using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

using System.Text.RegularExpressions;

namespace DAL

{

public class MySQLDBHelp

{

/// <summary>

/// 连接数据库

/// </summary>

/// <returns>返回MySqlConnection对象</returns>

public MySqlConnection getMySqlCon()

{

string constructorString = "server=IP地址;User Id=XXX;password=XXXX@;Database=XXXXX";

MySqlConnection myConnnect = new MySqlConnection(constructorString);

return myConnnect;

}

#region 执行MySqlCommand增删改操作，返回受影响的行数。

/// <summary>

/// 执行MySqlCommand

/// </summary>

/// <param name="M\_str\_sqlstr">SQL语句</param>

public int ExecuteNonMySQL(string M\_str\_sqlstr)

{

MySqlConnection mysqlcon = this.getMySqlCon();

mysqlcon.Open();

MySqlCommand mysqlcom = new MySqlCommand(M\_str\_sqlstr, mysqlcon);

int count = mysqlcom.ExecuteNonQuery();

mysqlcom.Dispose();

mysqlcon.Close();

mysqlcon.Dispose();

return count;

}

/// <summary>

/// 执行MySqlCommand

/// </summary>

/// <param name="M\_str\_sqlstr">SQL语句</param>

public int ExecuteNonMySQL(string M\_str\_sqlstr ,params MySqlParameter[] parameters )

{

MySqlConnection mysqlcon = this.getMySqlCon();

mysqlcon.Open();

MySqlCommand mysqlcom = new MySqlCommand(M\_str\_sqlstr, mysqlcon);

mysqlcom.Parameters.AddRange(parameters);

int count = mysqlcom.ExecuteNonQuery();

mysqlcom.Dispose();

mysqlcon.Close();

mysqlcon.Dispose();

return count;

}

/// <summary>

/// 对SQLite数据库执行增删改操作，返回受影响的行数。

/// </summary>

/// <param name="sql">要执行的增删改的SQL语句</param>

/// <returns></returns>

public int ExecuteNonQuery(String sql)

{

try

{

using (MySqlConnection connection = this.getMySqlCon())

{

connection.Open();

MySqlTransaction transaction = connection.BeginTransaction();

using (MySqlCommand cmd = new MySqlCommand())

{

try

{

PrepareCommand(cmd, connection, transaction, CommandType.Text, sql, null);

int rows = cmd.ExecuteNonQuery();

transaction.Commit();

cmd.Parameters.Clear();

return rows;

}

catch (MySqlException e1)

{

try

{

transaction.Rollback();

}

catch (Exception e2)

{

throw e2;

}

throw e1;

}

}

}

}

catch (Exception e)

{

throw e;

}

}

/// <summary>

/// 对SQLite数据库执行增删改操作，返回受影响的行数。

/// </summary>

/// <param name="sql">要执行的增删改的SQL语句</param>

/// <returns></returns>

public int ExecuteNonQuery(String sql, MySqlParameter[] cmdParams)

{

try

{

using (MySqlConnection connection = this.getMySqlCon())

{

connection.Open();

MySqlTransaction transaction = connection.BeginTransaction();

using (MySqlCommand cmd = new MySqlCommand())

{

try

{

PrepareCommand(cmd, connection, transaction, CommandType.Text, sql, cmdParams);

int rows = cmd.ExecuteNonQuery();

transaction.Commit();

cmd.Parameters.Clear();

return rows;

}

catch (MySqlException e1)

{

try

{

transaction.Rollback();

}

catch (Exception e2)

{

throw e2;

}

throw e1;

}

}

}

}

catch (Exception e)

{

throw e;

}

}

#endregion

#region 对数据库执行查询操作

/// <summary>

/// 创建一个MySqlDataReader对象

/// </summary>

/// <param name="M\_str\_sqlstr">SQL语句</param>

/// <returns>返回MySqlDataReader对象</returns>

public DataTable getMySqlRead(string M\_str\_sqlstr)

{

MySqlConnection mysqlcon = this.getMySqlCon();

mysqlcon.Open();

MySqlCommand mysqlcom = new MySqlCommand(M\_str\_sqlstr, mysqlcon);

MySqlDataAdapter mda = new MySqlDataAdapter(mysqlcom);

DataTable dt = new DataTable();

mda.Fill(dt);

mysqlcon.Close();

return dt;

}

/// <summary>

/// 创建一个MySqlDataReader对象

/// </summary>

/// <param name="M\_str\_sqlstr">SQL语句</param>

/// <returns>返回MySqlDataReader对象</returns>

public DataTable getMySqlRead(string M\_str\_sqlstr ,params MySqlParameter[] parameters)

{

MySqlConnection mysqlcon = this.getMySqlCon();

mysqlcon.Open();

