МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

“Харківський авіаційний інститут”

Факультет СУЛА

Кафедра математичного моделювання та штучного інтелекту

**Лабораторна робота №5**

з Організація баз даних

на тему: «ADO.NET: LINQ to SQL»

Виконав: студент 3 курсу групи №335a

напряму підготовки(спеціальності):

122 Комп’ютерні науки

(шифр і назва напряму підготовки(спеціальності))

Гринюк М.О.

(прізвище й ініціали студента)

Прийняв: ст. викл., к.т.н. Коробчинський К.П.

(посада, науковий ступінь, прізвище й ініціали)

Харків – 2020

**Цель работы:** Изучить принципы работы LINQ to Enteties.

**Ход работы:**

1. Пройти тест
2. Использовать в своём проекте класс логирования **(использовать текстовые файлы)**
3. Показать созданную базу данных
4. Подключить проект C# проверить соединение с БД вывести отладочную информацию в лог файл.
5. Создать **используя LINQ to Entities**и реализовать действия:

* Введение в LINQ to Entities
* Выборка и проекция из базы данных
* Сортировка
* Соединение таблиц
* Группировка
* Операции с множествами: объединение, пересечение, разность
* Агрегатные операции
* IEnumerable и IQueryable в Entity Framework
* Метод AsNoTracking

1. Оформить отчёт и сохранить на сервере;
2. **Знать ответы на контрольные вопросы.**

**Содержание работы:**

1. Для своего варианта на основе построенной логической модели своего варианта создать БД.
2. Подключить класс для вывода логирования.
3. Выполнить соединение с правильными и не правильными данными
4. Создать форму для входа и после успешной авторизации запустить основную, если авторизация не успешная, то главная  форма не должна создаваться. (повышенная сложность на максимальное количество баллов)
5. Для каждого пункта показать программный код и пример работы(скрин)
6. Для своего варианта показать подход**LINQ to Entities**

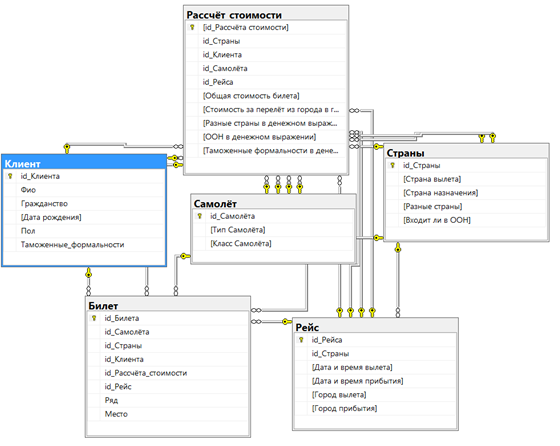
1. Показать пример работы с правильными и не правильными данными;

**Выполнение работы**

1. Для своего варианта на основе построенной логической модели своего

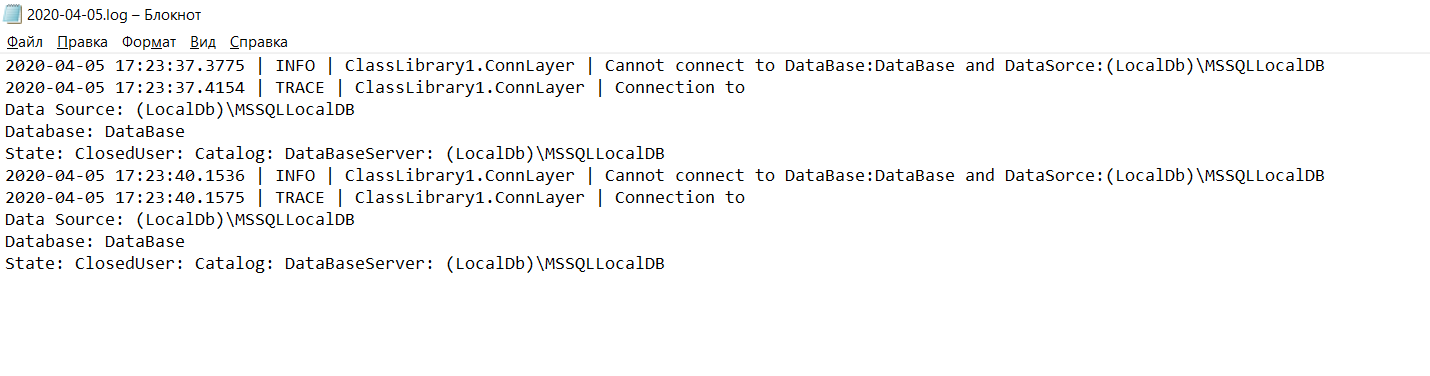
варианта создать БД;

