



Урок 7

Объектно - Ориентированное Программирование

Введение в базы данных. MS ACCESS. MS SQL. Подключение к базе данных.

[Реляционные базы данных](#)

[ADO.NET](#)

[Объектная модель ADO.NET](#)

[Класс SqlConnectionStringBuilder](#)

[Класс SqlConnection](#)

[Класс SqlCommand](#)

[Класс SqlDataReader](#)

[Класс SqlDataAdapter](#)

[Класс DataTable](#)

[Класс DataRow](#)

[Класс DataSet](#)

[Классы DataSet со строгим контролем типов](#)

[Практика](#)

[Создание локальной базы данных в MS Access](#)

[Подключение к базе данных Access из C# в консоли. Прямой доступ](#)

[Подключение к базе данных Access из C# в консоли. Автономный режим](#)

[Создание локальной базы данных в Visual Studio Community 2015](#)

[Подключение к базе данных с использованием C# \(ADO.NET\)](#)

[Подключение к базе данных из Windows Forms](#)

[Создание источника данных \(DataSet\)](#)

[Написание ежедневника с использованием базы данных \(Windows Forms\)](#)

[Разработка базы данных](#)

[Подключение с использованием DataSet](#)

[Примеры работы с базой данных](#)

[Создание таблицы](#)

[Удаление таблицы](#)

[Обновление базы данных](#)

[Вставка записи в базу данных](#)

[Удаление базы данных](#)

[Пример использования DataSet для получения данных из служб Интернета](#)

[Домашняя работа](#)

[Дополнительные материалы](#)

[Используемая литература](#)

Реляционные базы данных

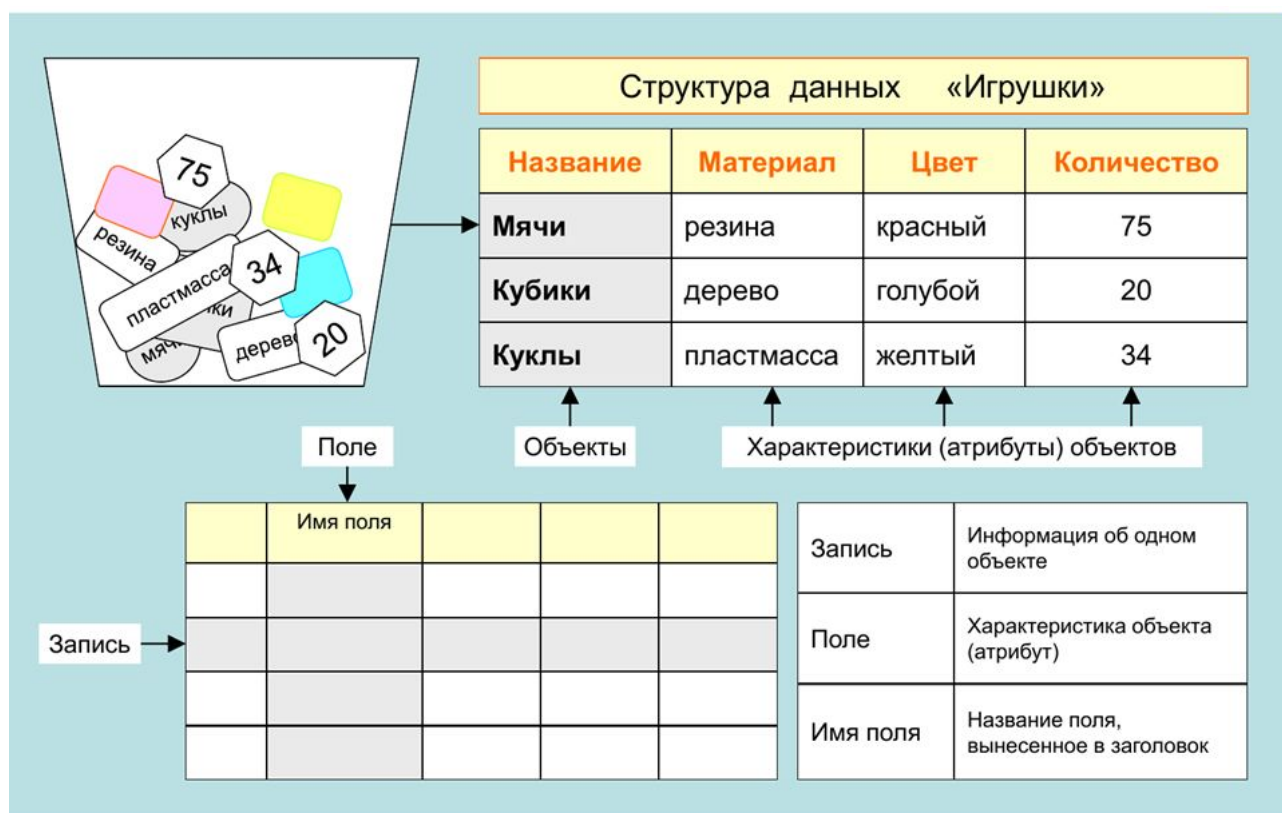
Базой данных (БД) называется организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой её части.