MySqlCommand mysqlcom = new MySqlCommand(M\_str\_sqlstr, mysqlcon);

mysqlcom.Parameters.AddRange(parameters);

MySqlDataAdapter mda = new MySqlDataAdapter(mysqlcom);

DataTable dt = new DataTable();

mda.Fill(dt);

mysqlcon.Close();

return dt;

}

/// <summary>

/// 执行一条计算查询结果语句，返回查询结果（object）。

/// </summary>

/// <param name="SQLString">计算查询结果语句</param>

/// <returns>查询结果（object）</returns>

private object ExecuteScalar(string SQLString)

{

using (MySqlConnection connection = this.getMySqlCon())

{

using (MySqlCommand cmd = new MySqlCommand(SQLString, connection))

{

try

{

connection.Open();

object obj = cmd.ExecuteScalar();

if ((Object.Equals(obj, null)) || (Object.Equals(obj, System.DBNull.Value)))

{

return null;

}

else

{

return obj;

}

}

catch (MySql.Data.MySqlClient.MySqlException e)

{

connection.Close();

throw e;

}

}

}

}

/// <summary>

/// 用执行的数据库连接执行一个返回数据集的sql命令

/// </summary>

/// <param name="sql"></param>

/// <returns></returns>

public MySqlDataReader ExecuteReader(String sql)

{

try

{

//创建一个MySqlConnection对象

using (MySqlConnection connection = this.getMySqlCon())

{

connection.Open();

MySqlTransaction transaction = connection.BeginTransaction();

//创建一个MySqlCommand对象

using (MySqlCommand cmd = new MySqlCommand())

{

try

{

PrepareCommand(cmd, connection, transaction, CommandType.Text, sql, null);

MySqlDataReader reader = cmd.ExecuteReader(CommandBehavior.CloseConnection);

transaction.Commit();

cmd.Parameters.Clear();

return reader;

}

catch (MySqlException e1)

{

try

{

transaction.Rollback();

}

catch (Exception e2)

{

throw e2;

}

throw e1;

}

}

}

}

catch (Exception e)

{

throw e;

}

}

/// <summary>

/// 查询返回Dtaset

/// </summary>

/// <param name="sql"></param>

/// <returns></returns>

public DataTable ExecuteDataSet(String sql)

{

try

{

//创建一个MySqlConnection对象

using (MySqlConnection connection = this.getMySqlCon())

{

connection.Open();

MySqlTransaction transaction = connection.BeginTransaction();

//创建一个MySqlCommand对象

using (MySqlCommand cmd = new MySqlCommand())

{

try

{

PrepareCommand(cmd, connection, transaction, CommandType.Text, sql, null);

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = cmd;

//DataSet ds = new DataSet();

DataTable dt = new DataTable();

adapter.Fill(dt);

//adapter.Fill(ds);

transaction.Commit();

//清除参数

cmd.Parameters.Clear();

return dt;

}

catch (MySqlException e1)

{

try

{

transaction.Rollback();

}

catch (Exception e2)

{

throw e2;

}

throw e1;

}

}

}

}

catch (Exception e)

{

throw e;

}

}

/// <summary>

/// 查询返回Dtaset

/// </summary>

/// <param name="sql"></param>

/// <returns></returns>

public DataSet ExecuteDataSet(String sql ,MySqlParameter[] cmdParams)

{

try

{

//创建一个MySqlConnection对象

using (MySqlConnection connection = this.getMySqlCon())

{

connection.Open();

MySqlTransaction transaction = connection.BeginTransaction();

//创建一个MySqlCommand对象

using (MySqlCommand cmd = new MySqlCommand())

{

try

{

PrepareCommand(cmd, connection, transaction, CommandType.Text, sql, cmdParams);

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = cmd;

DataSet ds = new DataSet();

adapter.Fill(ds);

transaction.Commit();

//清除参数

cmd.Parameters.Clear();

return ds;

}

catch (MySqlException e1)

{

try

{

transaction.Rollback();

}

catch (Exception e2)

{

throw e2;

}

throw e1;

}

}

}

}

catch (Exception e)

{

throw e;

}

}

#endregion

#region 对数据执行分页操作

/// <summary>

/// 执行查询语句，返回DataTable

/// </summary>

/// <param name="SQLString">查询语句</param>

/// <returns>DataTable</returns>

private DataTable ExecuteDataTable(string SQLString)

{

using (MySqlConnection connection = this.getMySqlCon())

{

DataSet ds = new DataSet();

try

{

connection.Open();

MySqlDataAdapter command = new MySqlDataAdapter(SQLString, connection);

command.Fill(ds, "ds");

}

catch (MySql.Data.MySqlClient.MySqlException ex)

{

throw new Exception(ex.Message);

}

connection.Close();

return ds.Tables[0];