**Логическая модель БД**



1. Выполнить соединение с правильными и не правильными данными

**Неправильные данные (базы данных не существует):**



[Table(Name = "Самолёт")]

public class Airplane

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "id\_Самолёта")]

public int Id { get; set; }

[Column(Name = "[Тип Самолёта]")]

public string Type\_Airplane { get; set; }

[Column(Name = "[Класс Самолёта]")]

public string Class\_Airplane { get; set; }

[Table(Name = "Клиент")]

class Client

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "id\_Клиента")]

public int Id { get; set; }

[Column(Name = "[Фио]")]

public string FullName { get; set; }

[Column(Name = "[Гражданство]")]

public string Nation { get; set; }

[Column(Name = "[Дата\_рождения]")]

public DateTime DateBirth { get; set; }

[Column(Name = "[Пол]")]

public byte Gender { get; set; }

[Column(Name = "[Таможенные\_формальности]")]

public byte Formalities { get; set; }

}

[Table(Name = "Страны")]

class Country

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "id\_Страны")]

public int Id { get; set; }

[Column(Name = "[Страна вылета]")]

public string CountryFrom { get; set; }

[Column(Name = "[Страна назначения]")]

public string CountryTo { get; set; }

[Column(Name = "[Разные страны]")]

public byte DifferentCountries { get; set; }

[Column(Name = "[Входит ли в ООН]")]

public byte IncludeOON { get; set; }

}

[Table(Name = "Customers")]

class Customers

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "idCustomer")]

public int Id { get; set; }

[Column(Name = "[UserName]")]

public string UserName { get; set; }

[Column(Name = "[Pass]")]

public string Pass { get; set; }

}

[Table(Name = "Рейс")]

class Flight

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "id\_Рейса")]

public int Id { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Страны")]

public int IDCountry { get; set; }

[Column(Name = "[Дата и время вылета]")]

public DateTime DateFrom { get; set; }

[Column(Name = "[Дата и время прибытия]")]

public DateTime DateTo { get; set; }

[Column(Name = "[Город вылета")]

public string TownFrom { get; set; }

[Column(Name = "[Город прибытия]")]

public string TownTo { get; set; }

}

[Table(Name = "Managers")]

class Managers

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "idManager")]

public int Id { get; set; }

[Column(Name = "[ManagerName]")]

public string ManagerName { get; set; }

[Column(Name = "[Pass]")]

public string Pass { get; set; }

}

[Table(Name = "Билет")]

class Ticket

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "id\_Билета")]

public int Id { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Самолёта")]

public int IdAirplane { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Страны")]

public int IdCountry { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Клиента")]

public int IdClient { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Рассчёта\_стоимости")]

public int IdCostCalc { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Рейс")]

public int IdFlight { get; set; }

[Column(Name = "[Ряд]")]

public int Row { get; set; }

[Column(Name = "[Место]")]

public int Place { get; set; }

}

[Table(Name = "Users")]

class Users

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "idUser")]

public int Id { get; set; }

[Column(Name = "[LoginName]")]

public string LoginName { get; set; }

[Column(Name = "[Pass]")]

