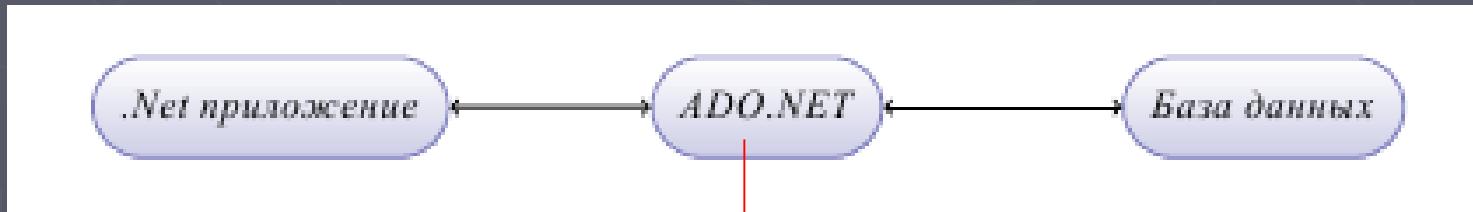


Работа с базами данных

ADO.Net Core

ADO.NET Core

интерфейс прикладного уровня Microsoft
(набор классов, предоставляющих
службы доступа к данным) для работы с
реляционными базами данных



WinForm, WPF, ASP.Net

Встроенные провайдеры :
для MS SQL Server
для OLE DB (Access, DB2, MySQL и Oracle)
для ODBC
для Oracle
И др.

Режимы ADO.NET

- ▶ отсоединенные данные

Disconnected Environment

- ▶ постоянное подключение

Connected Environment

Постоянное (активное)подключение

- ▶ Установка соединения
- ▶ Подготовка и выполнение команды
- ▶ Работа с данными
 - ▶ чтение, запись
 - ▶ фильтрация, сортировка
 - ▶ тоже в пакетном режиме
 - ▶ блокировки, совместное использование
- ▶ Закрытие соединение и обработка ошибок

▶ Работает в режиме удержания подключения к базе.
▶ Обеспечивает максимальную гибкость и эффективность.
▶ Обеспечивает минимальный расход оперативной памяти.

Отсоединенные (автономные) данные

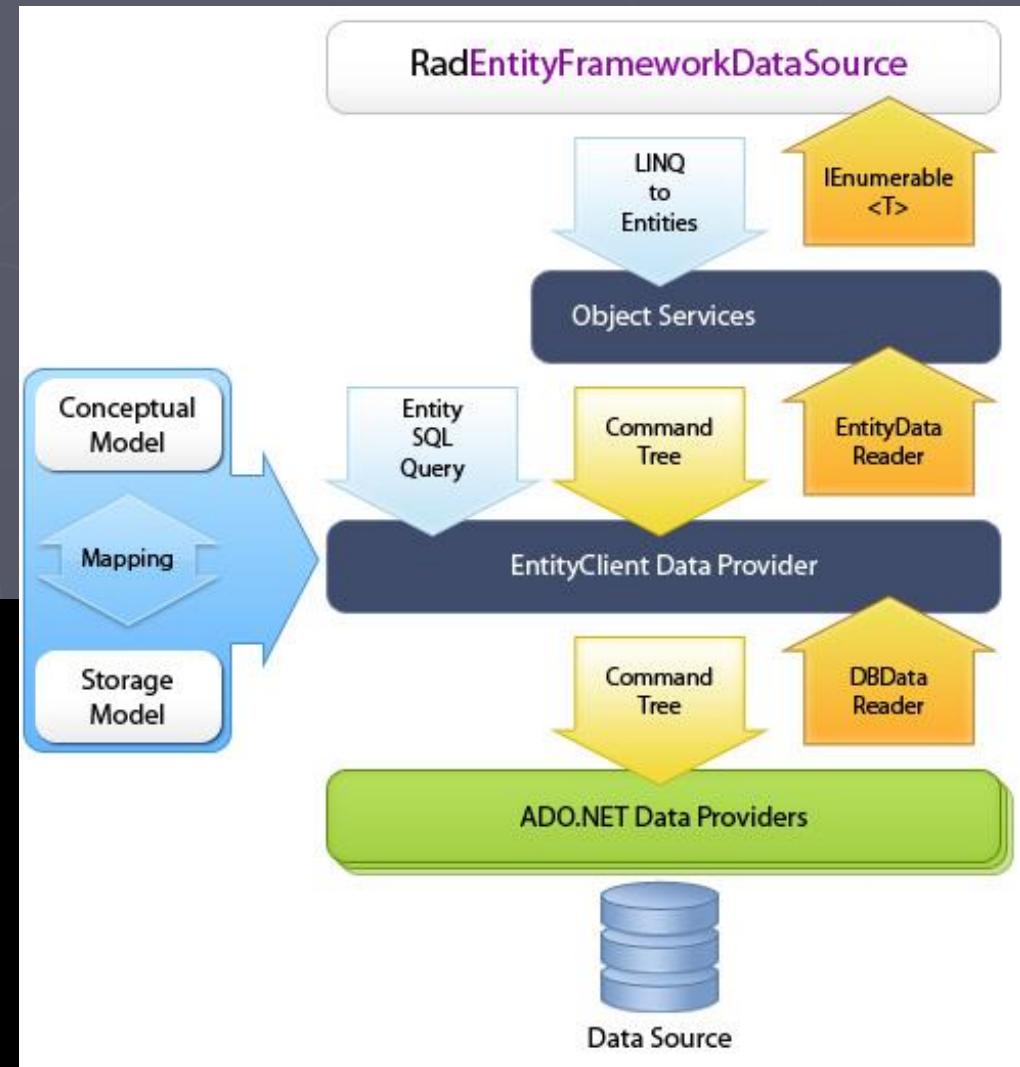
- ▶ Загрузка данных с сервера;
- ▶ Изменение данных в наборе на локальной машине;
- ▶ Обновление данных на сервере на основе локальной копии.

- ▶ Обеспечивает работу с данными в отсутствии подключения к БД.
- ▶ Удобна для переноса данных по сети.
- ▶ Расходует достаточно много памяти

Entity Framework

► объектно-реляционное отображение (object-relational mapping — ORM)

Тяжеловесен
Проще писать и сопровождать.
Сокращает время разработки.



Классы ADO.NET

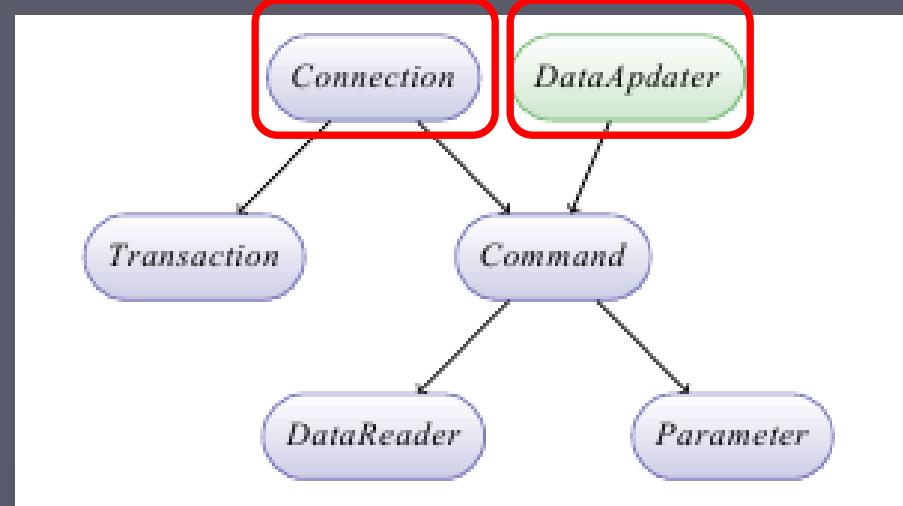
классы объектов-
провайдеров(поставщиков) данных .NET

специфичны для каждого типа источников
данных

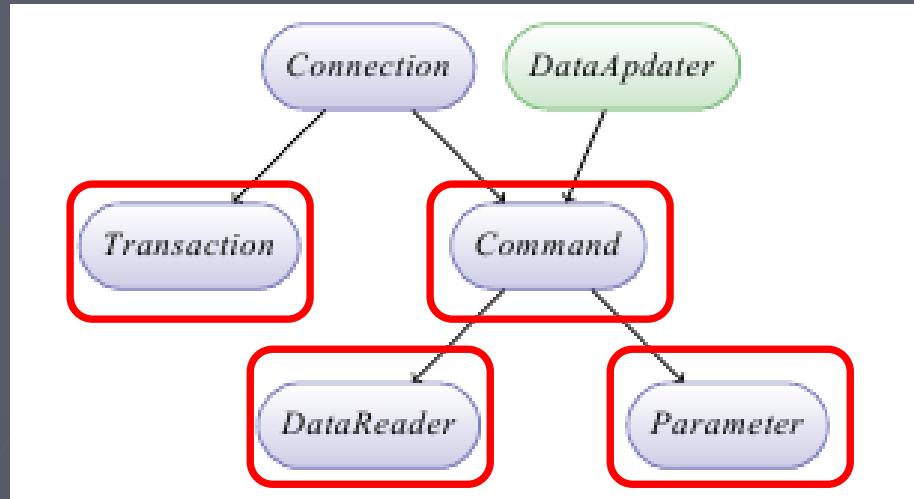
классы объектов потребителей.

для доступа и манипуляции данными после того, как они
прочитаны в память.

Объекты-провайдер ADO.NET



- ▶ **DataAdapter** - шлюз между автономными и подключенными аспектами ADO.NET. (**DataAdapter** - **SqlDataAdapter**, **OracleDataAdapter**)
Содержит экземпляр соединения и команды.
- ▶ **Connection**. Применяется для создания канала связи между программой и источником данных. Он позволяет устанавливать строку подключения, управлять транзакциями и устанавливать тип курсора (серверные и клиентские курсоры)



Command - класс представляющий исполняемую команду в базовом источнике данных (произвольные SQL-строки или хранимых процедур, поддерживает параметры)

DataReader - эквивалент конвейерного курсора с возможностью только чтения.

Transaction – объект транзакций(OleDbTransaction, SqlTransaction, OracleTransaction) System.Transaction

Parameter - объект параметр команды

провайдеры данных:

Извлечение данных из ист., обновление данных

- ▶ – ODBC.NET Data Provider
- ▶ – SQL Server .NET Data Provider – СУБД Microsoft Sql Server;
- ▶ - Oracle Data Provider
- ▶ – OleDb .NET Data Provider – осуществляет взаимодействие с БД других типов.



Connection, Command, DataReader, DataAdapter

Основные классы провайдера

SqlConnection и **OleDbConnection** –

обеспечивают подключение к БД.

SqlCommand и **OleDbCommand** **OdbcCommand** –

управляют источником данных с помощью SQL.

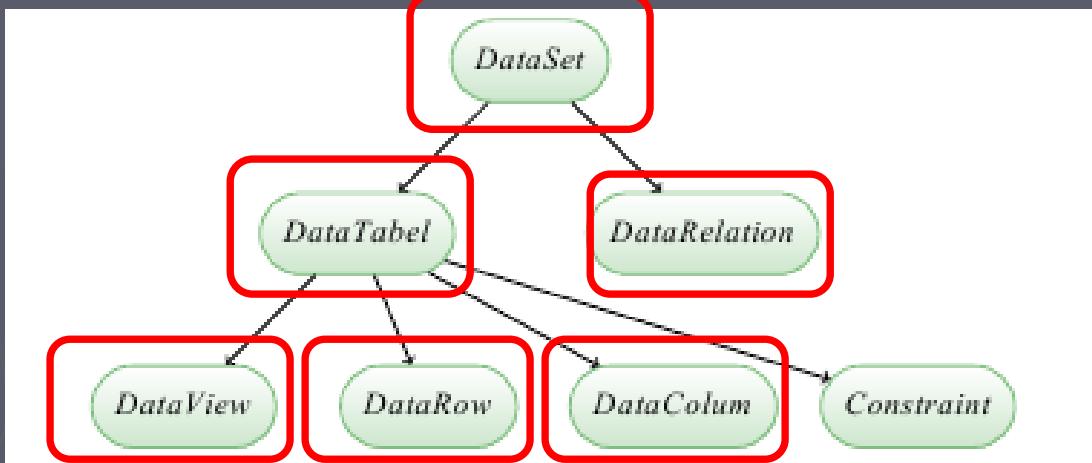
SqlDataReader и **OleDbDataReader** **OdbcDataReader** –

обеспечивают последовательный доступ к результату выполнения команды Select. Для работы требуют монопольный доступ к соединению.

SqlDataAdapter и **OleDbDataAdapter** **OdbcDataAdapter** –

заполняют отсоединеный объект DataSet или обновляют данные на сервере из DataSet.

System.Data Объекты-потребители (Автономные объекты) ADO.NET



DataSet - ядро автономного режима доступа к данным в ADO.NET (своя маленькая СУБД, полностью находящаяся в памяти)

DataTable - класс похож на таблицу БД. Он состоит из объектов *DataColumn*, *DataRow*, представляющих из себя строки и столбцы.

DataRelation - класс позволяет задавать отношения между различными таблицами, с помощью которых можно проверять соответствие данных из различных

таблиц.

