**Лекція 17 Триланкова система технології CORBA**

• Технологія CORBA

• IIOPNet

• Приклади систем, що складаються з різнорідних компонентів

• IDL і його використання в рамках IIOPNet

• Створення простого CORBA-клієнта на C #

• Створення простого CORBA-сервера на C #

**Технологія CORBA**

CORBA –це *CommonObjectRequestBrokerArchitecture*(загальна архітектура брокера об’єктних запитів). Це незалежна від мови програмування модель взаємодії розподілених об’єктів.

Технологія надає три наступні переваги:

1. Прозорість виклику (invocationtransparency)
2. Прозорість реалізації (implementationtransparency)
3. Прозорість локалізації (locationtransparency)

CORBA чітко відрізняє сервер від його об’єктів. *Сервер* являє собою виконавчу одиницю; це – процес. *Об’єкт* реалізує інтерфейс. Сервер може містити один або декілька об’єктів. CORBA визначає чотири стратегії активації:

1. Розділений сервер (***shared server)*** – декілька об’єктів можуть розміщуватися в рамках однієї й тої самої програми (процеса);
2. Виділений сервер(***unshared server***) – кожний об’єкт розміщується у окремому серверному процесі;
3. Сервер з єдиним методом(***server-per-method***) – новий сервер завантажується при виникненні нового запиту;
4. Сервер довгострокового зберігання(***persistent server)*** – сервери активуються незалежно від BOA.

## OMG – Object Management Group

OMG – це консорціум, який включає більше 700 компаній, і розробляє специфікації інтерфейсів.

CORBA є промисловим стандартом для реалізації функціональності міжплатформову взаємодії гетероморфний систем. Основні розділи цього стандарту - це стандарт для брокерів об'єктних запитів (CORBA ORB), стандарт мови визначення інтерфейсів OMG IDL, і стандарт переносимого протоколу взаємодії між ORB (IIOP).

Достоїнствами стандарту CORBA є висока переносимість, гнучкість, ефективність і масштабованість рішень заснованих на ньому, а також наявність великої кількості (часто вільних) реалізацій брокерів об'єктних запитів під різні програмно-апаратні платформи і технології розробки.

Недоліками цього стандарту є його безпосередня складність (і, іноді, неоднозначність), складність навчання розробці на основі цього стандарту, і високий рівень спадкоємності багатьох недоліків (як і достоїнств) технологій C++ і Java.

На сьогоднішній день можна відзначити рівень розвитку стандарту CORBA і його реалізацій, достатній для того, щоб будувати на основі нього модулі взаємодії (компонентів) промислових інформаційних систем.

Таким чином, при наявності відповідних інструментальних засобів, на даний момент технологію CORBA можна досить вигідно використовувати як інтеграційне засіб для наведення програмних мостів між гетероморфний системами.

# 1. Основні поняття в технології CORBA

## IDL

IDL – мова однорідної специфікації інтерфейсів різноманітних інформаційних ресурсів, інкапсульованих за допомогою CORBA. IDL – це контрактна мова. Контракт IDL містить опис будь-яких ресурсів або сервісів, які серверний компонент бажає надати своїм клієнтам. CORBA IDL має наступні основні елементи:

* ***Модулі*** – простір імен для групування набору описів класів. Модуль має імя, яке складається із одного чи декількох ідентифікаторів розділених символами «::».
* ***Інтерфейси*** – визначають набір методів, які можуть бути викликані клієнтом на даному об’єкті.
* ***Операції***  - еквівалент CORBA для терміна метод. Це сервіс, що може бути викликаний клієнтом. IDL визначає сигнатуру операції, тобто параметри методу і результати, які повертаються.
* ***Типи даних*** – використовуються для опису допустимих значень параметрів, атрибутів, виключних ситуацій та результатів, які повертаються.

## ORB - Object Request Broker

Брокер об’єктних запитів – це об’єктна шина. Вона дозволяє об’єктам прозоро генерувати запити і отримувати відповідні відклики від інших локальних чи віддалених об’єктів. Клієнт нічого не знає про механізми, які використовуються для комунікації, активізації або зберігання серверних об’єктів. ORB – проміжне програмне забезпечення, яке встановлює клієнт-серверні відносини між об’єктами.