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определённого вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов – атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов).

Строки таблицы называются записями. Все записи таблицы имеют одинаковую структуру – они состоят из полей (элементов данных), в которых хранятся атрибуты объекта (см. рисунок). Каждое поле записи содержит одну характеристику объекта и представляет собой заданный тип данных (например, текстовая строка, число, дата). Для идентификации записей используется первичный ключ. Первичным ключом называется набор полей таблицы, комбинация значений которых однозначно определяет каждую запись в таблице.



Пример структуры реляционной базы данных



Элементы базы данных.

Для работы с данными используются системы управления базами данных (СУБД). Основные функции СУБД:

- определение данных (описание структуры баз данных);
- обработка данных;
- управление данными.

Разработка структуры БД – важная задача, решаемая при проектировании БД. Структура БД (набор, форма и связи ее таблиц) – это одно из основных проектных решений при создании приложений с использованием БД. Созданная разработчиком структура БД описывается на языке определения данных СУБД.

Любая СУБД позволяет выполнять следующие операции с данными:

- добавление записей в таблицы;
- удаление записей из таблицы;
- обновление значений некоторых полей в одной или нескольких записях в таблицах БД;
- поиск одной или нескольких записей, удовлетворяющих заданному условию.

Для выполнения этих операций применяется механизм запросов. Результатом выполнения запросов является либо отобранное по определенным критериям множество записей, либо изменения в таблицах. Запросы к базе формируются на специально созданном для этого языке, который так и называется «язык структурированных запросов» (SQL – Structured Query Language).

Под управлением данными обычно понимают защиту данных от несанкционированного доступа, поддержку многопользовательского режима работы с данными и обеспечение целостности и согласованности данных.

ADO.NET

ADO (ActiveX Data Object) — это набор библиотек, поставляемый с Microsoft .NET Framework и предназначенный для взаимодействия с различными хранилищами данных из .NET-приложений. Библиотеки ADO.NET включают классы, которые служат для подсоединения к источнику данных, выполнения запросов и обработки их результатов. ADO.NET можно использовать и в качестве надёжного иерархически организованного отсоединённого кэша данных для автономной работы с данными.

Главный отсоединённый объект DataSet позволяет сортировать, искать, фильтровать, сохранять отложенные изменения и перемещаться по иерархичным данным.

Кроме того, он включает ряд функций, сокращающих разрыв между традиционным доступом к данным и программированием с использованием XML. Теперь разработчики получили возможность работать с XML-данными через обычные интерфейсы доступа к данным.

Одним словом, библиотеки ADO.NET рекомендуется использовать при создании приложений для работы с данными

Объектная модель ADO.NET

Технология ADO.NET призвана помогать разработке эффективных многоуровневых приложений для работы с БД в интрасетях и Интернете, для чего она и предоставляет все необходимые средства.

Для управления соединением, транзакциями, для выборки данных и передачи изменений они взаимодействуют непосредственно (прямой доступ) с БД. Объекты, указанные в правой части, называются отсоединёнными (автономными); они дают возможность работать с данными автономно.

Подсоединённые классы	Отсоединённые классы
ProviderFactory Connection Transaction ConnectionStringBuilder DataAdapter Command Parameter DataReader	DataSet DataTable DataView DataColumn DataRowView DataRow Constraint DataRelation

Объекты, составляющие отсоединённую (автономную) часть модели ADO.NET, не взаимодействуют напрямую с подсоединёнными объектами

Класс ConnectionStringBuilder

ConnectionStringBuilder — ещё один новый класс в ADO.NET версии 2.0. Этот класс облегчает процесс построения строк подключения для поставщика данных .NET.

Каждый класс ConnectionStringBuilder предоставляет свойства, которые соответствуют опциям, доступным в той самой строке подключения поставщика данных. Например, класс Odbc ConnectionStringBuilder содержит атрибут Driver, а класс SqlConnectionStringBuilder — свойство Provider. Создав строку подключения с помощью класса ConnectionStringBuilder, можно получить доступ к строке подключения средствами свойства ConnectionString класса ConnectionStringBuilder.