DataView - объект представлений базы данных

Constraint— ограничение на один или несколько столбцов в локальной таблице, служащее для поддержания целостности данных.

NuGet — решение MainWindow.xaml MainWindow.xaml.cs AssemblyInfo.cs

Обзор Установлено Обновления Консолидировать Управление пакетами для

data.sql x ↻ Включить предварительные версии Источник пакета: nuget.org

 System.Data.SqlClient автор: Microsoft v4.8.2	 System.Data.SqlClient
.NET Provides the data provider for SQL Server. These classes provide access to versi...	Версии — 1
<input checked="" type="checkbox"/> Проект	
<input checked="" type="checkbox"/> AppAdo	4.8.2

Все пакеты лицензируются их владельцами. NuGet не несет ответственности за пакеты сторонних производителей и не предоставляет лицензии на такие пакеты.

<w:Desktop">

Добавить подключение

?

X

Введите данные для подключения к выбранному источнику данных или нажмите кнопку "Изменить", чтобы выбрать другой источник данных и (или)

Источник данных:

Файл базы данных Microsoft SQL Server (SqlClient)

Изменить...

Имя файла базы данных (новой или существующей):

Обзор...

Вход на сервер

 Использовать проверку подлинности Windows Использовать аутентификацию SQL Server

Имя пользователя:

Пароль:

 Сохранить пароль

Дополнительно...

Проверить подключение

OK

Отмена



- ▲ Azure (Подключение не установлено)
- ▷ Data Lake Analytics
- ▷ HDInsight
- ▷ Stream Analytics
- ▷ Базы данных SQL
- ▷ Виртуальные машины
- ▷ Облачные службы (классические)
- ▷ Служба приложений
- ▷ Хранилище
- ▷ Центры уведомлений
- ▲ Подключения данных
 - ▷ comuper.mdf
 - ▲ Серверы
 - ▷ DESKTOP-8K3SG88

dbo.Study [Конструктор]*

dbo.Compy [Конструктор]*

MainWindow.xaml

MainWindow.xaml.cs

Обновить

Файл скрипта:

dbo.Table_1.sql*

Имя	Тип данных	Допустимы значения NULL	По умолчанию
Idcompy	uniqueidentifier	<input type="checkbox"/>	NEWID()
company	nchar(10)	<input checked="" type="checkbox"/>	
processor	nchar(10)	<input checked="" type="checkbox"/>	

ючи (1)

<Без имени> (Первичный ключ, Clu

оверочное ограничение (0)

дексы (0)

ешние ключи (0)

иггеры (0)

Проектирование

T-SQL

```
1 CREATE TABLE [dbo].[Compy]
2 (
3     [Idcompy] UNIQUEIDENTIFIER NOT NULL PRIMARY KEY DEFAULT NEWID(),
4     [company] NCHAR(10) NULL,
5     [processor] NCHAR(10) NULL
6 )
7
8
```

Предварительный просмотр обновлений базы данных

?

Выделение

Нет

Действия пользователя

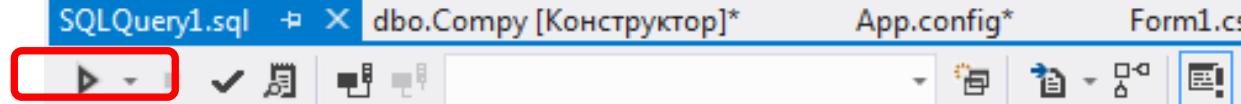
Создать

[dbo].[Table] (Таблица)

SQLQuery1.sql X dbo.Compy [Консоль]

Корректирующие действия

Her

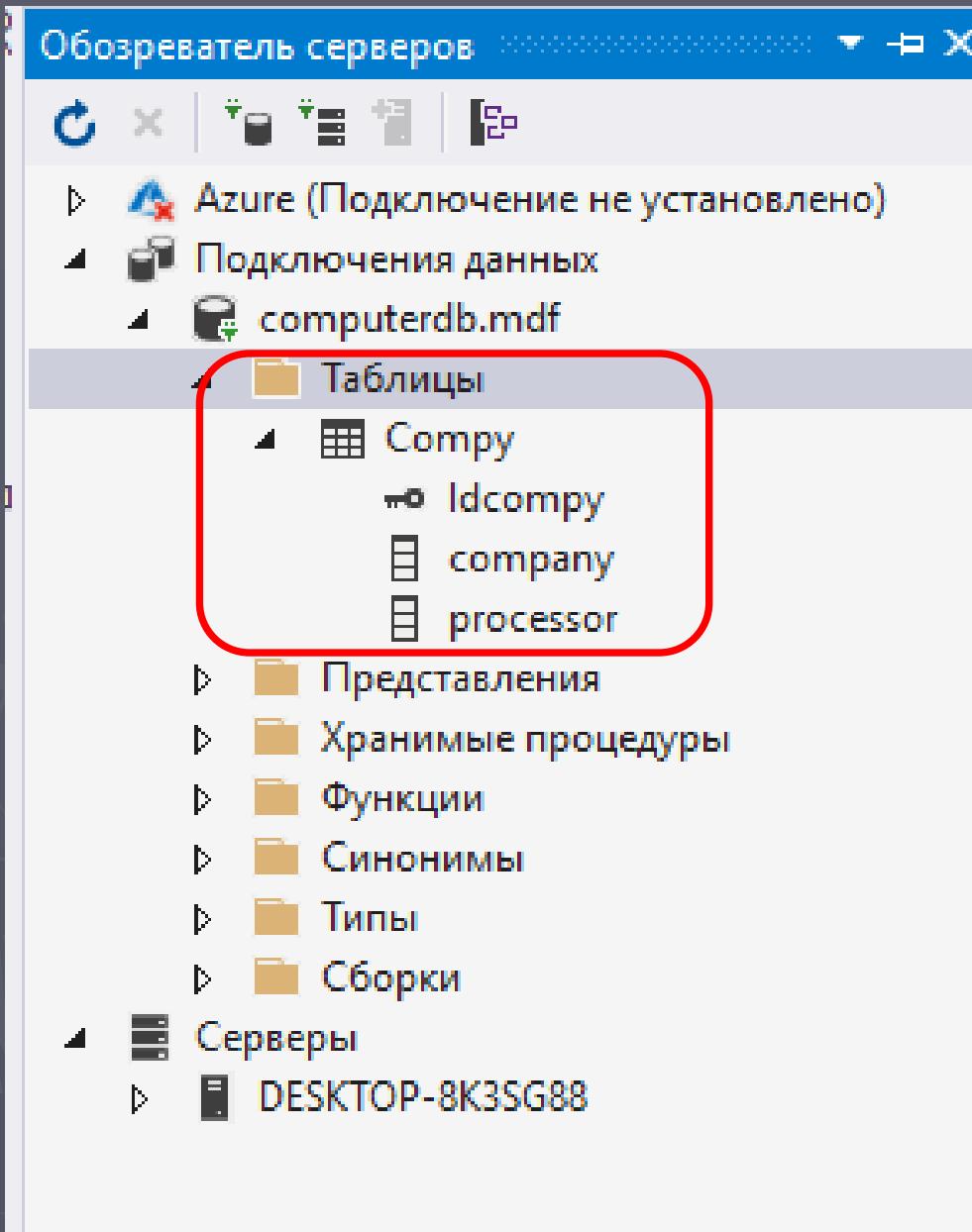


```
GO  
BEGIN TRANSACTION  
GO  
PRINT N'Выполняется создание [dbo].[Compy]...';
```

```
GO
CREATE TABLE [dbo].[Compy] (
    [Idcomp]     UNIQUEIDENTIFIER NOT NULL,
    [company]    NCHAR (10)        NULL,
    [processor]   NCHAR (10)        NULL,
    PRIMARY KEY CLUSTERED ([Idcomp] ASC)
);
```

```
GO  
IF @@ERROR <> 0  
    AND @@TRANCOUNT > 0  
        BEGIN  
            ROLLBACK;  
        END
```

TF តម្លៃ TRANCOINT = ០



dbo.Study [Конструктор]* X MainWindow.xaml MainWindow.xaml.cs AppAdo.csproj X ⚙

Обновить | Файл скрипта: dbo.Table_1.sql*

Имя	Тип данных	Допустимы значения NULL	По умолчанию
Idstud	uniqueidentifier	<input type="checkbox"/>	NEWID()
name	nvarchar(50)	<input checked="" type="checkbox"/>	
compld	uniqueidentifier	<input type="checkbox"/>	
		<input type="checkbox"/>	

Ключи (1)
«Без имени» (Первичный ключ, Cl)

Проверочное ограничение (0)

Индексы (0)

Внешние ключи (1)
FK_Table_ToCompy (Idcompy)

Тrigгеры (0)

Проектирование T-SQL

```
CREATE TABLE [dbo].[Study]
(
    [Idstud] UNIQUEIDENTIFIER NOT NULL PRIMARY KEY DEFAULT NEWID(),
    [name] NVARCHAR(50) NULL,
    [compld] UNIQUEIDENTIFIER NOT NULL,
    CONSTRAINT [FK_Table_ToCompy] FOREIGN KEY ([compld]) REFERENCES [Compy]([Idcompy])
)
```

computerdb.mdf

Таблицы

Compry

Idcompry

company

processor

Study

Idstud

name

compld

Представления

Удаленные представления

dbo.Study [Данные]

dbo.Compry [Данные]

MainWindow



Максимальное количество строк: 1000

	Idstud	name	compld
▶	10a-63ee56028371	Pavel	0d353e93-013b...
	00ff7c9e-7c85-4ac...	Dima	86f882d7-6706-...
*	NULL	NULL	NULL

dbo.Compry [Данные]

dbo.Compry [Конструктор]

10

	Idcompry	company	processor
▶	1cd-07ea543eb2dd	HP	ADM
	b98155f8-4b99-	Sony	Intel
	0d353e93-013b...	Lenovo	Athlon
*	NULL	NULL	NULL

Добавление нового элемента - AppAdo

? X

Установленные

Элементы Visual C#

WPF

Веб

Данные

Код

Общие

MySQL

SQL Server

Storm Items

Xamarin.Forms

Графика

В сети

Сортировка: По умолчанию



XML-файл

Элементы Visual C#



XSLT-файл

Элементы Visual C#



База данных на службах

Элементы Visual C#



Набор данных

Элементы Visual C#



Схема XML

Элементы Visual C#

Поиск (Ctrl+E)



Тип: Элементы Visual C#

Пустая база данных SQL Server для доступа к данным, основанного на службах

Имя:

Database1.mdf

Добавить

Отмена

1. Подключение

```
using System.Data;  
  
using System.Data.SqlClient;  
using System.Data.OleDb;  
using System.Data.Odbc;
```

- ▶ классы ADO
- ▶ сослаться на родного провайдера данных .NET

Изменение подключения

Введите данные для подключения к выбранному источнику данных или нажмите кнопку "Изменить", чтобы выбрать другой источник данных и (или) поставщик.

Источник данных:

Файл базы данных Microsoft

Имя файла базы данных (ново

computerdb.mdf

Вход на сервер

Использовать проверку

Использовать аутентиф

Имя пользователя:

Пароль:

Проверить подключение

Расширенные свойства



A
Z ↓



Type	System	Version	Latest
Инициализация			
ApplicationIntent		ReadWrite	
Asynchronous Processing		False	
Connect Timeout		30	
Current Language			
Источник			
AttachDbFilename		C:\Users\User\Documents\computerdb.mdf	
Context Connection		False	

Data Source

(LocalDB)\MSSQLLocalDB

Data Source

Указывает имя источника данных, к которому необходимо установить подключение.

Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Documents\computerdb.mdf;

OK

Отмена

SqlConnection — имя объекта соединения
для провайдера данных SQL .NET;
OleDbConnection — OLE DB

```
connectionString="Data  
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Documen  
ts\computerdb.mdf;Integrated Security=True;Connect Timeout=30"
```

имя_компьютера\имя_экземпляра.

устанавливает проверку
подлинности

Точка— ссылка на экземпляр сервера, запущенный на текущей
машине; local; действительное сетевое имя компьютера.

файл . MDF

способ входа в базу (имя пользователя и пароль)

User=admin; PWD=admin.