## GIOP - GeneralInter-OrbProtocol

Загальний протокол обміну між брокерами об’єктних запитів. Протокол визначає набір форматів повідомлень і загального представлення даних (commondatarepresentation) для комунікацій між ORB. GIOP був спеціально створений для забезпечення взаємодії ORB-to-ORB. Протокол визначає сім форматів повідомлень, які охоплюють всю семантику запит/відповідь для ORB.

## IIOP - InternetInter-ORB Protocol

IIOP – це протокол обміну повідомленнями між об’єктними брокерами. В доповнення до IIOP архітектура CORBA підтримує *ESIOP (Environment-Specific Inter-ORB Protocol) – Залежні від Середовища Протоколи Взаємодії ORB* для взаємодії в спеціалізованих мережевих середовищах.

## CORBAServices

Сервіси CORBA – набір служб системного рівня, запакованих разом з інтерфейсами IDL. Використовуються для створення компонентів, іменування і впровадження їх у середовище. OMG опублікувала стандарти п’ятнадцяти сервісів:

1. **Сервіс Життєвого Циклу (Life Cycle Service)**
2. **Сервіс Довгострокового Зберігання (Persistence Service)**
3. **Сервіс Найменування (Naming Service) –** дозволяє компонентам знаходити інші компоненти на шині по їх іменам
4. **Сервіс Подій (Event Service)**
5. **Сервіс Контролю Сумісного Доступу (Concurrency Control Service)**
6. **Сервіс Транзакцій (Transaction Service)**
7. **Сервіс Відносин (Relationship Service)**
8. **Сервіс Зовнішнього Представлення(Externalization Service)**
9. **Сервіс Запитів(Query Service)**
10. **Сервіс Ліцензування(Licensing Service)**
11. **Сервіс Властивостей(Properties Service)**
12. **Сервіс Часу(Time Service)**
13. **Сервіс Безпеки(Security Service)**
14. **Сервіс Комерції або Трейдер Сервіс(Trader Service)**
15. **Сервіс Контейнерів(Collection Service).**

## Object Reference

Об’єктне посилання – посилання на об’єкт надає всю необхідну інформацію, що унікально визначає об’єкт у розподіленому середовищі, та являється унікальним іменем або ідентифікатором. Реалізація об’єктного посилання не визначена специфікаціями CORBA, і тому залежить від реалізації системи. Крім того, існують *Переносні або Інтероперабельні Об’єктні Посилання* *(IOR – Interoperable Object Reference),* які призначені для передачі об’єктних посилань між різними ORB.

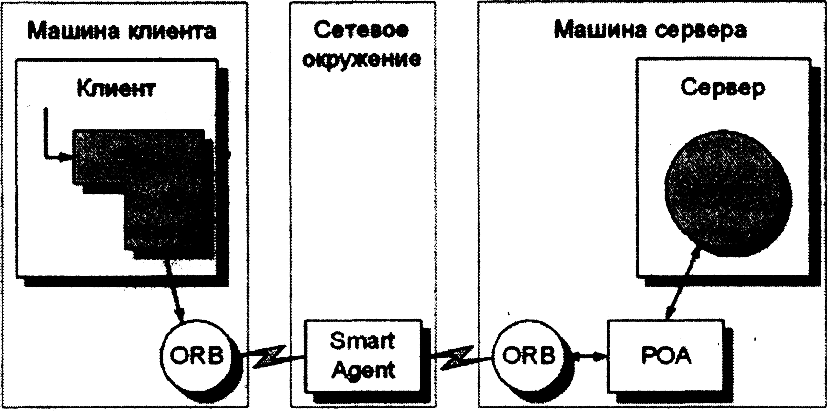
Об’єктні адаптери – це об’єкти, що розміщуються між клієнтом і сервером, для управління доступом до розподіленого об’єкта.