public string Pass { get; set; }

}

[Table(Name = "Рассчёт\_стоимости")]

class Сost\_Сalculation

{

[Column(IsPrimaryKey = true, IsDbGenerated = true, Name = "[id\_Рассчёта стоимости]")]

public int Id { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Страны")]

public virtual Country IdCountry { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Клиента")]

public virtual Country IdClient { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Самолёта")]

public virtual Country IdAirplane { get; set; }

[Column(IsDbGenerated = true, Name = "id\_Рейса")]

public int IdFlight { get; set; }

[Column(IsDbGenerated = true, Name = "Общая стоимость билета")]

public int TotalCost { get; set; }

[Column(Name = "[Стоимость за перелёт из города в город]")]

public int CostFlight { get; set; }

[Column(Name = "[Разные страны в денежном выражении]")]

public int DifferentCountriesInValue { get; set; }

[Column(Name = "[ООН в денежном выражении]")]

public int OONinValue { get; set; }

[Column(Name = "[Таможенные формальности в денежном выражении]")]

public int FormalnostiInValue { get; set; }

}

2. Подключить класс для вывода логирования;

public class ConnectedLayer

{

public string connectionString = System.Configuration.ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;

private SqlCommand oSqlCom;

private SqlDataAdapter oSqlDtAdptr;

public string str1 { get; set; }

public string str2 { get; set; }

public DataTable datatable1;

public DataTable datatable2;

private DataSet ds;

private DataTable dt;

Logger logger = LogManager.GetCurrentClassLogger();

SqlConnectionStringBuilder builder;

public ConnectedLayer()

{

builder = new SqlConnectionStringBuilder(connectionString);

//ds = new DataSet();

var oSqlCon = new SqlConnection(connectionString);

using (SqlConnection connection = new SqlConnection(connectionString))

{

logger.Trace(

"Connection to" + Environment.NewLine +

"Data Source: " + oSqlCon.DataSource + Environment.NewLine +

"Database: " + oSqlCon.Database + Environment.NewLine +

"State: " + oSqlCon.State +

"User: " + builder.UserID +

"Catalog: " + builder.InitialCatalog +

"Server: " + builder.DataSource

);

Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);

config.ConnectionStrings.ConnectionStrings.Add(

new ConnectionStringSettings("ConnectionStr", "SomeConnectionString")

);

config.Save();

ConnectionStringsSection section = config.GetSection("connectionStrings") as ConnectionStringsSection;

if (section.SectionInformation.IsProtected)

{

section.SectionInformation.UnprotectSection();