таймаут соединения (сек)

Пользовательский экземпляр

► using System.Configuration;

```
string connectionString =  
ConfigurationManager.ConnectionStrings["WpfDb.Properties.Settings.user  
ConnectionString"].ConnectionString;
```

► Другие параметры строки

- **AttachDBFileName**
- **Connect Timeout**
- **Data Source - Server Address Addr NetworkAddress**
- **Encrypt**
- **Initial Catalog**
- **Database**
- **Integrated Security**
- **Trusted_Connection**
- **Packet Size**
- **Persist Security Info**
- **Password**
- **User ID**
- **Pooling**

```
SqlConnectionStringBuilder connectionStringBuilder = new  
SqlConnectionStringBuilder(); // создание конструктора строк подключения  
  
connectionStringBuilder.DataSource = @"(LocalDB)\MSSQLLocalDB";  
connectionStringBuilder.InitialCatalog = "user";  
connectionStringBuilder.UserID = "admin";  
connectionStringBuilder.Password = "admin";  
using (SqlConnection connection = new  
SqlConnection(connectionStringBuilder.ConnectionString))  
{  
    try  
    {  
        connection.Open();  
  
    }  
    catch (Exception exception)  
    {  
  
    }  
}
```

Строки подключения в App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<connectionStrings>
<add name= "connect"
      connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\User\Documents\computerdb.mdf;Integrated
Security=True;Connect Timeout=30"
            providerName="System.Data.SqlClient" />
<add name= "WpfDb.Properties.Settings.userConnectionString" connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\user.mdf;Integrated Security=True;Connect
Timeout=30"
            providerName="System.Data.SqlClient" />
<add name="WpfDb.Properties.Settings.computerdbConnectionString"
      connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\computerdb.mdf;Integrated Security=True;Connect
Timeout=30"
            providerName="System.Data.SqlClient" />
<add name="WpfDb.Properties.Settings.NORTHWNDConnectionString"
      connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\NORTHWND.MDF;Integrated Security=True;Connect
Timeout=30"
            providerName="System.Data.SqlClient" />
</connectionStrings>
</configuration>
```

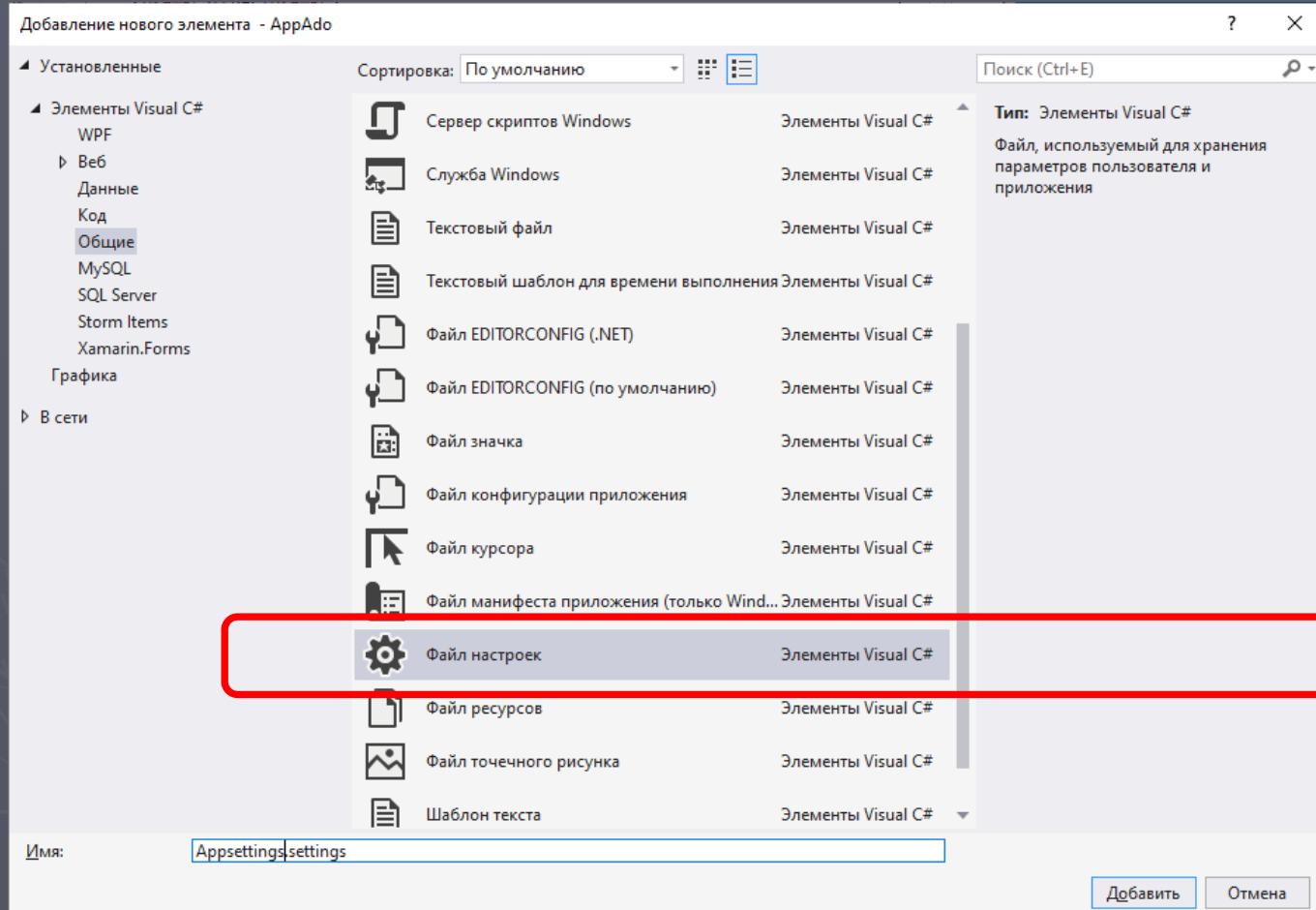
System.Configuration.ConfigurationManager Включить предварительные версии



System.Configuration.ConfigurationManager автор: Microsoft

Provides types that support using configuration files.

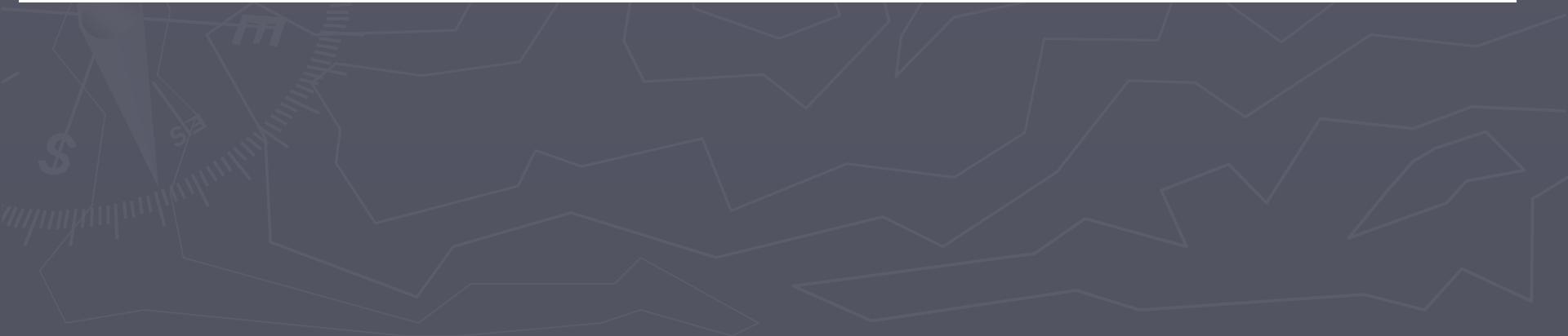
► 1) settings



```
try
{
    // определяем строку подключения
    SqlConnection thisConnection =
        new SqlConnection(
            ConfigurationManager.
                ConnectionStrings["connect"].
               ConnectionString);

    //открыть соединение
    thisConnection.Open();
    //thisConnection.OpenAsync();
}
```

```
private static async Task ConnectDB()
{
    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.
.ConnectionStrings["connect"].ConnectionString))
    {
        await connection.OpenAsync();
    }
}
```



2. Чтение данных посредством DataReader

```
//Создать команду для соединения
```

```
SqlCommand getCommand =  
thisConnection.CreateCommand();
```

```
//Специфицировать запрос SQL для команды
```

```
getCommand.CommandText =  
    "SELECT company, processor from Compy";
```

```
//Выполнить команду и получить данные
```

```
SqlDataReader thisReader =  
getCommand.ExecuteReader();
```

```
// пока есть строки для чтения
while (thisReader.Read())
{
    result.Items.Insert(0,
        (string)thisReader.GetValue(0) +
        (string)thisReader.GetValue(1));
}
//закрыть читатель
thisReader.Close();
//закрыть соединение
thisConnection.Close();
}
```

MainWindow

- □ ×

Sony intel
HP ADM
Lenovo Athlon

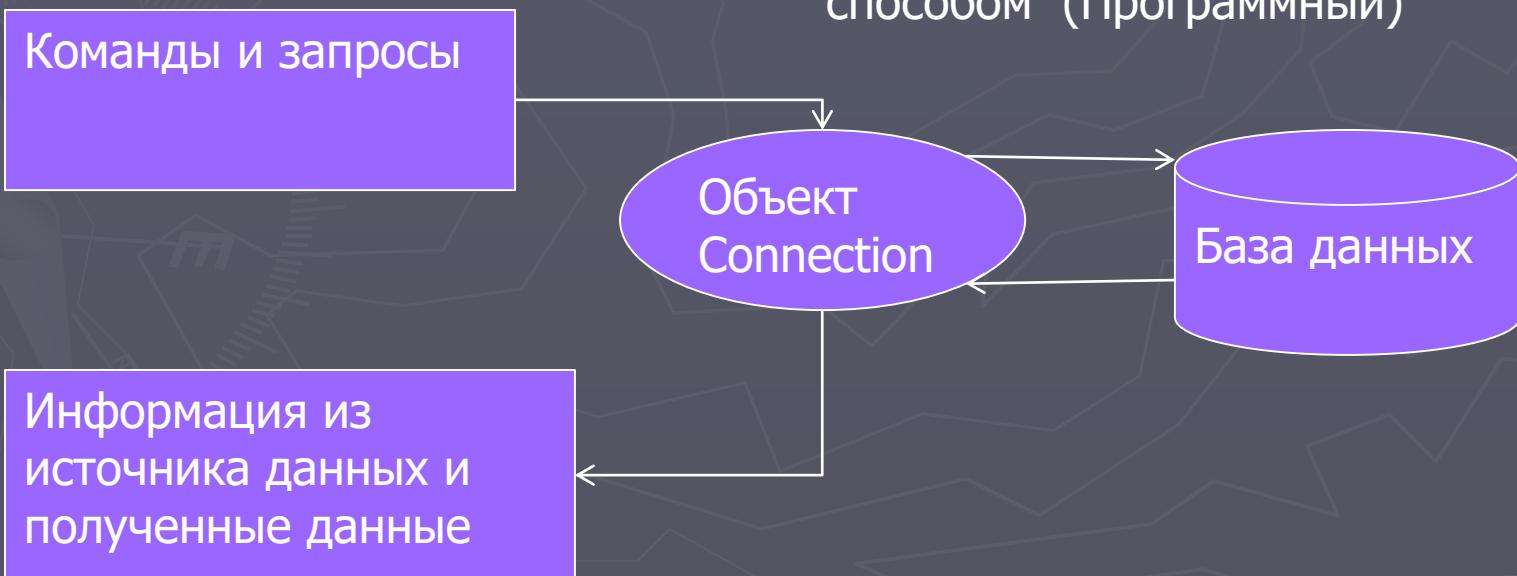
Connect

Постоянное подключение

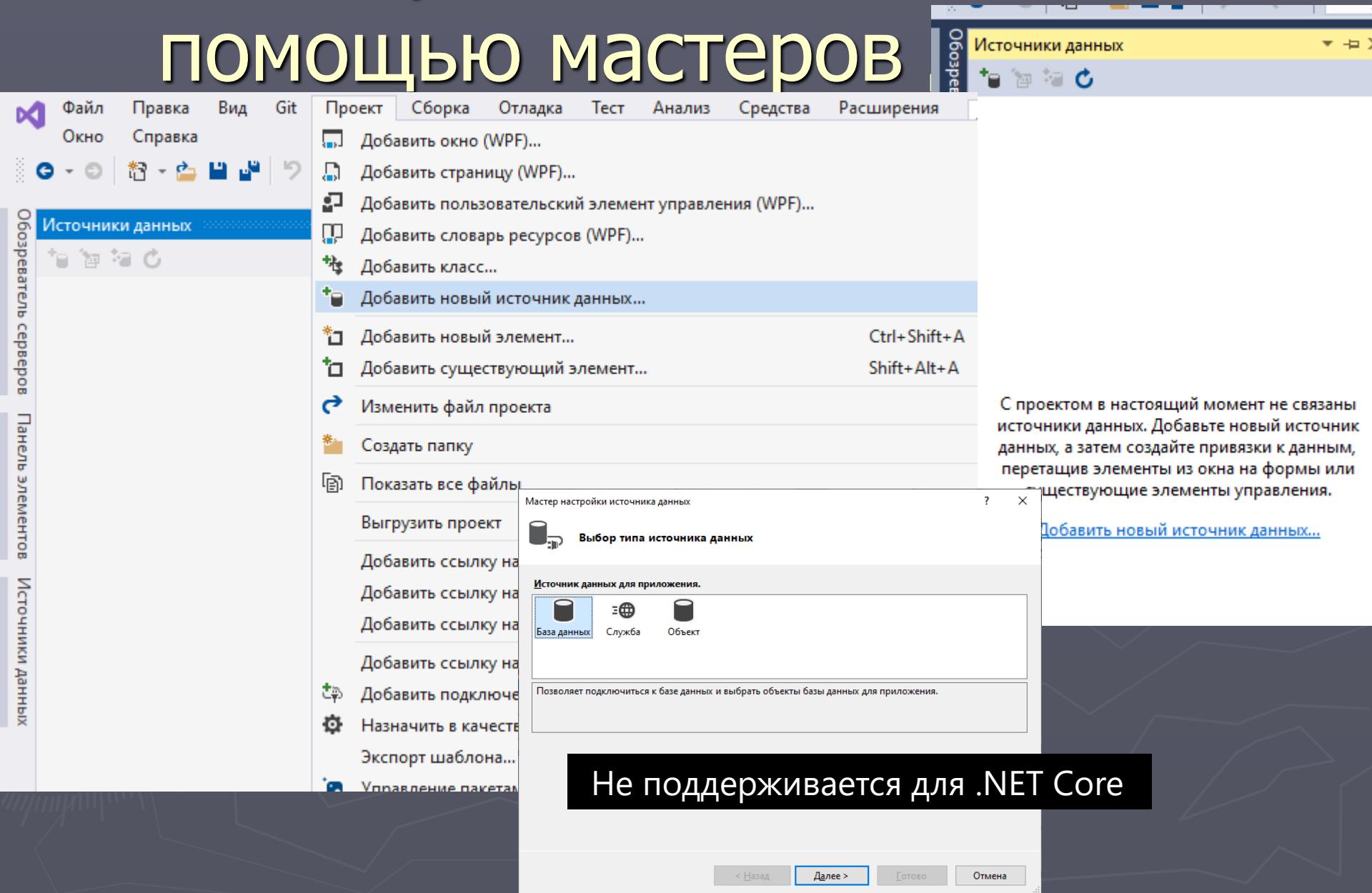
Настройка подключений

Connection – представление открытого подключения к источнику данных (канал)

1. Настройка подключений с помощью мастеров данных (Визуальный (Add connection))
2. Создание объектов программным способом (Программный)



1. Настройка подключений с помощью мастеров



Мастер настройки источника данных

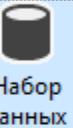
?

X



Выбор модели базы данных

Укажите тип модели базы данных, которую следует использовать.



Набор
данных

Выбранная модель базы данных определяет типы данных объектов, используемых кодом приложения. В проект будет добавлен файл набора данных.

< Назад

Далее >

Готово

Отмена

Мастер настройки источника данных



Выбор подключения к базе данных

Какое подключение ваше приложение должно использовать для работы с базой данных?

coputerdb.mdf

[Создать подключение...](#)

По-видимому, эта строка подключения содержит конфиденциальные данные (например, пароль), необходимые для создания подключения с базой данных. Хранение таких сведений в строке подключения представляет потенциальную угрозу безопасности. Добавить конфиденциальные данные в строку подключения?

- Нет, исключить конфиденциальные данные из строки подключения. Эти данные будут заданы в коде приложения.
- Да, включить конфиденциальные данные в строку подключения.

Показать строку подключения, которая будет сохранена в приложении

< Назад

Далее >

Готово

Отмена

Добавить подключение

?

×

Введите данные для подключения к выбранному источнику данных или нажмите кнопку "Изменить", чтобы выбрать другой источник данных и (или) поставщик.

Источник данных:

Файл базы данных Microsoft SQL Server (SqlClient)

[Изменить...](#)

Имя файла базы данных (новой или существующей):

C:\Users\User\Documents\comuper.mdf

[Обзор...](#)

Вход на сервер

Использовать проверку подлинности Windows

Использовать аутентификацию SQL Server

Имя пользователя:

Пароль:

Сохранить пароль

[Дополнительно...](#)

[Проверить подключение](#)

OK

Отмена



Выбор подключения к базе данных

Какое подключение ваше приложение должно использовать для работы с базой данных?

coputerdb.mdf

Создать подключение...

По-видимому, эта строка подключения содержит конфиденциальные данные (например, пароль), необходимые для создания подключения с базой данных. Хранение таких сведений в строке подключения представляет потенциальную угрозу безопасности. Добавить конфиденциальные данные в строку подключения?

- Нет, исключить конфиденциальные данные из строки подключения. Эти данные будут залены в коде приложения.
- Да, включить конфиденциальные данные

Показать строку подключения, которая будет с