}

}

/// <summary>

/// 执行查询语句，返回DataTable

/// </summary>

/// <param name="db">数据库类型（Nozzle,Feeder,Head）</param>

/// <param name="SQLString">查询语句</param>

/// <returns>DataTable</returns>

private DataTable ExecuteDataTable(string db, string SQLString)

{

using (MySqlConnection connection = this.getMySqlCon())

{

DataSet ds = new DataSet();

try

{

connection.Open();

MySqlDataAdapter command = new MySqlDataAdapter(SQLString, connection);

command.Fill(ds, "ds");

}

catch (MySql.Data.MySqlClient.MySqlException ex)

{

throw new Exception(ex.Message);

}

return ds.Tables[0];

}

}

/// <summary>

/// 执行查询语句，返回DataSet

/// </summary>

/// <param name="SQLString">查询语句</param>

/// <returns>DataTable</returns>

private DataTable ExecuteDataTable(string SQLString ,params MySqlParameter[] cmdParms)

{

using (MySqlConnection connection = this.getMySqlCon())

{

MySqlCommand cmd = new MySqlCommand();

PrepareCommand(cmd, connection, null, CommandType.Text, SQLString, cmdParms);

using (MySqlDataAdapter da = new MySqlDataAdapter(cmd))

{

DataSet ds = new DataSet();

try

{

da.Fill(ds, "ds");

cmd.Parameters.Clear();

}

catch (MySql.Data.MySqlClient.MySqlException ex)

{

throw new Exception(ex.Message);

}

return ds.Tables[0];

}

}

}

//获取起始页码和结束页码

private DataTable ExecuteDataTable(string cmdText, int startResord, int maxRecord)

{

using (MySqlConnection connection = this.getMySqlCon())

{

DataSet ds = new DataSet();

try

{

connection.Open();

MySqlDataAdapter command = new MySqlDataAdapter(cmdText, connection);

command.Fill(ds, startResord, maxRecord, "ds");

}

catch (MySql.Data.MySqlClient.MySqlException ex)

{

throw new Exception(ex.Message);

}

return ds.Tables[0];

}

}

/// <summary>

/// 获取分页数据 在不用存储过程情况下

/// </summary>

/// <param name="recordCount">总记录条数</param>

/// <param name="selectList">选择的列逗号隔开,支持top num</param>

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

/// <param name="whereStr">条件字符 必须前加 and</param>

/// <param name="orderExpression">排序 例如 ID</param>

/// <param name="pageIdex">当前索引页</param>

/// <param name="pageSize">每页记录数</param>

/// <returns></returns>

public DataTable getPager(out int recordCount, string selectList, string tableName, string whereStr, string orderExpression, int pageIdex, int pageSize)

{

int rows = 0;

DataTable dt = new DataTable();

MatchCollection matchs = Regex.Matches(selectList, @"top\s+\d{1,}", RegexOptions.IgnoreCase);//含有top

string sqlStr = sqlStr = string.Format("select {0} from {1} where 1=1 {2}", selectList, tableName, whereStr);

if (!string.IsNullOrEmpty(orderExpression)) { sqlStr += string.Format(" Order by {0}", orderExpression); }

if (matchs.Count > 0) //含有top的时候

{

DataTable dtTemp = ExecuteDataTable(sqlStr);

rows = dtTemp.Rows.Count;

}

else //不含有top的时候

{

string sqlCount = string.Format("select count(\*) from {0} where 1=1 {1} ", tableName, whereStr);

//获取行数

object obj = ExecuteScalar(sqlCount);

if (obj != null)

{

rows = Convert.ToInt32(obj);

}

}

dt = ExecuteDataTable(sqlStr, (pageIdex - 1) \* pageSize, pageSize);

recordCount = rows;

return dt;

}

#endregion

/// <summary>

/// 准备执行一个命令

/// </summary>

/// <param name="cmd">sql命令</param>

/// <param name="conn">OleDb连接</param>

/// <param name="trans">OleDb事务</param>

/// <param name="cmdType">命令类型例如 存储过程或者文本</param>

/// <param name="cmdText">命令文本,例如:Select \* from Products</param>

/// <param name="cmdParms">执行命令的参数</param>

private void PrepareCommand(MySqlCommand cmd, MySqlConnection conn, MySqlTransaction trans, CommandType cmdType, string cmdText, MySqlParameter[] cmdParms)

{

if (conn.State != ConnectionState.Open)

conn.Open();

cmd.Connection = conn;

cmd.CommandText = cmdText;

if (trans != null)

cmd.Transaction = trans;

cmd.CommandType = cmdType;

if (cmdParms != null)

{

foreach (MySqlParameter parm in cmdParms)

cmd.Parameters.Add(parm);

}

}

}

}