}

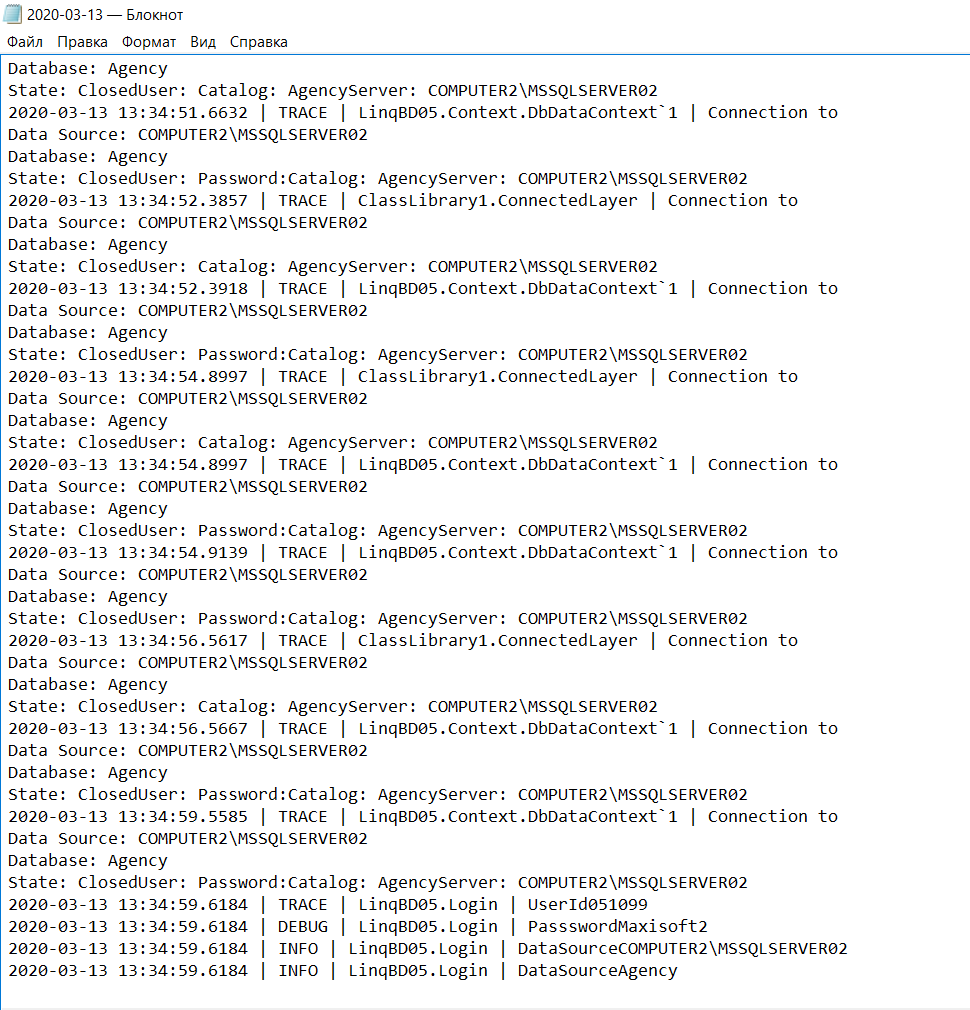
else

{

section.SectionInformation.ProtectSection("DataProtectionConfigurationProvider");

}

config.Save();

**Скриншоты работы:**

1. Для своего варианта реализовать **LINQ to SQL.**

**Класс DbdataContext**

public class DbDataContext<T> : DataContext where T : class

{

public string connectionString = System.Configuration.ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;

Logger logger = LogManager.GetCurrentClassLogger();

DataContext dc;

SqlConnectionStringBuilder builder;

ConnectedLayer connectedLayer;

public DbDataContext(string str)

: base(str)

{

builder = new SqlConnectionStringBuilder(connectionString);

var oSqlCon = new SqlConnection(connectionString);

using (SqlConnection connection = new SqlConnection(connectionString))

{

logger.Trace(

"Connection to" + Environment.NewLine +

"Data Source: " + oSqlCon.DataSource + Environment.NewLine +

"Database: " + oSqlCon.Database + Environment.NewLine +

"State: " + oSqlCon.State +

"User: " + builder.UserID +

"Catalog: " + builder.InitialCatalog +

"Server: " + builder.DataSource

);

Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);

config.ConnectionStrings.ConnectionStrings.Add(

new ConnectionStringSettings("ConnectionStr", "SomeConnectionString")

);

config.Save();

ConnectionStringsSection section = config.GetSection("connectionStrings") as ConnectionStringsSection;

if (section.SectionInformation.IsProtected)

{

section.SectionInformation.UnprotectSection();

}

else

{

section.SectionInformation.ProtectSection("DataProtectionConfigurationProvider");

}

config.Save();

}

}

public void Insert(ref T readed)

{

dc = new DataContext(connectionString);

dc.GetTable<T>().InsertOnSubmit(readed);

dc.SubmitChanges();

}

public void Delete(int readed)

{

dc = new DataContext(connectionString);

var air = dc.GetTable<Airplane>();

var query = from u in dc.GetTable<Airplane>()

where u.Id == readed

select u;

dc.GetTable<Airplane>().DeleteOnSubmit(query.First());

dc.SubmitChanges();