```
Data Source=(LocalDB)\MSSQLLocalDB;AttachDbName=|Data File|;Integrated Security=True;Connect Timeout=30
```

Мастер настройки источника данных

?

X



Сохранение подключения в файле конфигурации приложения

Хранение строк с параметрами подключений в файле конфигурации приложения облегчает сопровождение и развертывание. Чтобы сохранить строку подключения в файле конфигурации приложения, введите имя в это поле, а затем нажмите кнопку "Далее".

Сохранить строку подключения в файле конфигурации приложения?

Да, сохранить подключение как:

< Назад

< Назад

Далее >

Готово

Отмена

Мастер настройки источника данных

Выбор объектов базы данных

Объекты базы данных для набора данных

- Таблицы
 - Compry
 - Study
- Представления
- Хранимые процедуры
- Функции

Имя набора данных (DataSet): coputerdbDataSet

< Назад Далее > Готово Отмена

Обозреватель решений

Обозреватель решений — поиск (Ctrl+;)

- Ссылки
- + App.config
- + App.xaml
- + App.xaml.cs
- coputerdb.mdf
- coputerdb_log.ldf
- + coputerdbDataSet.xsd
 - + coputerdbDataSet.Designer.cs
 - + coputerdbDataSet.xsc
 - + coputerdbDataSet.xss
- + MainWindow.xaml
- + MainWindow.xaml.cs

Источники данных

l.cs* App.config App.xaml.cs App.xaml

coputerdbDataSet

- Copy
- Idcomp
- company
- processor
- Study

Study

l.cs*

App.config

App.xaml.cs

App.xaml

М

89

69

Connect

DataGrid

```
rect Click="DoConnection" HorizontalAlignment="Left" Margin="10,10,10,10" HorizontalAlignment="Left" Height="197" Margin="96,89,0,0" VerticalAlignment="Top" VerticalAlignment="Left" Height="100" Margin="313,89,0,0" VerticalAlignment="Top"/>
```

Источники данных

+	coputerdbDataSet
-	Compy
-	Company
-	Idcomp
-	company
-	processor
-	Study
-	Study
-	Idstud
-	Name
-	compld

I.cs

App.config

App.xaml.cs

App.xaml

Idcomp	company	processor	FKStudy_ToCompy

Idstud:

Name:

comp Id:

Idcomp	company	proces
c1e54ace-360d-4dfa-bb6b-0039fa8075e7	Lenovo	Athlon
7f3a18bb-4573-42d9-9d57-0a61b103ff04	HP	ADM
d9c9e7cb-c5da-4ae9-85af-b839d454dba6	Sony	intel

Idstud: a841905d-35e8-4ab8-

Name: Dima

comp Id: 7f3a18bb-4573-42d9-

Connect

2. Создание объектов connection

программным способом

► Свойства, методы и события

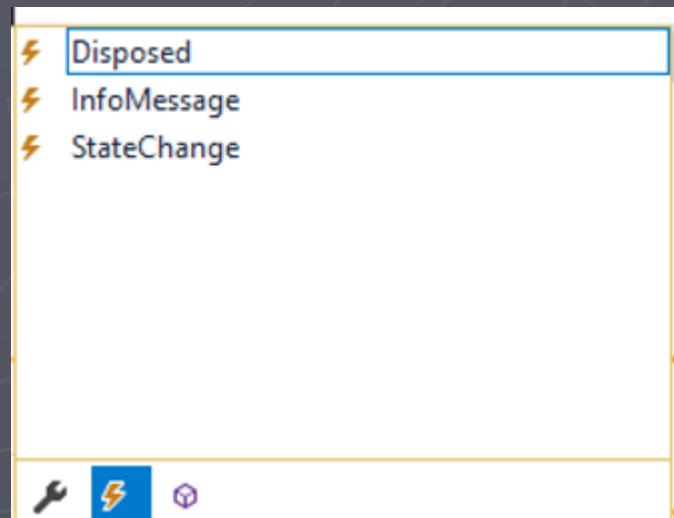
connection

The screenshot shows the Microsoft .NET API Explorer interface with three main sections:

- SqlConnection**: Shows properties like AccessToken, ClientConnectionId, ConnectionString, ConnectionTimeout, Container, Credential, Database, DataSource, and FireInfoMessageEventOnUserErrors.
- OleDbConnection**: Shows methods like BeginTransaction, ChangeDatabase, Close, CreateCommand, CreateObjRef, Dispose, EnlistDistributedTransaction, EnlistTransaction, Equals, GetLifetimeService, GetSchema, GetType, InitializeLifetimeService, Open, OpenAsync, ResetStatistics, RetrieveStatistics, and ToString.
- OdbcConnection**: Shows events like Disposed, InfoMessage, and StateChange.

Each section includes icons for edit, delete, and copy operations at the bottom.

- ▶ StateChange - изменение состояния БД
open- > close
- ▶ InfoMessage - сервер возвращает предупреждение или сообщения



3. Работа с пулами подключений

- ▶ Повторное использование сущ. подключений (позволяет использовать ранее созданные подключения) В пул помещаются подключения только с одинаковой конфигурацией
- ▶ Высокая производительность
- ▶ Разделены процессами, доменами приложений, строками подключений
- ▶ Активируется по умолчанию при создании connection

```
var setting = new ConnectionStringSettings
{
    Name = "userConnectionString",
        //имя строки подключения в конфигурационном файле
   ConnectionString = @"Data Source=\MSSQLLocalDB;
                        Initial Catalog=user;
                        Integrated Security=True;"
};

Configuration config;
    // Объект Config представляет конфигурационный файл
config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
    // Объект ConfigurationManager предоставляет доступ к файлам конфигурации
config.ConnectionStrings.ConnectionStrings.Add(setting);
config.Save();

    // Получение строки подключения.
string thisConnString=
ConfigurationManager.ConnectionStrings["userConnectionString"].ConnectionString;
```

4. Обработка ошибок подключения

SQL Server → ошибку или исключение - >
создается SQLException →
SQLException.Errors - коллекция ошибок
- > объекты SQLError

```
private void ConnectToDatabase(string connectionString)
{
    SqlConnection connection = new SqlConnection(connectionString);

    try
    {
        connection.Open();
    }
    catch (SqlException ex)
    {
        string errorMessage = "";
        foreach (SqlError ConnectionError in ex.Errors)
        {
            errorMessage += ConnectionError.Message + " (error: " + ConnectionError.Number.ToString() + ")" + Environment.NewLine;
            if (ConnectionError.Number == 18452)
            {
                MessageBox.Show("Invalid Login Detected, please provide valid credentials!");
            }
        }
        MessageBox.Show(errorMessage);
    }
    finally
    {
        connection.Close();
    }
}
```

5. Поиск доступных SQL Server

```
System.Data.SqlClient.SqlDataSourceEnumerator instance =  
    System.Data.SqlClient.SqlDataSourceEnumerator.Instance;  
VisibleSqlServers.DataSource = instance.GetDataSources();
```

Работа с данными

Command содержит информацию для:

- ▶ Выполнения выражений SQL
- ▶ Хранимых процедур и функций
- ▶ Возвращение данных в приложение
- ▶ Создание, удаление и изменение каталогов Бд.

SqlCommand

OleDbCommand

OdbcCommand

OracleCommand

► Основные свойства, события и методы command

d.

aR

ad

sa

il

SQL или
ХП

Properties:

- ColumnEncryptionSetting
- CommandText
- CommandTimeout
- CommandType
- Connection
- Container
- Notification
- NotificationAutoEnlist
- Parameters

Events:

- Disposed
- StatementCompleted

Text

StoredProcedure

Methods:

- BeginExecuteNonQuery
- BeginExecuteReader
- BeginExecuteXmlReader
- Cancel
- Clone
- CreateObjRef
- CreateParameter
- Dispose
- EndExecuteNonQuery

Equals

- ExecuteNonQuery
- ExecuteNonQueryAsync
- ExecuteReader
- ExecuteReaderAsync
- ExecuteScalar
- ExecuteScalarAsync
- ExecuteXmlReader
- ExecuteXmlReaderAsync

Methods:

- ExecuteXmlReader
- ExecuteXmlReaderAsync
- GetHashCode
- GetLifetimeService
- GetType
- InitializeLifetimeService
- Prepare
- ResetCommandTimeout
- ToString

1. Создание и настройка Command

► 1.1. Выполнение SQL

```
using (SqlConnection connection = new SqlConnection(connectionString))
{
    SqlCommand command = new SqlCommand();
    command.Connection = connection;
    command.CommandType = CommandType.Text;
    command.CommandText = "SELECT * From Compy";
    command.Connection.Open();

    SqlDataReader reader = command.ExecuteReader();
```

1.2 Выполнение хранимой процедуры

```
CREATE PROCEDURE [dbo].[selectCompyName]
AS
SELECT Study.Name, Compy.company
FROM Study, Compy
WHERE Study.compId = Compy.Idcomp;
RETURN 0
```

```
using (SqlConnection connection = new SqlConnection(connect)
{
    connection.Open();
    SqlCommand procCommand = new SqlCommand();
    procCommand.Connection = connection;
    procCommand.CommandType = CommandType.StoredProcedure;
    procCommand.CommandText = "selectCompyName";
```



```
SqlDataReader reader = procCommand.ExecuteReader();
while (reader.Read())
{
```

Dima	HP
Nikita	Sony
Olga	Sony
Pavel	Lenovo
Olga	Sony

► 1.3. Операции с каталогами (не возвр знач Insert delete update)