## BOA – BasicObjectAdapter- Базовий об’єктний адаптер

## POA – PortableObjectAdapter - Переносимий об’єктний адаптер

## Servant - Функціональність об’єкта недоступна клієнту до тих пір, поки в серверному додатку на створено CORBA-об’єкта, який дозволяє отримувати доступ до методів, які визначені в IDL-інтерфейсі. Модуль створення CORBA-об’єкта називається сервантом

## На рис. 1. представлена типова архітектура розподіленої системи CORBA. Запит посилається від клієнта до серверу. Клієнт - додаток , який виконує операції над об’єктом , а реалізація об’єкта – э код и дані, які виконують цю операцію. ORB здатен виконати всі дії щодо знаходження реалізації вказаного об’єкту, підготовці даної реалізації до виконання та передачу даних запиту.



**Рис. 1. Архитектура распределенной системы CORBA.**

**Протокол IIOPNet – основа реалізації стандарту CORBA**

ПротоколIIOPNet (Internet Inter ORB Pro визначає стандарт протоколу взаємодії між ОRB(Object Request Brocer.

IIOPNet є засобом, що дозволяє програмним компонентам, побудованим на базі платформи Microsoft. NET, використовувати багато можливостей стандарту CORBA. IIOPNet розповсюджується за ліцензією LGPL.

Перевагою використання IIOPNet є наявність простих способів використання можливостей CORBA при його допомозі.

IIOPNet використовує архітектуру. NET Remoting як базову, що дозволяє використовувати протокол IIOP для маршалинга викликів методів віддалених об'єктів, розміщених не тільки на серверах CORBA, але і на серверах. NET Remoting. Крім того, при роботі з IIOPNet проявляються багато переваг архітектури Remoting в порівнянні з «рідним» середовищем CORBA. Наприклад, при роботі з IIOPNet можна і потрібно використовувати ідіоми Remoting, пов'язані з часом життя серверних об'єктів і збором сміття на сервері (пов'язані з інтерфейсом ILease і т.д.); реалізація ж подібного механізму, наприклад, в C + +, є дуже трудомісткою ( особливо в налагодженні) завданням.

IIOPNet надає два основних варіанти використання.

1. Реалізація CORBA-сервера на базі .NET. У цьому випадку використовується бібліотека IIOPChannel:

using Ch.Elca.Iiop;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using System.Threading;

* оголошується інтерфейс з методами мовою IDL, наприклад Icoob
* створюється клас реалізації методів інтерфейсу, наприклад coob
* створюється серверний канал IIOPChannel:

port = Int32.Parse(tPort.Text);

IiopChannel chan = new IiopChannel(port);

* реєструеться канал:  
   ChannelServices.RegisterChannel(chan, false);
* створюється об'єкт:

coob ob = new coob();

* публікується об'єкт

RemotingServices.Marshal(ob, "triangle");

Після цього сервер може використовуватися з додатків, написаних на будь-яких мовах, підтримуваних виробниками CORBA ORB. Для такого використання необхідно, щоб малися опису IDL, відповідні публікованим інтерфейсам сервера. Ці описи можуть бути створені вручну (з подальшою трансляцією в. NET проксі-збірку), або згенеровані з публікованих. NET інтерфейсів розроблювального серверного додатка.

1. Реалізація CORBA-клієнта на базі . NET. На клієнті виконуються наступні дії:

using System.Runtime.Remoting;

using Ch.Elca.Iiop;

using System.Runtime.Remoting.Channels;

using System.Threading;

* Створюється клієнтській канал IIOPChannel:

IiopClientChannel Сchannel = new IiopClientChannel();

* реєструеться канал:  
   ChannelServices.RegisterChannel(Сchan, false);
* Створюється об'єкт:

host = tHost.Text;

port = tPort.Text;

!!!!! Icoob clo = (Icoob )RemotingServices.Connect(typeof(Icoob), "iiop://" + host + ":" + port + "/triangle");

* Виконується звернення до до методів

res = clo.Perimetr(a, b, c);

За допомогою бібліотеки IIOPChannel, що входить до складу IIOPNet реєструється канал підтримки IIOP; після цього можна використовувати сервер CORBA, що працює на будь-якій платформі також прозоро, як це відбувається при застосуванні Remoting. При цьому необхідно, щоб розробляється посилалося на збірку з проксі-класами, скомпільовану за допомогою IIOPNet з даних IDL-файлів CORBA-сервера.