}

public void Update(int readed)

{

connectedLayer = new ConnectedLayer();

dc = new DataContext(connectionString);

var query = from u in dc.GetTable<Airplane>()

where u.Id == readed

select u;

foreach (Airplane airplane in query)

{

airplane.Type\_Airplane = DataCenter.Str1;

airplane.Class\_Airplane = DataCenter.Str2;

}

dc.SubmitChanges();

}

public Table<Airplane> airplanes { get { return this.GetTable<Airplane>(); } }

[Function(Name = "SelectId2")]

[return: Parameter(DbType = "Int")]

public int GetAgeRange([Parameter(Name = "id", DbType = "Int")] ref int id)

{

IExecuteResult result = this.ExecuteMethodCall(this, ((MethodInfo)(MethodInfo.GetCurrentMethod())), id);

id = ((int)(result.GetParameterValue(0)));

return id;

}}

**3.1 Чтение данных**

private void readToolStripMenuItem\_Click(object sender, EventArgs e)

{

connlayer = new ConnectedLayer();

dc = new DbDataContext<Airplane>(connlayer.connectionString);

Airplane airplane = new Airplane();

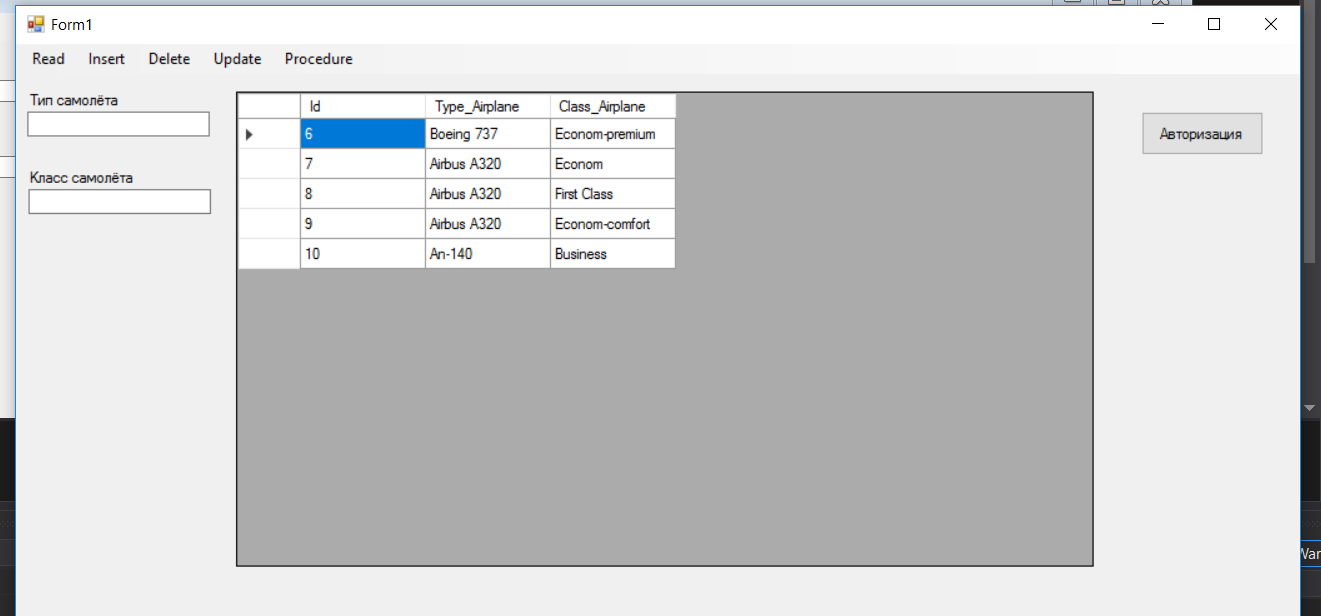
var query = from u in dc.GetTable<Airplane>()

where u.Id > 5

select u;

dataGridView1.DataSource = query.ToList();

dc.SubmitChanges();