```
SqlCommand CreateTableCommand = new SqlCommand();
```

```
CreateTableCommand.Connection = NorthWindConnection;  
CreateTableCommand.CommandType = CommandType.Text;  
CreateTableCommand.CommandText = "CREATE TABLE SalesPersons (" +  
    "[SalesPersonID] [int] IDENTITY(1,1) NOT NULL, " +  
    "[FirstName] [nvarchar](50) NULL, " +  
    "[LastName] [nvarchar](50) NULL");
```

```
CreateTableCommand.Connection.Open();  
CreateTableCommand.ExecuteNonQuery();  
CreateTableCommand.Connection.Close();
```

► 1.4. Возврат одного значения

```
using (SqlConnection connection = new  
SqlConnection(connectionString))  
{  
  
    SqlCommand command = new SqlCommand();  
    command.Connection = connection;  
    command.CommandType = CommandType.Text;  
    command.CommandText =  
        "SELECT Count(*) From Compy";  
    command.Connection.Open();  
    int CompyCount = (int)command.ExecuteScalar();  
    MessageBox.Show(CompyCount.ToString());  
  
    SqlDataReader reader = command.ExecuteReader();  
  
}
```

ExecuteScalarAsync()

2. Асинхронное выполнение команд

Процесс выполнения команды в потоке ,
отличном от потока приложения

- ▶ Позволяют:

Работать над другими заданиями
приложения

```
private static async Task ReadFromStudyAsync()
{
    string connectionString =
ConfigurationManager.ConnectionStrings["connect"].ConnectionString;

    string sqlExpression = "WAITFOR DELAY '00:00:05';"+
                            "SELECT * FROM Study";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        await connection.OpenAsync();

        SqlCommand command = new SqlCommand(sqlExpression, connection);

        SqlDataReader reader = await command.ExecuteReaderAsync();
    }
}
```

```
if (reader.HasRows)
{
    MessageBox.Show((string)reader.GetName(0)+" "+(string)reader.GetName(1)+" "+(string) reader.GetName(2));

    while (await reader.ReadAsync())
    {
        object id = reader.GetValue(0);
        object name = reader.GetValue(1);
        object compyid = reader.GetValue(2);
        MessageBox.Show((string)id + " "+(string)name+ " "+(string)compyid);
    }
    reader.Close();
}
}
```

ReadFromStudyAsync().GetAwaiter();

Получение типизированных данных

sql	.net	метод
bigint	Int64	GetInt64
binary	Byte[]	GetBytes
bit	Boolean	GetBoolean
char	String и Char[]	GetString и GetChars
datetime	DateTime	GetDateTime
decimal	Decimal	GetDecimal
float	Double	GetDouble
image и long varbinary	Byte[]	GetBytes и GetStream
int	Int32	GetInt32
money	Decimal	GetDecimal
nchar	String и Char[]	GetString и GetChars
ntext	String и Char[]	GetString и GetChars
numeric	Decimal	GetDecimal
nvarchar	String и Char[]	GetString и GetChars
real	Single (float)	GetFloat
smalldatetime	DateTime	GetDateTime
smallint	Int16	GetInt16
smallmoney	Decimal	GetDecimal
sql variant	Object	GetValue
long varchar	String и Char[]	GetString и GetChars
timestamp	Byte[]	GetBytes
tinyint	Byte	GetByte
uniqueidentifier	Guid	GetGuid
varbinary	Byte[]	GetBytes

```
class A
```

```
{
```

```
    static async Task<int> Method(SqlConnection conn, SqlCommand cmd)
```

```
{
```

```
        await conn.OpenAsync();
```

```
        await cmd.ExecuteNonQueryAsync();
```

```
        return 1;
```

```
}
```

```
////////////////////////////////////////////////////////////////
```

```
{
```

```
    using (SqlConnection conn = new SqlConnection(connection))
```

```
{
```

```
        SqlCommand command = new SqlCommand("select top 2 * from C
```

```
        int result = A.Method(conn, command).Result;
```

```
        SqlDataReader reader = command.ExecuteReader();
```

```
        while (reader.Read())
```

```
            //...
```

```
}
```

```
}
```

3. Выполнение нескольких SQL выражений

```
ExecuteSqlCommand.CommandText = "SELECT CustomerID, CompanyName FROM " +  
    "Customers; SELECT ProductName, UnitsInStock FROM Products";  
SqlDataReader reader = ExecuteSqlCommand.ExecuteReader();  
bool MoreResults = false;  
do  
{  
    while (reader.Read())  
    {  
        for (int i = 0; i < reader.FieldCount; i++)  
        {  
        }  
    }  
    MoreResults = reader.NextResult();  
} while (MoreResults);
```

Итого

- ▶ Для выполнения SQL и ХП - Command
- ▶ Существуют для каждого провайдера
- ▶ Используются для выполнения операций с каталогами БД
- ▶ Могут выполняться асинхронно
- ▶ Могут возвращаться в виде таблицы
DataReader

4. Работа с параметрами в командах SQL

- ▶ Параметр – тип переменной, используемая для передачи значений между приложением и Бд.

WHERE

```
string insertcommand = String.Format("INSERT  
INTO Study (Idstud, Name, compId) " +  
"VALUES ('{0}', {1},{3})", a, b, c);
```

c = "10);INSERT INTO Users (Name, Age) VALUES('admin','admin")

4.1. Создание и выполнение параметризованного запроса

SqlParameter

CompareInfo	Size
DbType	SourceColumn
Direction	SourceColumnNullMapping
ForceColumnEncryption	SourceVersion
IsNullable	SqIDbType
LocaleId	SqlValue
Offset	TypeName
ParameterName	UdtTypeName
Precision	Value
 	 

```

string sqlExpression = "INSERT INTO Study (Idstud, Name, compId)
VALUES (@idS, @name, @idC)";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    SqlCommand command = new SqlCommand(sqlExpression, connection);

    // создаем параметр для id
    SqlParameter isSPParam = new SqlParameter("@idS", Guid.NewGuid().ToString());
    // добавляем параметр к команде
    command.Parameters.Add(isSPParam);

    // создаем параметр для имени
    SqlParameter nameParam = new SqlParameter("@name", "Nikita");
    // добавляем параметр к команде
    command.Parameters.Add(nameParam);
    // создаем параметр для idC
    SqlParameter idCParam = new SqlParameter("@idC", "d9c9e7cb-c5da-4ae9-85af-
b839d454dba6");
    // добавляем параметр к команде
    command.Parameters.Add(idCParam);
    int number = command.ExecuteNonQuery();
    rezult.Items.Add($"Добавлено объектов: {number}" ); // 1
}

```

	Idstud	Name	compld
	a841905d-35e8-4ab8-b93c-067...	Dima	7f3a18bb-4573-42d9-9d57-0a61b103ff04
	40bb8e45-31a1-4667-86f3-1e0...	Nikita	d9c9e7cb-c5da-4ae9-85af-b839d454dba6
▶	04d1fe6f-2a2d-4e7a-a9b5-f07e...	Pavel	c1e54ace-360d-4dfa-bb6b-0039fa8075e7
*	NULL	NULL	NULL

Добавлено объектов: 1

Direction - > из перечисления ParameterDirection

- ▶ Input Для передачи данных в БД - по умолчанию
- ▶ Output Для получения данных из БД
- ▶ InputOutput Для передачи и получения
- ▶ ReturnValue

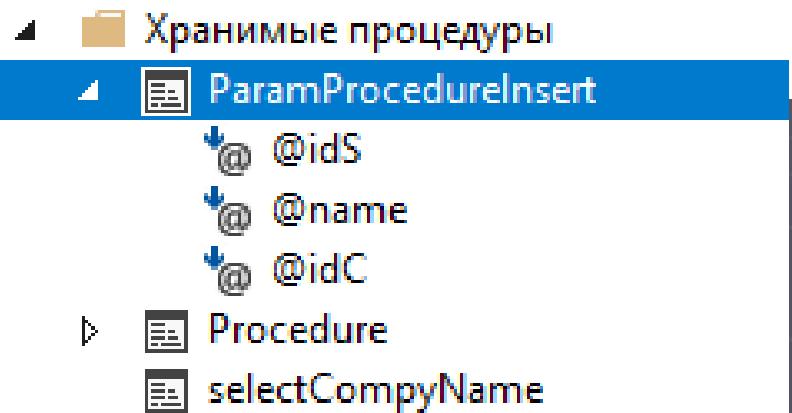
: 1, Id студента 306091ff-020e-4711-8319-f5347146fc2e

```
//...
// создаем параметр для id
SqlParameter isSSParam = new SqlParameter("@idS",
                                         Guid.NewGuid().ToString());
isSSParam.Direction = ParameterDirection.InputOutput;
// добавляем параметр к команде
command.Parameters.Add(isSSParam);
// создаем параметр для имени
SqlParameter nameParam = new SqlParameter("@name", "Olga");
//...
int number = command.ExecuteNonQuery();

result.Items.Add($"Добавлено объектов: {number},
Id студента {isSSParam.Value}" ); // 1
```

4.2. Параметризованные ХП

```
CREATE PROCEDURE
[dbo].[ParamProcedureInsert]
@idS uniqueidentifier,
@name nvarchar(10),
@idC uniqueidentifier
AS
INSERT INTO Study (Idstud, Name, compId)
VALUES (@idS, @name, @idC)
GO
```



```
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();

    SqlCommand procParamCommand = new SqlCommand();
    procParamCommand.Connection = connection;
    procParamCommand.CommandType = CommandType.StoredProcedure;
    procParamCommand.CommandText = "ParamProcedureInsert";

    //создаем параметры
    SqlParameter IdParameter = new SqlParameter();
    IdParameter.ParameterName = "@idS";
    IdParameter.SqlDbType = SqlDbType.UniqueIdentifier;
    IdParameter.Value = Guid.NewGuid().ToString();

    SqlParameter NameParameter = new
SqlParameter("@name", (string)"Azamat");
    SqlParameter IdCompParameter = new SqlParameter("@idC",
SqlDbType.UniqueIdentifier);
    IdCompParameter.Value = "d9c9e7cb - c5da - 4ae9 - 85af -
b839d454dba6";

    procParamCommand.Parameters.Add(IdParameter);
    procParamCommand.Parameters.Add(NameParameter);
    procParamCommand.Parameters.Add(IdCompParameter);
    int number = procParamCommand.ExecuteNonQuery();
```

5. Выполнение транзакций

Команды, которые выполняются как группа.

Класс Transaction

сущ. для каждого провайдера

Используют для сохранения целостности данных.

Бывают - локальные и распределенные

```
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    SqlTransaction transact = connection.BeginTransaction();
    SqlCommand transCommand = connection.CreateCommand();
    transCommand.Transaction = transact;
    try
    { // выполняем две отдельные команды
        string idC = Guid.NewGuid().ToString();
        transCommand.CommandText = "INSERT INTO
            Compy (Idcomp, company, processor)
            VALUES ('" + idC + "', 'Apple', 'NoName')";
        transCommand.ExecuteNonQuery();
        string idS = Guid.NewGuid().ToString();
        transCommand.CommandText = "INSERT INTO Study (Idstud, Name, compId)
            VALUES ('" + idS + "', 'Abdul', '" + idC + "')";
        // transCommand.CommandText = " UPDATE Compy
        // SET processor = 'Intel Core i7' WHERE company = 'Apple'";
        transCommand.ExecuteNonQuery();
        // подтверждаем транзакцию
        transact.Commit();
        result.Items.Add("Commit");
    }
    catch (Exception ex)
    {
        // откатить транзакцию
        transact.Rollback();
        result.Items.Add("Rollback");
    }
}
```

Commit

► 5.3. Пример

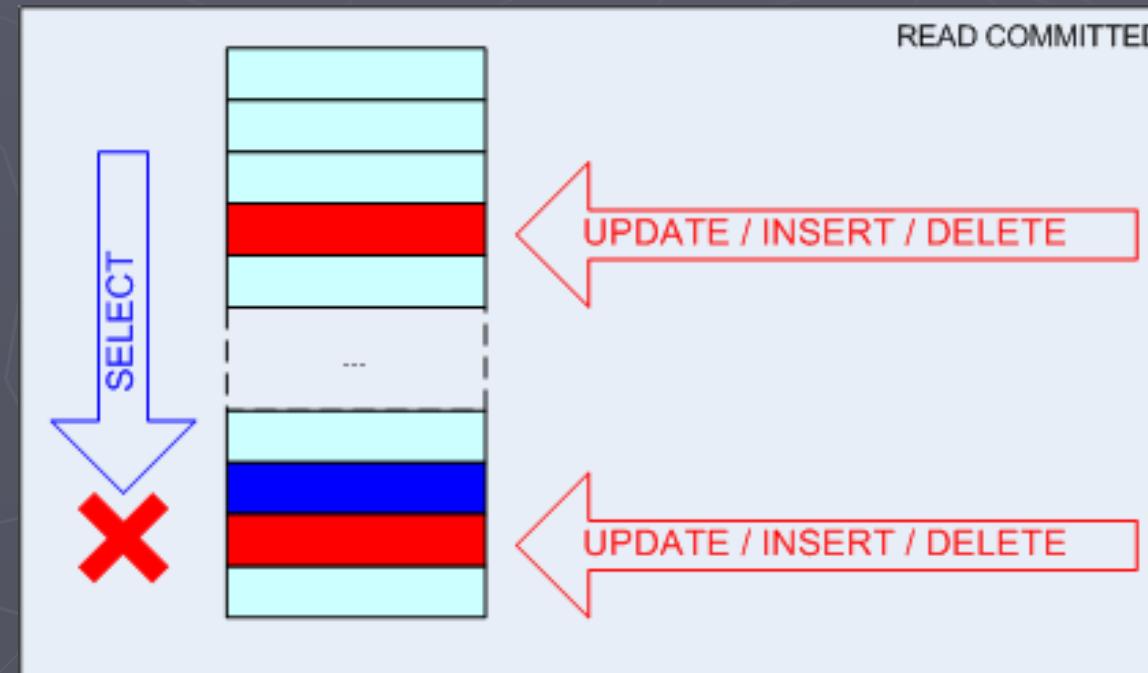
```
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    SqlTransaction transact = connection.BeginTransaction();
    // выполняем две отдельные команды
    string idC = Guid.NewGuid().ToString();
    SqlCommand transCommand1 = connection.CreateCommand();
    transCommand1.CommandText = "INSERT INTO Compy (Idcomp, company, processor)
        VALUES ('" + idC + "', 'Apple', 'NoName')";
    SqlCommand transCommand2 = connection.CreateCommand();
    string idS = Guid.NewGuid().ToString();
    transCommand2.CommandText = "INSERT INTO Study (Idstud, Name, compId)
        VALUES ('" + idS + "', 'Abdul', '" + idC + "')";
    transCommand1.Transaction = transact;
    transCommand2.Transaction = transact;
    transCommand1.ExecuteNonQuery();
    transCommand2.ExecuteNonQuery();