Для реалізації описаних вище варіантів використання потрібно написати всього декілька рядків коду (як це буде видно в прикладах далі).

Для опису інтерфейсів сервера в рамках технології CORBA використовується мова OMG IDL. Для опису інтерфейсів в додатках на базі. NET використовуються власне інтерфейси. NET, інформація про які доступна у формі метаданих. Хороша сторона IIOPNet полягає в тому, що цей засіб містить інструменти, які дозволяють перетворювати інформацію про інтерфейси з одного виду в інший( наприклад из IDL- представлення до C# - представлення та навпаки). Таким чином, для кожного з описаних вище варіантів використання в залежності від характеру первинної інформації про інтерфейси потрібно використовувати один засобів - CLSToIDLGenerator або IDLToCLSCompiler.

# Практична реалізація клієнт-серверного додатка на прикладі IIOP.NET(InternetInter-ORB Protocol)

# Підготовчий етап.

# Створення збірки IIOPChannel.dll

Для створення бібліотеки IIOPChannel.dll (реалізація CORBA ) необхідно скачати із сайту [**http://sourceforge.net/projects/iiop-net/програмний**](http://sourceforge.net/projects/iiop-net/програмний) **код бібліотеки (файл IIOPNet.src.1.9.3.zip)**

та скомпілювати у VisualStudio 2010. В результаті в папці d:\IIOPNet.src.1.9.3\bin\Debug\ буде створено файл **IIOPChannel.dll,** який містить системне програмне забезпечення технології CORBA. Саме на цю бібліотеку в подальшому (під час створення сервера та клієнта) мають бути створені посилання ( References)

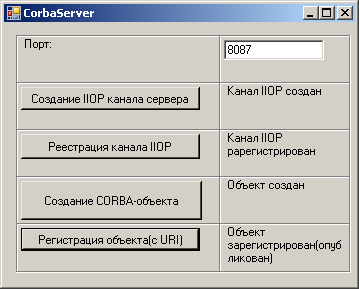
В збірці реалізовано клас **IiopChannel,** що реалізує три інтефейси: **IChannel, IChannelReceiver, IChannelSender**, - із збірки **System.Runtime.Remoting.Channels.**

Окремо реалізовані класи для клієнтських і серверних каналів, які реалізують інтерфейси IChannelSender та IChannelReceiver відповідно. IIOPNet активно використовує засоби Remoting для реалізації своєї функціональності. Для використання протоколу IIOP необхідно створити та зареєструвати канал IIopChannel.

**Приклад системи клієнт – сервер з методом звернення до базі даних**

**СОRBA – SERVER**

Форма сервера після активізації



**Оголошення інтерфейсу**

using System;

using System.Runtime.Remoting;

using System.Drawing;

using System.Data;

using Ch.Elca.Iiop.Idl;

using System.IO;

namespace CorbaDemo

{

public interface ITriangle

{

double Perimetr(Point a, Point b, Point c);

double Square(Point a, Point b, Point c);

// byte[] GetTable(string name);

MemoryStream GetTable(string name);

string GetDataSet(string name);

}

}

**Клас реалізації**

using System;

using System.Drawing;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using System.Data;

using System.Data.OleDb;

using System.Data.Sql;

using System.IO;

using System.Data.SqlClient;

using System.Windows.Forms;

using System.Runtime.Serialization;

namespace CorbaDemo

{

public class Triangle: MarshalByRefObject, ITriangle

{

private string \_name;

public override object InitializeLifetimeService()

{

return null;

}

private double Length(Point a, Point b)

{

return Math.Sqrt(Math.Pow(a.X - b.X, 2.0) + Math.Pow(a.Y - b.Y, 2.0));

}

public double Perimetr(Point a, Point b, Point c)