}  


**3.2** **Вставка данных в базу данных**private void insertToolStripMenuItem\_Click(object sender, EventArgs e)

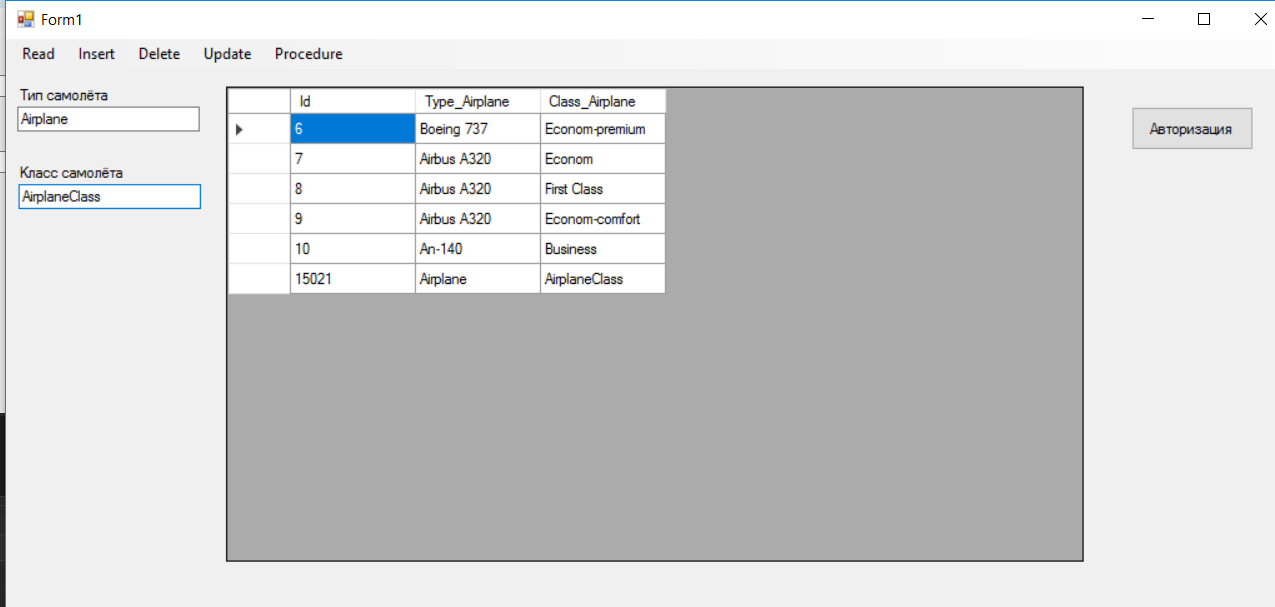
{

connlayer = new ConnectedLayer();

dc = new DbDataContext<Airplane>(connlayer.connectionString);

Airplane airplane = new Airplane { Type\_Airplane = $"{textBox1.Text}", Class\_Airplane = $"{textBox2.Text}" };

dc.Insert(ref airplane);

}  


**3.3.** **Удаление данных с бд**

private void deleteToolStripMenuItem\_Click(object sender, EventArgs e)

{

connlayer = new ConnectedLayer();

dc = new DbDataContext<Airplane>(connlayer.connectionString);

Airplane airplane = new Airplane { Id = (int)dataGridView1.CurrentCell.Value };

dc.Delete(airplane.Id);

}

**3.4 Изменение данных**

private void UpdateToolStripMenuItem\_Click(object sender, EventArgs e)

{

DataCenter.Str1 = textBox1.Text;

DataCenter.Str2 = textBox2.Text;

dc = new DbDataContext<Airplane>(connlayer.connectionString);

Airplane airplane = new Airplane { Id = (int)dataGridView1.CurrentCell.Value };

dc.Update(airplane.Id);