    //диалог
    MessageBoxResult response = MessageBox.Show("Команда выполнена" + Environment.NewLine +
        "отменить?", "продолжить", MessageBoxButtons.YesNo);
    switch (response)
    {
        case MessageBoxResult.Yes:
        // подтверждаем транзакцию
        transact.Commit();
        break;
        case MessageBoxResult.No:
        // откатить транзакцию
        transact.Rollback();
        break;
    }
}
```

► 5.4. Установка уровня изоляции

Управление возможностью других потоков получать доступ к данным, находящимся в обращении у процесса текущей транзакции

более высокий уровень изолированности соответствует лучшей согласованности данных, но его использование может снижать количество физически параллельно выполняемых транзакций. И наоборот, более низкий уровень изолированности позволяет выполнять больше параллельных транзакций, но снижает точность данных.



Проблемы при параллельном выполнении транз:

- потеряное обновление (*lost update*) — при одновременном изменении одного блока данных разными транзакциями одно из изменений теряется;
- «грязное» чтение (*dirty read*) — чтение данных, которое впоследствии не подтверждится (откатится);
- неповторяющееся чтение (*non-repeatable read*) — при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными;
- фантомное чтение (*phantom reads*) — одни и те же выборки дают разные множества строк.

```

connection = new SqlConnection(connectionString))

transaction = connection.BeginTransaction(IsolationLevel.);

```

Чтобы убедиться в том, что изолированная транзакция не влияет на другие транзакции, можно использовать метод IsolationLevel.ReadCommitted.

```

newGuidId().ToString();
command1 = connection.CreateCommand();
commandText = "INSERT INTO Compy (Idcomp, company,
command2 = connection.CreateCommand();
newGuidId().ToString();

```

- Chaos
- ReadCommitted
- ReadUncommitted
- RepeatableRead
- Serializable

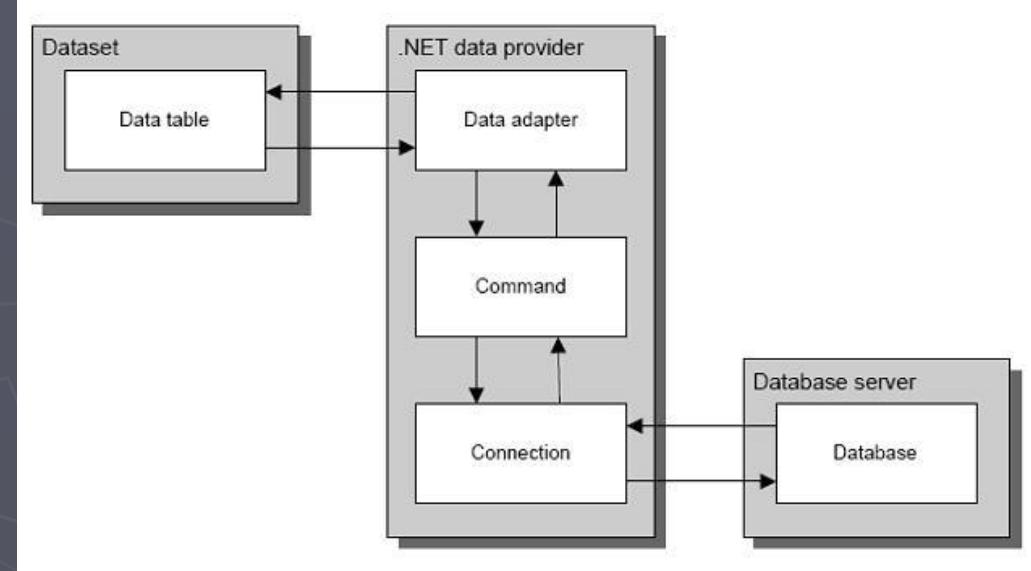
		Эффекты			
		Потерянное обновление	Грязное чтение	Неповторяющееся чтение	Фантомное чтение
Уровни изоляции	Read uncommitted	Нет /Есть (*)	Есть	Есть	Есть
	Read committed или Read committed Snapshot (**)	Нет /Есть (*)	Нет	Есть	Есть
	Repeatable read	Нет	Нет	Нет	Есть
	Serializable или Snapshot (**)	Нет	Нет	Нет	Нет

Работа с данными в отсоединенной среде

► *disconnected layer*

DataSet - хранилище данных

SqlDataAdapter - объект, который заполняет
DataSet данными из БД

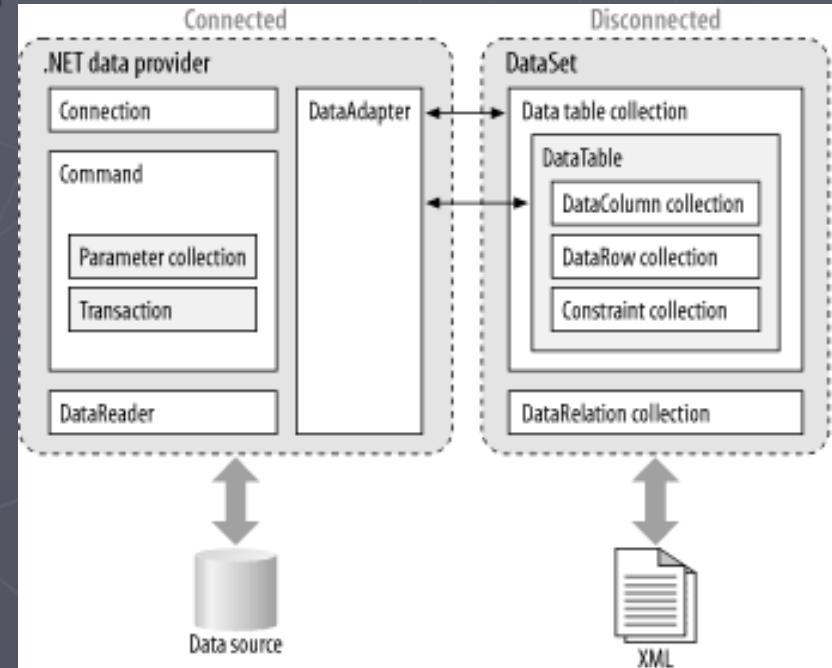


- .NET Framework (all versions)
- .NET Core and later
- .NET 5.0 and later

- ▶ Кэширование данных в приложение
- ▶ Отсоединение от БД
- ▶ Работа с данными
- ▶ Подключение к БД
- ▶ Сохранение в источнике данных

DataSet

DataTable



1. Создание объектов DataSet

- ▶ **System.Data**
- ▶ Типизированные – наследуют схему из *.xsd и содержат типизированные коллекции определенного объекта (CustomerTable)
- ▶ Нетипизированные – стандартные обобщенные экземпляры DataSet со схемой и объектами DataTable - > к коллекции Tables, DataColumn.
Обращение через индексы.

Способы создания DataSet

- ▶ 1. Программно в коде
- ▶ 2. инструменты времени разработки –
DataSet Designer , Data Source
Configuration Wizard
- ▶ 3. На основе DataAdapter

Создание DataSet программно

```
DataSet NorthwindDataSet = new DataSet("NorthwindDataSet");
```

```
DataTable CustomersTable = new DataTable();  
DataTable OrderDataTable = new DataTable();
```

```
NorthwindDataSet.Tables.Add(CustomersTable);  
NorthwindDataSet.Tables.Add(OrderDataTable);
```

Создать набор, таблицы, добавить их к коллекции набора

Создание и добавление связей между таблицами

```
DataRelation CustomerOrders = new DataRelation("CustomerOrders",
    CustomersTable.Columns["CustomerID"],
    OrderDataTable.Columns["CustomerID"]);
```

```
NorthwindDataSet.Relations.Add(CustomerOrders);
```

Перемещение между связанными таблицами

- ▶ Выбрать DataRow
- ▶ Вызвать GetParentRow (1) или GetChildRows (массив)

Объединение наборов

DataSet.Merge()

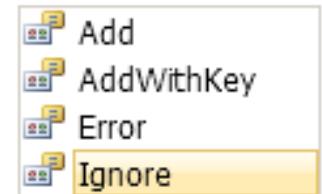
PreserveChanges

Merge(DataRow[] rows, bool preserveChanges, MissingSchemaAction missingSchemaAction)

Метод слияния массива объектов System.Data.DataRow и текущего объекта System.Data.DataSet, сохраняя или удаляя любые изменения схемы в соответствии с заданными аргументами.

maAction: Одно из значений типа System.Data.MissingSchemaAction.

Set2.Merge(NorthwindDataSet, true, MissingSchemaAction.



record (DataRow) to add to the table.

Режим объединения

```
DataSet NorthwindDataSet2 = new DataSet("NorthwindDataSet2");
```

```
NorthwindDataSet2.Merge(NorthwindDataSet, true, MissingSchemaAction.Ignore);
```

Действия с наборами

► 1) копирование