{

return Length(a, b) + Length(b, c) + Length(c, a);

}

public double Square(Point a, Point b, Point c)

{

double p = Perimetr(a, b, c)/2.0;

return Math.Sqrt(p \* (p - Length(a, b)) \* (p - Length(b, c)) \* (p - Length(c, a)));

}

public MemoryStream GetTable(string name)

{

//MessageBox.Show("Call getadotable bt");

String connstr = @"Data Source=R501N1\SQLEXPRESS;Initial Catalog=MyTestDB;Integrated Security=True";

SqlCommand Ocmd; SqlConnection Conn;

Conn = new SqlConnection(connstr);

Ocmd = Conn.CreateCommand();

Ocmd.CommandText = name;

Ocmd.CommandType = CommandType.Text;

Conn.Open();

SqlDataAdapter da = new SqlDataAdapter(Ocmd.CommandText, Conn);

// MessageBox.Show("befor WriteXml");

MemoryStream strm=new MemoryStream();

DataSet ds = new DataSet();

da.Fill(ds);

ds.WriteXml(strm, XmlWriteMode.WriteSchema);

// MessageBox.Show("befor create byte ");

return strm;

}

public string GetDataSet(string name)

{

DataSet ds = null;

OleDbConnection Conn;

string strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\\ASOER\_CB\\ASOER\_DB\\ASOER\_ML.mdb";

Conn = new OleDbConnection(strConnect);

try

{

Conn.Open();

string strSQL = name;

\_name = name;

OleDbDataAdapter sda = new OleDbDataAdapter(strSQL,Conn);

ds = new DataSet();

sda.Fill(ds,"Table1");

}

finally

{

Conn.Close();

}

MessageBox.Show(ds.GetXml());

return ds.GetXml();

}

}

}

**Код Сервера**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Ch.Elca.Iiop;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using System.Threading;

namespace CorbaDemo

{

public partial class CorbaServer : Form

{

private IiopChannel chan=null;

private int port = 8087;

private Triangle Otriangle = null;

public CorbaServer()

{

InitializeComponent();

}

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

{ // створення канала

try

{

port = Int32.Parse(tPort.Text);

chan = new IiopChannel(port);

lCreate.Text = "Канал IIOP cоздан";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message,"Ошибка",MessageBoxButtons.OK,MessageBoxIcon.Error);

}

}

private void button2\_Click(object sender, EventArgs e)

{ // реєстрація канала

try

{

ChannelServices.RegisterChannel(chan, false);

lRegister.Text = "Канал IIOP pарегистрирован";

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{ Otriangle = new Triangle();

lServant.Text = "Объект создан ";

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button4\_Click(object sender, EventArgs e)

{

try

{ RemotingServices.Marshal(Otriangle, "triangle");

lMarshal.Text = "Объект зарегистрирован(опубликован)";

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

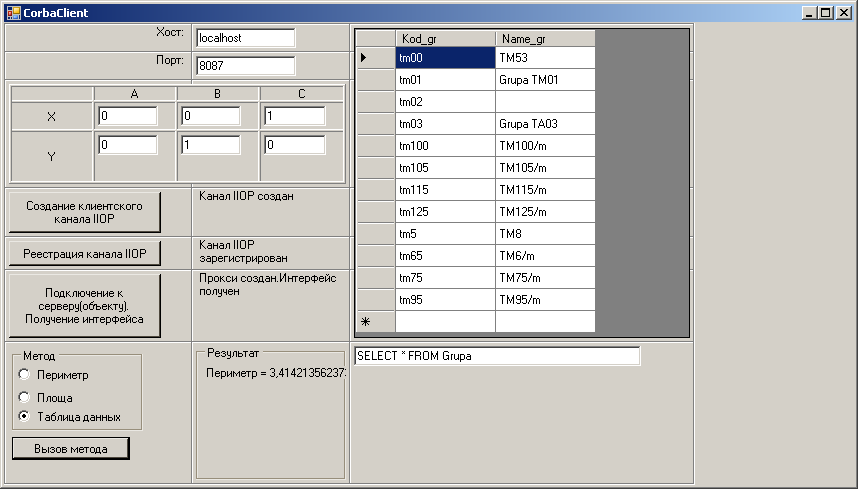
}

}

}

**CORBA – КЛІЄНТ**

**Форма клієнта з результатами виконання**



**Оголошення інтерфейсу**

using System;