}

* 1. **Процедура**

Выборсамолёта по его id.private void procedureToolStripMenuItem1\_Click(object sender, EventArgs e)

{

dc = new DbDataContext<Airplane>(connlayer.connectionString);

int id1 = (int)dataGridView1.CurrentCell.Value;

var air = dc.GetTable<Airplane>();

int \_id = dc.GetAgeRange(ref id1);

var query = from i in air

where i.Id == \_id

select i;

foreach (Airplane airplane in query)

dataGridView1.DataSource = query;

}

Создать форму для входа и после успешной авторизации запустить основную, если авторизация не успешная, то главная форма не должна создаваться. (повышенная сложность на максимальное количество баллов);  
  
**Код программы:**  
public partial class Login : Form

{

public string connectionString = System.Configuration.ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;

public Login()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder(connectionString);

Form1 form1 = new Form1();

NLog.Logger logger = LogManager.GetCurrentClassLogger();

using (SqlConnection conn = new SqlConnection(connectionString))

{

try

{

sb.UserID = textBox1.Text;

sb.Password = textBox2.Text;

conn.StateChange += connection\_StateChange;

conn.Open();

MessageBox.Show("Logined");

conn.Close();

MessageBox.Show("Connection closed");

this.Hide();

form1.ShowDialog();

}

catch

{

logger.Trace("UserId" + sb.UserID);

logger.Debug("Passsword" + sb.Password);

logger.Info("DataSource" + sb.DataSource);

logger.Info("DataSource" + sb.InitialCatalog);

}

finally

{

conn.Close();

}

}

}

static void connection\_StateChange(object sender, System.Data.StateChangeEventArgs e)

{

SqlConnection connection = sender as SqlConnection;

MessageBox.Show //вывод информации о соединении и его состоянии

(

"Connection to" + Environment.NewLine +

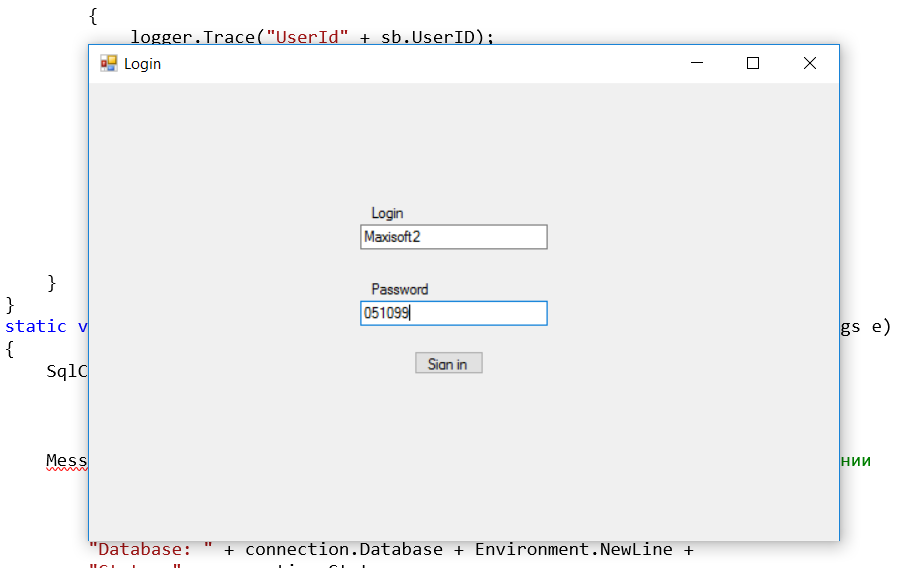
"Data Source: " + connection.DataSource + Environment.NewLine +

"Database: " + connection.Database + Environment.NewLine +

"State: " + connection.State

);

}

}  
  
  
  
**Скриншоты работы:**  


**Вывод:** Изучили принципы работы LINQ to SQL.