```
DataSet nDS = new DataSet();
nDS = nDS.Copy();
```

► 2) очистка

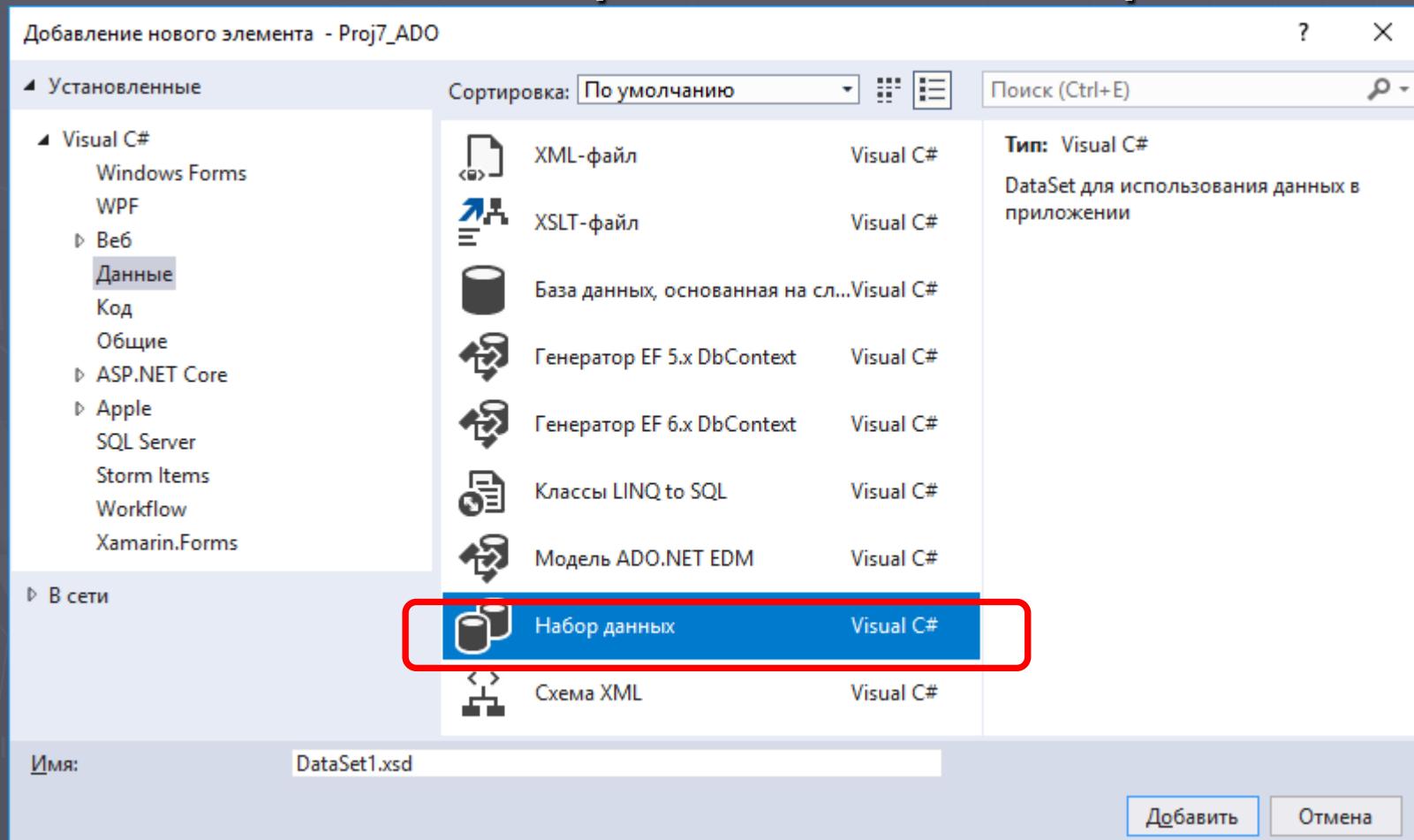
```
nDS = nDS.Clear();
```

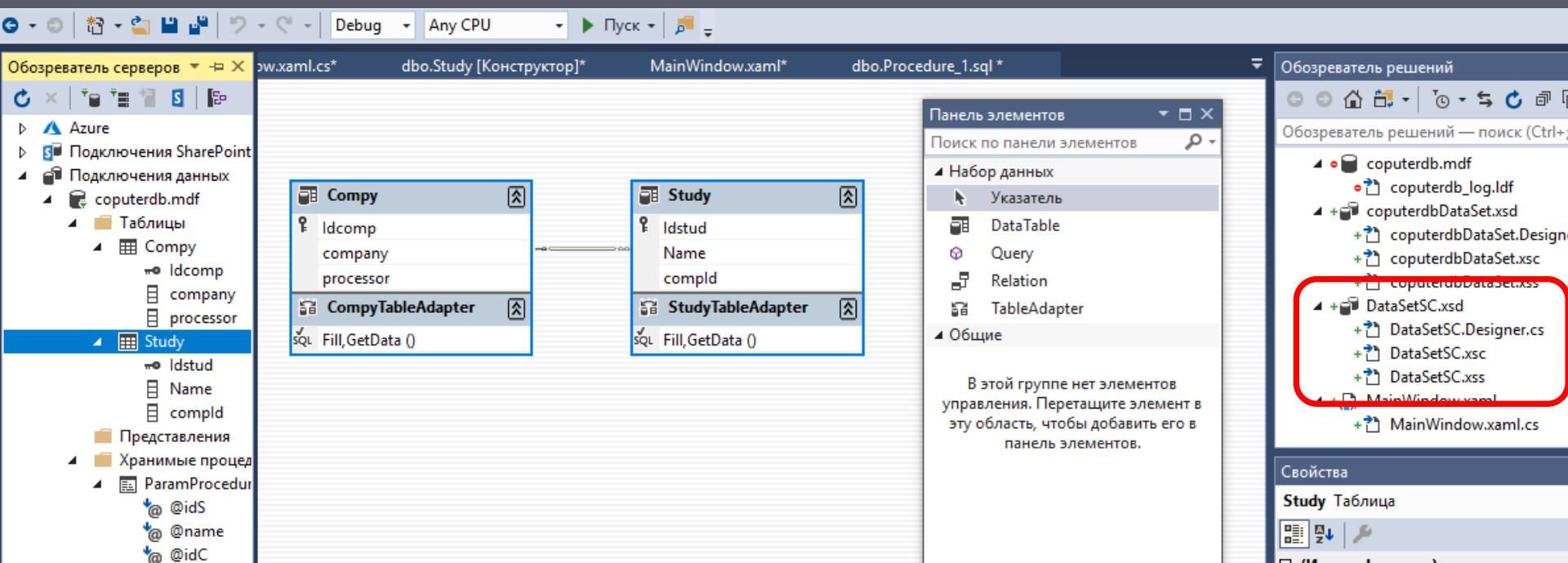
► 3) ReadXml() , WriteXml()

► 4) AcceptChanges() GetChanges() RejectChanges()

Создание набора в Dataset Designer

DataSet - типизированный набор

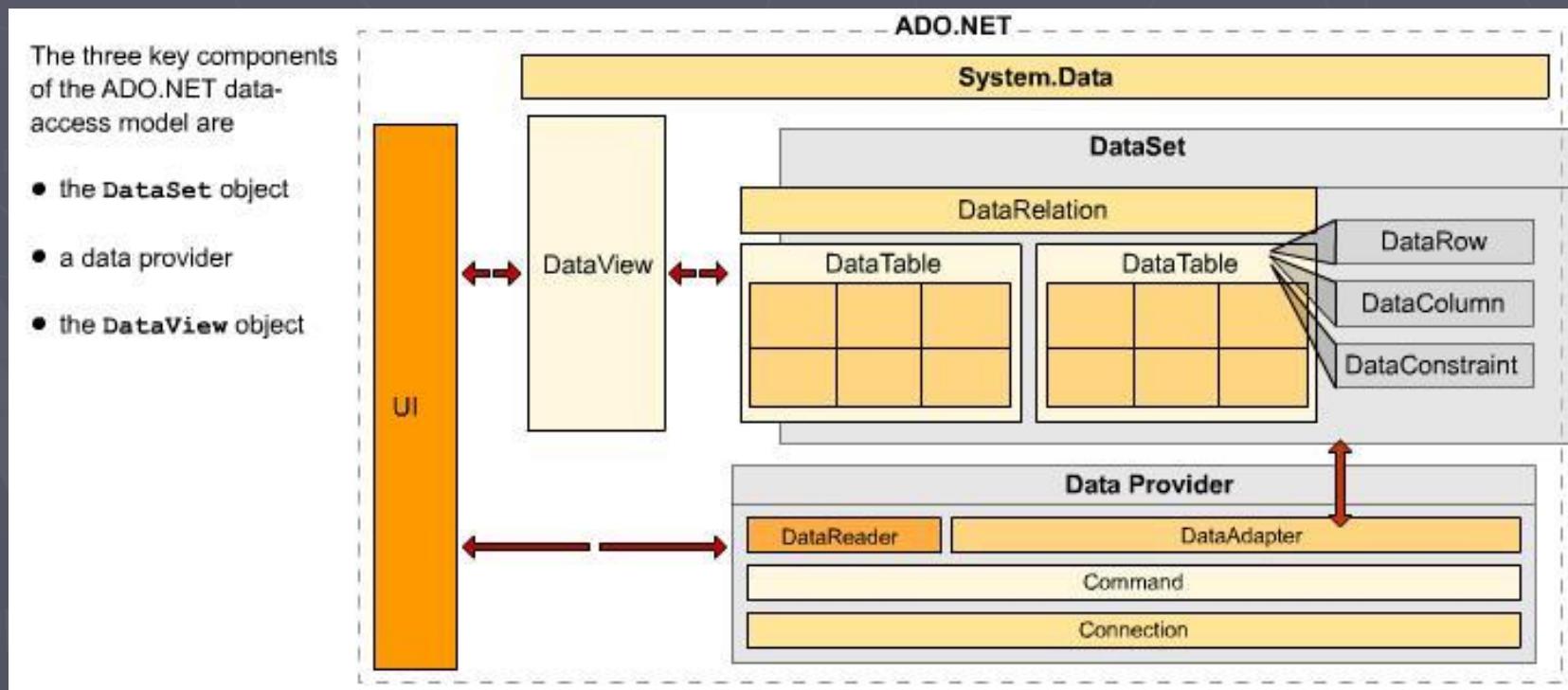




Перетаскиваем из обозревателя серверов на поверхность конструктора

2. Работа с объектами DataTable

- ▶ Могут добавляться к DataSet и связываться
- ▶ Могут быть использованы как автономные объекты без DataSet



► 1) создание

► 2) добавление таблиц к набору

► New DataSet New DataTable Add

Id	Name
1	Анна
2	Влад

```
DataSet studGroup = new DataSet("StudGroup");
    DataTable studTable = new DataTable("Student");
    // добавляем таблицу в dataset
    studGroup.Tables.Add(studTable);

    // создаем столбцы для таблицы
    DataColumn idColumn = new DataColumn("Id", Type.GetType("System.Int32"));
    idColumn.Unique = true; // столбец будет иметь уникальное значение
    idColumn.AllowDBNull = false; // не может принимать null
    idColumn.AutoIncrement = true; // будет автоинкрементироваться
    idColumn.AutoIncrementSeed = 1; // начальное значение
    idColumn.AutoIncrementStep = 1; // приращении при добавлении новой строки

    DataColumn nameColumn = new DataColumn("Name", Type.GetType("System.String"));

    studTable.Columns.Add(idColumn);
    studTable.Columns.Add(nameColumn);

    // определяем первичный ключ таблицы books
    studTable.PrimaryKey = new DataColumn[] { studTable.Columns["Id"] };

    DataRow row = studTable.NewRow();
    row.ItemArray = new object[] { null, "Анна" };
    studTable.Rows.Add(row); // добавляем первую строку
    studTable.Rows.Add(new object[] { null, "Влад" }); // добавляем вторую строку

    dataGridCompy.ItemsSource = studGroup.Tables[0].DefaultView;
```

2.1. Создание вычисляемых столбцов

```
DataTable TotalPrice = new DataColumn("TotalPrice", System.Type.GetType("System.Double"));

TotalPrice.Expression = ("UnitPrice * Quantity");
NorthwindDataSet.OrderDataTable.Add( TotalPrice);
```

► 2.3. Создание отношений

```
dataset1.Relations.Add(...)
```

2.4. Добавление ограничений в таблице

Ограничение внешнего ключа

```
ForeignKeyConstraint Key = new ForeignKeyConstraint (|
```

▲ 1 из 6 ▼ ForeignKeyConstraint.ForeignKeyConstraint(DataColumn parentColumn, DataColumn childColumn)

Инициализирует новый экземпляр класса System.Data.ForeignKeyConstraint с указанным parentColumn: Родительский объект System.Data.DataColumn в ограничении.

```
ForeignKeyConstraint foreignKey = new  
ForeignKeyConstraint(studTable.Columns["Id"],  
compTable.Columns["CompyId"])
```

```
{
```

```
    ConstraintName = "StudentComnyForeignKey",
```

```
    DeleteRule = Rule.SetNull,
```

```
    UpdateRule = Rule.Cascade
```

```
};
```

```
// добавляем внешний ключ в dataset
```

```
studGroup.Tables["Compy"].Constraints.Add(foreignKey);
```

```
// применяем внешний ключ
```

```
studGroup.EnforceConstraints = true;
```

главный столбец и
зависимый столбец

название внешнего ключа.
Свойства DeleteRule И UpdateRule опре-
деляют поведение при удалении или
обновлении данных

3. Создание DataAdapter

Специфичны для провайдера

- ▶ Содержат сведения

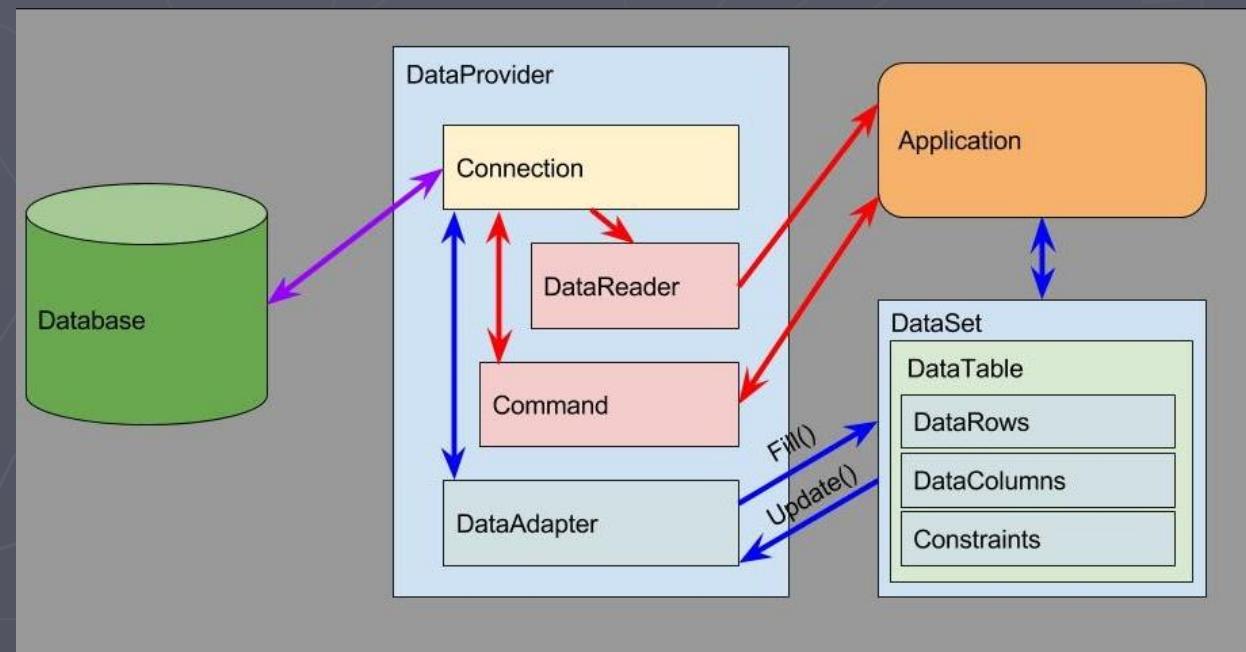
О подключениях

Команды SELECT, INSERT, DELETE, UPDATE

Для заполнения DataSet И DataTable

- ▶ Создаются

Программно
Визуально



3.1. Создать объекта и команды

