// 使用连接字符串创建连接

string connStr = "Server=localhost;Port=3306;Uid =root;Password=123456;Database=ct;ConnectionTimeout=3";

using (MySqlConnection conn1 = new MySqlConnection(connStr))

{

conn1.Open();

//if(conn1.State == ConnectionState.Open) { } // 获取连接的状态，打开，关闭，执行命令中，获取数据中...

// DO Something

}

// 使用构造连接字符串创建连接

MySqlConnectionStringBuilder builder = new MySqlConnectionStringBuilder();

builder.Server = "localhost";

builder.Port = 3306;

builder.UserID = "root";

builder.Password = "123456";

builder.Database = "ct";

builder.ConnectionTimeout = 3; // 单位：秒

using (MySqlConnection conn2 = new MySqlConnection(builder.ConnectionString))

{

conn2.Open();

// DO Something

}

// 关闭数据库连接 Close 关闭连接，但是后续还可以再次Open使用 Dispose 不仅关闭连接，还会清理连接占用的资源，后续不能再Open使用

MySqlConnection conn3 = new MySqlConnection("Server=localhost;Port=3306;Uid=root;Pwd=123456;Database=ct;ConnectionTimeout=3");

conn3.Open();

conn3.Close();

conn3.Open();

conn3.Dispose();

Thread.Sleep(100);

MessageBox.Show(conn3.State.ToString());

conn3.Open();

MessageBox.Show(conn3.State.ToString());

/\*

\* Command

\* CommandText: Sql语句

\* CommandType：文本或存储过程

\* Connection：数据库连接

\* Parameters: Sql文本参数 或 存储过程的参数

\* Transaction：Command对象的事务

\*

\*

\* ExecuteScalar 返回结果集的第一行第一列，object引用，具体类型取决于结果集和ADO NET对SQL的字段类型的封装。

\* ExecuteNonQuery 返回受影响的行数，int类型

\* ExecuteReader 返回一个DataReader对象

\*

\*/

using (MySqlConnection conn = new MySqlConnection("server=localhost;uid=root;pwd=123456"))

{

using (MySqlCommand cmd = new MySqlCommand("Select \* from testdb; Select Name from testdb", conn))

{

using (MySqlDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

// 读取第一个结果集

// 直接读进一个Datatable，Table的列名默认的就是结果集的列名

DataTable dataTable = new DataTable();

dataTable.Load(reader);

}

// 读取第二个结果集，如果有的话

if (reader.NextResult())

{

while (reader.Read())

{

reader.GetInt32(0);

reader.GetBoolean("Sex");

reader.GetDateTime("Time");

Convert.ToDecimal(reader[3]);

}

}

}

}

}

// DataSet添加表

DataSet dataSet = new DataSet();

dataSet.Tables.Add(new DataTable("Student"));

// 表添加列 索引表

DataColumn dataColumn = new DataColumn();

dataColumn.ColumnName = "SNO";

dataColumn.DataType = typeof(Int32);

dataSet.Tables[0].Columns.Add(dataColumn);

dataSet.Tables["Student"].Columns.Add(new DataColumn("SName", typeof(string)));

// 表添加行 第一种方法

dataSet.Tables[0].Rows.Add(100, "张三");

// 表添加行 第二种方法

DataRow row = dataSet.Tables[0].NewRow();

row["SNO"] = 1001;

row["SName"] = "李四";

dataSet.Tables[0].Rows.Add(row);

/\*DataTable的查改删\*/

/\* DataTable的Find方法针对的是主键列\*/

/\*DataTable可以利用Find查找，但是只能根据主键查找\*/

// Table必须先设置主键，才能用Find

dataSet.Tables["Student"].PrimaryKey = new DataColumn[] { dataSet.Tables["Student"].Columns["SNO"] };

row = dataSet.Tables["Student"].Rows.Find(10086);

// 遍历查找 比较灵活，可以利用任何列进行查找

for (int i = 0; i < dataSet.Tables["Student"].Rows.Count; ++i)

{

if (Convert.ToInt32(dataSet.Tables["Student"].Rows[i]["SNO"]) == 10086)

{

row = dataSet.Tables["Student"].Rows[i];

break;

}

}

// 删除

if (row != null)

{

dataSet.Tables["Student"].Rows.Remove(row);

}

// 根据行号删除

dataSet.Tables["Student"].Rows.RemoveAt(0);

// 改的时候要锁定行

row.BeginEdit();

row.ItemArray[0] = 11111;

row["SNO"] = 11111;

row.EndEdit();