**Лекция 12. Технология Remoting. Объект активизируемый клиентом**

Типовая архитектура трехзвенной системы с отдельно создаваемым Remoting-объектом в виде библиотеки, Remoting сервером, предоставляющим объект клиентам, Remoting клиентом, получающим обект и сервером бд

Сервер БД

Реализация запросов

1.Создание канала

2.Регистрация канала

3.Регистрация объекта

Remoting-сервер

Remoting-клиент

1. Регистрация объекта
2. Создание объекта
3. Вызов методов

Remoting-объект

С методами и реализацией

# Создание объекта

Объект можно создать как обычную dll –библиотеку классов(Class Library), в которой создается класс, содержащий реализацию методов предназначенных для вызова клиентами. Имя класса – имя remoting – объекта

В зависимости от способа предоставления объекта клиентам имеются два варианта объявления класса:

**1 вариант: Передача объекта по ссылке**

public class RemOb1 : System.MarshalByRefObject

{

public RemOb1()

{

// MessageBox.Show("Constructor RemOb1 called");

}

public DataTable GetADOTable(string name)

{ String connstr = " Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\\ASOER\_CB\\ASOER\_DB\\ASOER\_ML.mdb";

DataTable restab;

OleDbCommand Ocmd; OleDbConnection Conn;

// DataSet XMLDataSet = new DataSet(); // DataRow dr;

Conn = new OleDbConnection(connstr); Ocmd = Conn.CreateCommand();

Ocmd.CommandText = name; Ocmd.CommandType = CommandType.Text;

Conn.Open();

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

restab = new DataTable();

da.Fill(restab);

// MessageBox.Show(" Metod GetADOTable called");

return (restab);

}

}

**2вариант: Передача объекта по значению( путем сериализации)**

[Serializable]

public class RemOb1

{

public RemOb1()

{ // MessageBox.Show("Constructor RemOb1 called"); }

public DataTable GetADOTable(string sqlname)

{ // аналогичная реализация метода }

}

# Серверное приложение

Объекты, активизируемые клиентом (Client-activated objects, CAO)

Объекты, активируемые клиентом, — это объекты, это объекты на серверной стороне, активизируемые по запросу клиента время существования которых не управляется сервером. Способ активизации этих серверных объектов во многом аналогичен активизации соклассов в классической модели COM. Когда клиент с помощью оператора new отправляет запрос к серверному объекту, удаленному приложению посылается сообщение с запросом на активизацию. Затем сервер создает экземпляр запрашиваемого класса и возвращает ObjRef вызывающему клиентскому приложению. Далее на клиентской стороне по полученному ObjRef создается прокси, через который клиент будет вызывать методы удаленного объекта. Объекты, активизируемые клиентом, могут сохранять информацию о состоянии между вызовами методов для конкретного клиента, но не для разных клиентских объектов. **Каждый вызов new возвращает прокси для независимого экземпляра типа, существующего на сервере.**

На сервере сначала необходимо создать и регистрировать канал доступа до объекта, например

// Создаем Tcp или Http канал channel

channel = new TcpChannel(888); // вариант HttpChannel(888)

// Регистрируем канал channel;

ChannelServices.RegisterChannel(channel, false);

Затем, для создания объекта активируемого сервером его необходимо зарегистрировать, а именно:

// регистрируем со стороны сервера RemOb1 объект как объект типа Activated( управляемый клиентом, без URI)

RemotingConfiguration.RegisterActivatedServiceType(typeof(RemOb1));

**Полное описание метода**

# Метод RemotingConfiguration.RegisterActivatedServiceType (Type)

Регистрирует указанный тип объектов, зарегистрированных со стороны службы, в качестве типа, который может быть активирован по запросу клиента.

**Список перегрузки**

|  |  |
| --- | --- |
| Имя | Описание |
| [RegisterActivatedServiceType](http://msdn.microsoft.com/ru-ru/library/bsz90thf.aspx)  [(ActivatedServiceTypeEntry)](http://msdn.microsoft.com/ru-ru/library/bsz90thf.aspx) | Регистрирует тип объекта предоставленный параметром [ActivatedServiceTypeEntry](http://msdn.microsoft.com/ru-ru/library/system.runtime.remoting.activatedservicetypeentry.aspx) в качестве типа, который может активироваться по запросу клиента. |
| [RegisterActivatedServiceType (Type)](http://msdn.microsoft.com/ru-ru/library/93z65zdc.aspx) | Регистрирует указанный тип объектов, зарегистрированных со стороны службы, в качестве типа, который может быть активирован по запросу клиента. |

Параметры type Тип: [System.Type](http://msdn.microsoft.com/ru-ru/library/system.type.aspx) тип объекта для регистрации.

**Исключения** : [SecurityException](http://msdn.microsoft.com/ru-ru/library/system.security.securityexception.aspx) По крайней мере у одного из вызывающих операторов, находящихся в верхней части стека вызовов, отсутствует разрешение на настройку типов и каналов удаленного взаимодействия.

**Использование конфигурационного файла**  позволяет администратору настраивать поведение приложения без необходимости перекомпилировать его. Но самое главное, можно заменить весь предыдущий код одной строкой:

{

**RemotingConfiguration.Configuref @"...\JobServer.exe.config" );**

}

ПРИМЕЧАНИЕ По соглашению имя конфигурационного файла образуется путем добавления к имени исполняемого файла приложения строки «.config».