```
SqlDataAdapter dataGridView1 = new SqlDataAdapter ("SELECT * FROM Customers", );
```

▲ 3 из 4 ▼ **SqlDataAdapter.SqlDataAdapter(string selectCommandText, SqlConnection selectConnection)**

Инициализирует новый экземпляр класса System.Data.SqlClient.SqlDataAdapter с помощью System.Data.SqlClient.SqlConnection.

selectConnection: System.Data.SqlClient.SqlConnection, предоставляющее подключение.

Insert Update Delete

```
string sqlCommand = "SELECT * FROM Compy";
    using (SqlConnection connection =
        new SqlConnection(connectionString))
{
    connection.Open();
    // Создаем DataAdapter
SqlDataAdapter adapter = new SqlDataAdapter(sqlCommand, connection)
    // Создаем Dataset
    DataSet dataSt = new DataSet();
    // Заполняем Dataset
    adapter.Fill(dataSt);
    // Отображаем данные
    dataGridViewCompy.ItemsSource = dataSt.Tables[0].DefaultView
}
```

Idcomp	company	processor	
c1e54ace-360d-4dfa-bb6b-0039fa8075e7	Lenovo	Athlon	
7f3a18bb-4573-42d9-9d57-0a61b103ff04	HP	ADM	
195dae17-fde5-44e4-b5c7-a4fd664e6155	Apple	NoName	
9ac5aeab-3170-4d30-ba09-b3bdfc8e5372	Apple	NoName	

DataAdapter

3.2 Генерирование команд на основе CommandBuilder

- ▶ Для модификации данных в БД в соответствии с изменениями в DataSet
SqlDataAdapter использует команды
InsertCommand, UpdateCommand и DeleteCommand.
- ▶ По Select генерирует INSERT UPDATE
DELETE
- ▶ SELECT должно возвращать ≥ 1 столбец с первич. ключом

SqlCommandBuilder позволяет автоматически сгенерировать выражения

```
string sqlCommand = "SELECT * FROM Compy";
    using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    /// Создаем DataAdapter
    SqlDataAdapter adapter = new SqlDataAdapter(sqlCommand, connection);
    // Создаем Dataset
    DataSet dataSt = new DataSet();
    // Заполняем Dataset
    adapter.Fill(dataSt);
    // Отображаем данные
    dataGridViewCompy.ItemsSource= dataSt.Tables[0].DefaultView;

    DataTable dt = dataSt.Tables[0];
    DataRow someRow = dt.Rows[2];
    someRow["processor"] = "intel Core7";
    // создаем объект SqlCommandBuilder
    SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);
    adapter.Update(dataSt);
    // заново получаем данные из бд
    // очищаем полностью DataSet
    dataSt.Clear();
    // перезагружаем данные
    adapter.Fill(dataSt);
```

3.3. Обновление базы на основе DataSet программно

```
string sqlCommand = "SELECT * FROM Study";
    using (SqlConnection connection = new
SqlConnection(connectionString))
{
    connection.Open();
    /// Создаем DataAdapter
    SqlDataAdapter adapter =
        new SqlDataAdapter(sqlCommand, connection);
```

```
//adapter.DeleteCommand  
//adapter.UpdateCommand  
    // устанавливаем команду на вставку  
adapter.InsertCommand = new SqlCommand("ParamProcedureInsert", connection);  
    // это будет хранимая процедура  
    adapter.InsertCommand.CommandType = CommandType.StoredProcedure;  
  
    adapter.InsertCommand.Parameters.Add(  
        new SqlParameter("@idS", SqlDbType.UniqueIdentifier, 38, "IdStud"))  
    adapter.InsertCommand.Parameters.Add(  
        new SqlParameter("@name", SqlDbType.NVarChar, 10, "Name"))  
    adapter.InsertCommand.Parameters.Add(  
        new SqlParameter("@idC", SqlDbType.UniqueIdentifier, 38, "compId"))  
  
    DataSet ds = new DataSet();  
    adapter.Fill(ds);  
    DataTable dt = ds.Tables[0];  
    DataRow nRow = dt.NewRow();  
    nRow["IdStud"] = Guid.NewGuid().ToString();  
    nRow["Name"] = "Anna";  
    nRow["compId"] = "d9c9e7cb-c5da-4ae9-85af-b839d454dba6";  
  
    adapter.Update(ds);  
    ds.AcceptChanges();
```

3.3.Разрешение конфликтов

- ▶ При не соответствии DataAdapter и DataSet
- ▶ MissingMappingAction (нет табл, столб) → Error, Ignore, Passthrough
- ▶ MissingSchemaAction (нет схемы) → Add, error, Ignore, AddWithKey

3.4 Выполнение пакетной обработки

- ▶ Для сокращения количества перемещений

```
// отправить пакеты 5-и предложений  
SqlDataAdapter1.UpdateBatchSize = 5;
```

События - RowUpdating – стр
RowUpdated – пакет (1)

LINQ to DataSet

технология доступа и управления реляционными данными

Менеджер ссылок - Proj7_ADO

Сборки Целевая платформа: .NET Framework 4.6.2

Платформа Имя Версия

System.Configuration.Install	4.0.0.0
System.Core	4.0.0.0
System.Data	4.0.0.0
System.Data.DataSetExtensions	4.0.0.0
System.Data.Entity	4.0.0.0
System.Data.Entity.Design	4.0.0.0
System.Data.Linq	4.0.0.0
System.Data.OracleClient	4.0.0.0
System.Data.Services	4.0.0.0
System.Data.Services.Client	4.0.0.0
System.Data.Service	
System.Data.SqlXml	
System.Deployment	
System.Design	
System.Device	
System.DirectorySer	
System.DirectorySer	
System.DirectorySer	
System.Drawing	
System.Drawing.Des	
System.Dynamic	
System.EnterpriseSe	
System.IdentityMod	
System.IdentityMod	
System.IdentityMod	
System.IO.Compres	

Поиск (Ctrl+E) ? X

Имя: System.Data.Linq
Кем создано: Microsoft Corporation
Версия: 4.0.0.0
Версия файла: 4.6.1586.0

Платформа Расширения Последние

Проекты Общие проекты COM Обзор

NuGet: AppAdo AppAdo: Опубликовать MainWindow.xaml.cs* AppAdo.csproj*

Обзор Установлено Обновления

Поиск (Ctrl+L) Включить предварительные

Для Core

Mindbox.Data.Linq автор: Minc v3.0.5 A clone of Microsoft System.Data.Linq to allow multi-DL...

.NET System.Configuration.Config v5.0.0 Provides types that support using configuration files.

.NET System.Data.SqlClient автор: N v4.8.2 Provides the data provider for SQL Server. These classes provide access...

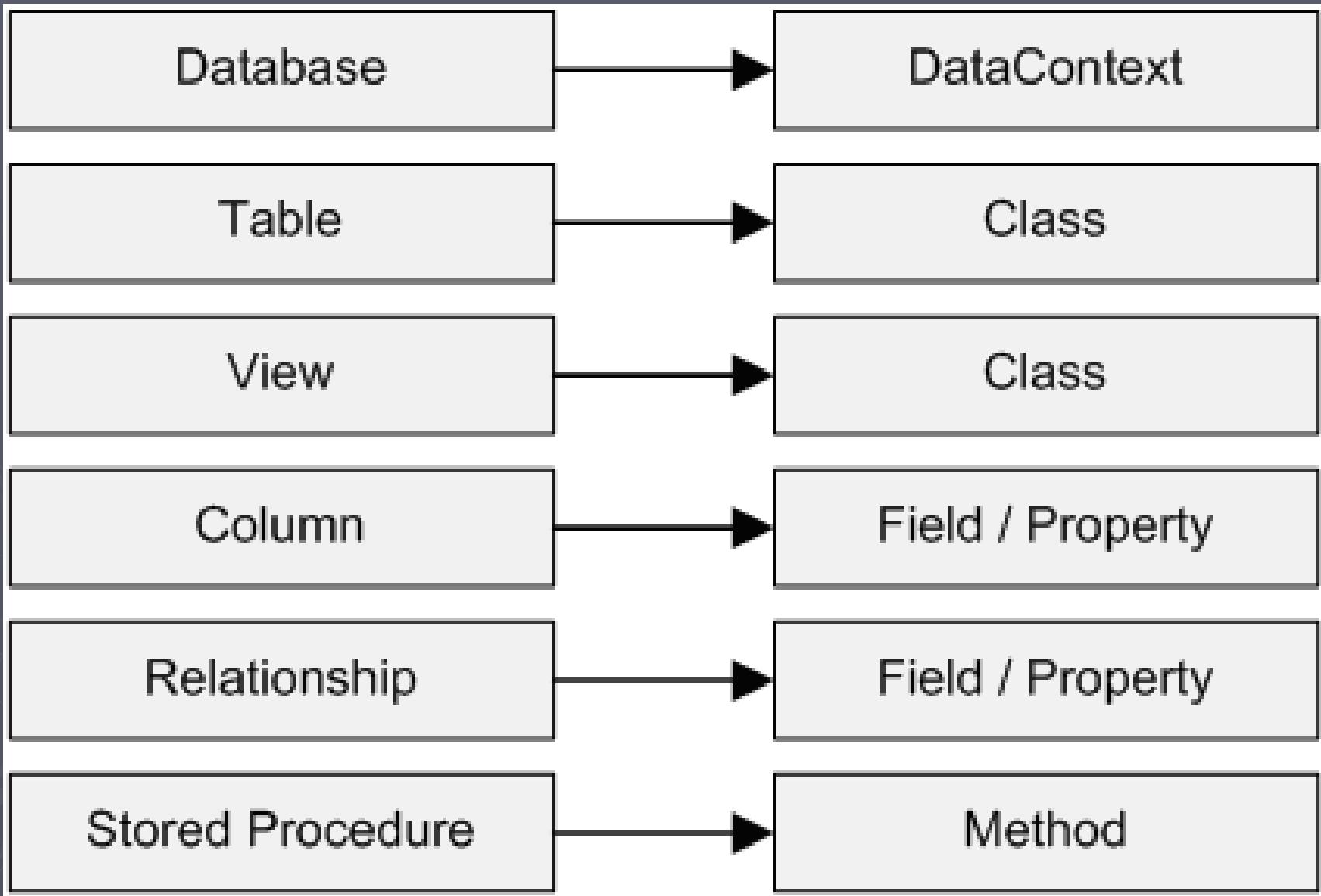
Установлено: 3.0.5 Удалить

Версия: 3.0.5 Обновить

Параметры

Описание

A clone of Microsoft System.Data.Linq to allow multi-DLL extensibility and EF compatibility.



Определение модели и контекста

	Имя	Тип данных	Допустимы значения NULL	По умолч
1	Idcomp	uniqueidentifier	<input type="checkbox"/>	(newid())
2	company	nchar(10)	<input checked="" type="checkbox"/>	
3	processor	nchar(10)	<input checked="" type="checkbox"/>	

```
[Table(Name = "Compy")]
```

```
public class Cmp
{
    [Column(IsPrimaryKey = true, IsDbGenerated = true)]
    public Guid Idcomp { get; set; }
    [Column(Name = "company")]
    public string Company { get; set; }
    [Column]
    public string Processor { get; set; }
}
```

сущность для работы с данными

Модель - обычный класс, который сопоставляется с одной из таблиц в базе данных

создается контекст данных,
который представлен
объектом **DataContext**

```
DataContext db = new DataContext(connectionString);
```

```
// Получаем таблицу пользователей
Table<Cmp> computers = db.GetTable<Cmp>();

var query = from u in db.GetTable<Cmp>()
            where u.Company.Equals("Sony")
            orderby u.Processor
            select u;
result.Items.Add(query.ToString());

foreach (Cmp c in query)
{
    result.Items.Add(c.Idcomp + " " + c.Processor);
}
```

```
SELECT [t0].[Idcomp], [t0].[company] AS [Company], [t0].[Processor]  
FROM [Compy] AS [t0]  
WHERE [t0].[company] = @p0  
ORDER BY [t0].[Processor]
```

d9c9e7cb-c5da-4ae9-85af-b839d454dba6 intel

Изменение объектов добавление и удаление

```
Cmp someComp = db.GetTable<Cmp>().FirstOrDefault();  
// изменим  
someComp.Processor = "Intel Core 13";  
// сохраним изменения  
db.SubmitChanges();
```

эти объекты
кэшируются, и у них
устанавливается
состояние Unchanged

▶ **InsertOnSubmit(newCmp)**

▶ **InsertAllOnSubmit()**

▶ **DeleteOnSubmit(oldCmp)**

▶ **DeleteAllOnSubmit()**

контекст данных
сверяет значения
оригинального объекта
и его измененной
копии. И если два
объекта отличаются, то
создается sql-
выражение UPDATE, с
помощью которого
происходит обновление
объекта в базе данных.

Выполнение кода Sql

```
db.ExecuteNonQuery(" INSERT...");  
ExecuteQuery()
```