString)command.Parameters[paramterName].Value).IsNull)

return string.Empty;

else

return command.Parameters[paramterName].Value;

}

}

/// <summary>

/// Sets the value of oracle parameter

/// </summary>

/// <param name="command"></param>

/// <param name="paramterName"></param>

/// <param name="value"></param>

public void SetParameterValue(OracleCommand command, string paramterName, object value)

{

command.Parameters[paramterName].Value = value;

}

/// <summary>

/// ExecuteNonQuery

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <returns></returns>

public int ExecuteNonQuery(OracleCommand command, OracleConnection connection)

{

command.Connection = connection;

return command.ExecuteNonQuery();

}

/// <summary>

/// ExecuteScalar

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <returns></returns>

public object ExecuteScalar(OracleCommand command, OracleConnection connection)

{

command.Connection = connection;

return command.ExecuteScalar();

}

/// <summary>

/// returns IDataReader

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <returns></returns>

public OracleDataReader ExecuteReader(OracleCommand command, OracleConnection connection)

{

command.Connection = connection;

return command.ExecuteReader(CommandBehavior.CloseConnection);

}

/// <summary>

/// returns IDataReader

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <param name="connection"></param>

/// <returns></returns>

public OracleDataReader ExecuteReader(OracleCommand command, OracleConnection connection,bool keepConnection)

{

command.Connection = connection;

if (keepConnection) return command.ExecuteReader();

else return command.ExecuteReader(CommandBehavior.CloseConnection);

}

/// <summary>

/// returns System.Xml.XmlReader

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <param name="rootTag"></param>

/// <param name="rowTag"></param>

/// <returns></returns>

public System.Xml.XmlReader ExecuteXMLReader(OracleCommand command, OracleConnection connection, string rootTag, string rowTag)

{

if (command.CommandType != CommandType.Text)

{

throw new System.ArgumentException("Command type can only be CommandType.Text");

}

command.XmlCommandType = OracleXmlCommandType.None;

command.XmlQueryProperties.RootTag = rootTag;

command.XmlQueryProperties.RowTag = rowTag;

command.Connection = connection;

return command.ExecuteXmlReader();

}

/// <summary>

/// returns DataSet

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <returns></returns>

public DataSet ExecuteDataSet(OracleCommand command, OracleConnection connection)

{

DataSet dataSet = null;

using (OracleDataAdapter dataAdapter = new OracleDataAdapter(command))

{

dataSet = new DataSet();

command.Connection = connection;

dataAdapter.Fill(dataSet);

}

return dataSet;

}

/// <summary>

/// returns DataSet

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <param name="dataSetName"></param>

/// <returns></returns>

public DataSet ExecuteDataSet(OracleCommand command, OracleConnection connection, string dataSetName)

{

DataSet dataSet = null;

using (OracleDataAdapter dataAdapter = new OracleDataAdapter(command))

{

dataSet = new DataSet(dataSetName);

command.Connection = connection;

dataAdapter.Fill(dataSet);

}

return dataSet;

}

/// <summary>

/// returns DataTable

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <returns></returns>

public DataTable ExecuteDataTable(OracleCommand command, OracleConnection connection)

{

DataTable dataTable = null;

using (OracleDataAdapter dataAdapter = new OracleDataAdapter(command))

{

dataTable = new DataTable();

command.Connection = connection;

dataAdapter.Fill(dataTable);

}

return dataTable;

}

/// <summary>

/// returns DataTable

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <param name="tableName"></param>

/// <returns></returns>

public DataTable ExecuteDataTable(OracleCommand command, OracleConnection connection, string tableName)

{

DataTable dataTable = null;

using (OracleDataAdapter dataAdapter = new OracleDataAdapter(command))

{

dataTable = new DataTable(tableName);

command.Connection = connection;

dataAdapter.Fill(dataTable);

}

return dataTable;

}

/// <summary>

/// returns DataTablel

/// </summary>

/// <param name="command"></param>

/// <param name="connection"></param>

/// <param name="tableName"></param>

/// <param name="tableNameSpace"></param>

/// <returns></returns>

public DataTable ExecuteDataTable(OracleCommand command, OracleConnection connection, string tableName, string tableNameSpace)

{

DataTable dataTable = null;

using (OracleDataAdapter dataAdapter = new OracleDataAdapter(command))

{

dataTable = new DataTable(tableName, tableNameSpace);

command.Connection = connection;

dataAdapter.Fill(dataTable);

}

return dataTable;

}

public DateTime GetCurrentDateTime()

{

DateTime today = DateTime.Today;

using (OracleConnection connection = this.CreateConnection())

{

using (OracleCommand command = this.GetSqlStringCommand("select sysdate currentdate from dual"))

{

try

{

connection.Open();

using (OracleDataReader reader = this.ExecuteReader(command, connection))

{

if (reader.Read())

{

today = DateTime.Parse(reader["currentdate"].ToString());

}

}

}

catch //(Exception ex) //not handling expection

{

}

finally

{

this.CloseConnection(connection);

}

}

}

return today;

}

public string[] PreparePaginatedQuery(string query, long rowFrom, long pageSize)

{

string[] paginationQuery = new string[2];

paginationQuery[0] = "select t2.\* from"

+ " (select t1.\*,rownum as row\_number from (" + query + ") t1"

+ " ) t2 where row\_number >= " + rowFrom + " and row\_number <= " + (rowFrom + (pageSize-1));

paginationQuery[1] = "select count(\*) totalRecords from (" + query + ")";

return paginationQuery;

}

}

}