Ниже показан конфигурационный файл JobServer.exe.config:

<configuration>

<system.runtime.remoting>

Application name="JobServer">

<service>

type="JobServerLib . JobServerlmpl , JobServerLib "

</service>

<channels>

<channel ref="http"

port="4000" />

</channels>

</application>

</system.runtime.remoting>

</configuration>

Обратите внимание на то, как соотносится содержимое конфигурационного файла и код удаленного взаимодействия, добавленный ранее. Элемент< channel /> содержит ту же самую информацию для настройки канала, которая была использована в первоначальном варианте программной конфигурации. Информация для регистра ции объектов с серверной активизацией заключена в элементе <service>. И элемент <service>, и элемент <channels> могут одержать несколько вложенных элементов. Как видно из данного примера, возможности конфигурационных файлов просто поразительны

**Клиентское приложение**

Чтобы создать экземпляр типа, активируемого сервером, клиентские приложения должны быть настроены программным образом или с помощью файла конфигурации. В случае настройки приложения программным образом независимо от способа активации имеется два способа получения объекта

**1 способ – с предварительной регистрацией Activated – объекта со стороны клиента**

Сначала проводится регистрация методом

RemotingConfiguration.RegisterActivatedClientType(typeof(имя\_объекта),"протокол://адрес:порт");

Object obj = null;

obj = new имя объекта();

**Пример:**

RemotingConfiguration.RegisterActivatedClientType(typeof(RemOb1), "TCP://localhost:888/");

**Создание объекта для вызова методов:**

RemOb1obj = null;

obj = new RemOb1();

Использование объекта для вызова методов

DataTable tb = new DataTable();

String SqlSt;

SqlSt = this.textBox1.Text;

tb = obj.GetADOTable(SqlSt);

# Полное описание метода регистрации

Метод RemotingConfiguration.RegisterActivatedClientType (ActivatedClientTypeEntry)

Регистрирует объект [Type](http://msdn.microsoft.com/ru-ru/library/system.type(v=VS.90).aspx) со стороны клиента в качестве типа, который может активироваться сервером.

Список перегрузки

|  |  |
| --- | --- |
| Имя | Описание |
| [RegisterActivatedClientType](http://msdn.microsoft.com/ru-ru/library/8kz5f4f8(v=VS.90).aspx)  [(ActivatedClientTypeEntry)](http://msdn.microsoft.com/ru-ru/library/8kz5f4f8(v=VS.90).aspx) | Регистрирует объект [Type](http://msdn.microsoft.com/ru-ru/library/system.type(v=VS.90).aspx), предоставленный [ActivatedClientTypeEntry](http://msdn.microsoft.com/ru-ru/library/system.runtime.remoting.activatedclienttypeentry(v=VS.90).aspx) на стороне клиента в качестве типа, который может быть активирован сервером по запросу клиента. |
| [RegisterActivatedClientType(Type, String)](http://msdn.microsoft.com/ru-ru/library/hkzabx8b(v=VS.90).aspx) | Регистрирует объект [Type](http://msdn.microsoft.com/ru-ru/library/system.type(v=VS.90).aspx) со стороны клиента в качестве типа, который может активироваться сервером по зопросу клиента, используя данные параметры Type и String |

Регистрирует объект [Type](http://msdn.microsoft.com/ru-ru/library/system.type(v=VS.90).aspx), записанный в предоставленный [ActivatedClientTypeEntry](http://msdn.microsoft.com/ru-ru/library/system.runtime.remoting.activatedclienttypeentry(v=VS.90).aspx) на стороне клиента в качестве типа, который может быть активирован сервером.

Параметры entry Тип: [System.Runtime.Remoting.ActivatedClientTypeEntry](http://msdn.microsoft.com/ru-ru/library/system.runtime.remoting.activatedclienttypeentry(v=VS.90).aspx)  
Параметры конфигурации для активированного клиентом типа.

Исключения : [SecurityException](http://msdn.microsoft.com/ru-ru/library/system.security.securityexception(v=VS.90).aspx) По крайней мере у одного из вызывающих операторов, находящихся в верхней части стека вызовов, отсутствует разрешение на настройку типов и каналов удаленного взаимодействия.

**2 способ – без предварительной регистрации**

В этом случае для создания на клиенте экземпляра объекта, активируемого сервером, используется метод Activator.CreateInstance

В следующем примере кода показан вызов метода Activator.CreateInstance

// получаем Activated удаленный объект сервера без регистрации

obj = null;

// вариант1

this.obj = (RemOb1)Activator.CreateInstance<RemOb1>();

// вариант2

this.obj = (RemOb1)Activator.CreateInstance (typeof(RemOb1), "TCP://localhost:888/");

if (obj == null)

{

this.label7.Text = "Сервер не определяется, объект не создан";

return;

}

else

{

this.label7.Text = "Сервер определяется, Activated объект создан успешно , без оегистрации";

**Вызов методов**

После получения объекта используя точечный синтаксис можно вызывать его методы, например

DataTabe dt = new DataTable();

String sqls = @”select \* from Personal”

dt = obj.GetADOTable(sqls)

**Использование конфигурационного файла для клиентского приложения**

Второй метод, предоставляемый .NET Remoting для настройки параметров удаленного взаимодействия, использует конфигурационные файлы. Ранее было рассмотрено использовании конфигурационных файлов в разделе «Реализация приложения JobServer». При настройке клиентского приложения применяются другие теги. Вот так выглядит XML-код конфигурационного файла приложения JobClient:

<configuration>

<system.runtime.remoting>

<Application name="JobCHent">

<client>

<Activated

type="JobserverLib.JobServerlmpl, JobServerLib"

url="http://localhost:400>

</client>

<channels>

<channel ref="http" port="0" />

</channels>

</application>

</system.runtime.remoting>

</configuration>

Использование файла JobClient.exe.config для настройки удаленного взаимодействия путем вызова RemotingConfiguration.Configure:

private IJobServe r GetJobServerQ

{

RemotingConfiguration.Configure(@"..\..\JobClient.exe.config"

return (IJobServer)new JobServerImpl;

};