using System.Runtime.Remoting;

using System.Drawing;

using System.Data;

using System.IO;

using Ch.Elca.Iiop.Idl;

namespace CorbaDemo

{

Public interface ITriangle

{

double Perimetr(Point a, Point b, Point c);

double Square(Point a, Point b, Point c);

// byte[] GetTable(string name);

MemoryStream GetTable(string name);

string GetDataSet(string name);

}

}

**Код клієнта**

**!!!! Перед записом отриманого від сервера XML до файлу d:\clientfstr.txt", необхідно його очисти !!!!**

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using CorbaDemo;

using System.IO;

using Ch.Elca.Iiop;

namespace CorbaClient

{

public partial class CorbaClient : Form

{

private string host="localhost", port="8087",paramStr="";

private IiopClientChannel channel;

private ITriangle Itriangle;

private Point a, b, c;

public CorbaClient()

{

InitializeComponent();

}

private void GetPoints()

{

try

{

a.X = Int32.Parse(tAx.Text); a.Y = Int32.Parse(tAy.Text);

b.X = Int32.Parse(tBx.Text); b.Y = Int32.Parse(tBy.Text);

c.X = Int32.Parse(tCx.Text); c.Y = Int32.Parse(tCy.Text);

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

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

{ // створення канала

try

{

channel = new IiopClientChannel();

lCreate.Text = "Канал IIOP создан";

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

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

{ // реєстрація канала

try

{

ChannelServices.RegisterChannel(channel, false);

lRegister.Text = "Канал IIOP зарегистрирован";

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

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

{

try

{ // отримання об’єкту

host = tHost.Text;

port = tPort.Text;

Itriangle = (ITriangle)RemotingServices.Connect(typeof(ITriangle), "iiop://" + host + ":" + port + "/triangle");

lConnect.Text = "Прокси создан.Интерфейс получен";

}

catch (Exception ex)

{ MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

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

{

double res;

DataTable dt;

DataSet ds = new DataSet();

GetPoints();

try

{

if (radioButton1.Checked == true)

{

res = Itriangle.Perimetr(a, b, c);

lResult.Text = String.Format("Периметр = {0} ед.", res);

}

else

if (radioButton2.Checked)

{

res = Itriangle.Square(a, b, c);

lResult.Text = String.Format("Площадь = {0} кв.ед.", res);

}

else

{

paramStr = tbParam.Text;

var ms = new MemoryStream(1000);

// MessageBox.Show("befor call server");

var charms = new char[1000] ;

string rs = "start";

charms = rs.ToCharArray();

ms= Itriangle.GetTable(paramStr);

int l= Convert.ToInt32(ms.Length);

byte[] bt = ms.ToArray();

int l1 = bt.Length;

// MessageBox.Show("ms.Length is: "+l.ToString()+"bt.Length is:"+l1.ToString()+", now prepare start elemet");

string start = "<?xml version=\"1.0\" standalone=\"yes\"?>\r\n";

int lstart = start.Length;

char[] chars = start.ToCharArray();

byte[] sbt = new byte[100];

for ( int k=0;k<lstart;k++)

{

sbt[k] = Convert.ToByte(chars[k]);

}

FileMode fm = new FileMode();

fm = FileMode.Open;

FileStream fstm = new FileStream(@"d:\clientfstr.txt", fm);

fstm.Write(sbt,0,lstart);

fstm.Write(bt, 0, l1);

fstm.Close();

// MessageBox.Show("after call server, 1="+ m[1].ToString()+", 2="+m[2].ToString());

ds.ReadXml(@"d:\clientfstr.txt", XmlReadMode.ReadSchema);

dataGridView1.DataSource = ds.Tables[0];

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка вызова метода, либо серванта", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}