Класс Connection

Класс `Connection` предоставляет соединение с источником данных. С помощью свойств этого объекта можно задать тип источника, его расположение и некоторые другие атрибуты. Объект `Connection` применяется для соединения с БД и отсоединения от неё. Объект `Connection` выступает в качестве канала, по которому другие классы, например, `DataAdapter` и `Command`, взаимодействуют с БД для передачи изменений и выборки их результатов.

Класс Command

Объекты `Command` по структуре аналогичны объектам `Command ADO` и `Query Def DAO`. Они могут осуществлять запрос к БД, вызов хранимой процедуры или прямой запрос на возврат содержимого конкретной таблицы.

Базы данных поддерживают множество разных типов запросов. Одни запросы возвращают записи данных, ссылаясь на одну или несколько таблиц или представлений, либо вызывая хранимую процедуру. Другие запросы изменяют записи данных, а все прочие — управляют структурой БД, создавая и изменяя такие объекты, как таблицы, представления и хранимые процедуры. С помощью объекта `Command` удаётся выполнить любой из этих запросов к БД.

Выполнить запрос к БД с использованием объекта `Command` очень просто. Задайте в свойстве `Connection` одноимённый объект, соединяющийся с БД, а затем в свойстве `CommandText` задайте текст запроса. Можно ввести обычный SQL-запрос, например:

```
SELECT CustomerID, CompanyName, ContactName, Phone FROM Customers
```

Вы можете указать имя таблицы, представления или хранимой процедуры и средствами свойства `CommandType` задать тип выполняемого запроса. Класс `Command` даёт возможность выполнять запрос разными способами. Если запрос не возвращает записи, вызовите метод `ExecuteNonQuery`. Кроме того, класс `Command` имеет метод `ExecuteReader`, который возвращает объект `DataReader`, позволяющий просматривать возвращённые запросом записи. Чтобы вернуть только первое поле первой записи, возвращённой запросом, можно сохранить несколько строчек кода, воспользовавшись методом `ExecuteScalar` объекта `Command`. У объекта `SqlCommand` есть ещё и четвёртый метод выполнения, `ExecuteXmlReader`, который аналогичен методу `ExecuteReader`, но предназначен для работы с запросами, возвращающими результаты в формате XML.

Класс DataReader

Класс `DataReader` предназначен для максимально быстрой выборки и просмотра возвращаемых запросом записей. Он позволяет просматривать результаты запроса по одной записи за раз. При переходе к следующей записи содержимое предыдущей отбрасывается. Объект `DataReader` не поддерживает обновление, и возвращаемые им данные доступны только для чтения. Поскольку класс `DataReader` реализует лишь ограниченный набор функций, он очень прост и имеет высокую производительность.

Класс DataAdapter

Класс `DataAdapter` — это своеобразный мост между БД и отсоединёнными объектами модели ADO.NET. Метод `DataAdapter Fill` предоставляет эффективный механизм выборки и переноса результатов запроса

в объект DataSet или DataTable для последующей автономной работы с ними. Кроме того, объекты DataAdapter позволяют передавать отложенные изменения из объектов DataSet в БД.

Класс DataTable

DataTable позволяет просматривать данные в виде наборов записей и столбцов. Чтобы поместить результаты запроса в объект DataTable, применяют метод DataAdapter Fill.

Класс DataRow

Обратиться к реальным значениям, хранящимся в объекте DataTable, позволяет набор Rows, содержащий объекты DataRow. Чтобы просмотреть содержимое конкретного поля определённой записи, воспользуйтесь свойством Item соответствующего объекта DataRow и считайте значение нужного поля. Класс DataRow предоставляет несколько перегруженных определений свойства Item. Можно выбрать поле для просмотра, передав свойству Item объекта DataRow имя, порядковый номер или сопоставленный с полем объект DataColumn. Здесь Item — свойство класса DataRow по умолчанию, поэтому его разрешено использовать неявно.

Класс DataSet

Как следует из имени, объект DataSet содержит набор данных. Его можно рассматривать в качестве контейнера объектов DataTable (хранящихся в наборе Tables объекта DataSet). Помните: предназначение модели ADO.NET — упростить разработку крупных многоуровневых приложений для работы с БД. Иногда требуется обратиться к компоненту на промежуточном сервере и выбрать содержимое нескольких таблиц. Тогда не нужно многократно обращаться к серверу и выбирать данные по одной таблице за раз; вместо этого можно поместить все данные в объект DataSet и вернуть его за один вызов. Тем не менее объект DataSet — это нечто большее, чем просто контейнер объектов DataTable. Данные в нём отсоединены от БД. Все изменения данных просто кэшируются в объектах DataRow. Когда придет время передать эти изменения в БД, то, вероятно, окажется, что неэффективно пересылать промежуточному серверу весь объект DataSet.

Стоит воспользоваться методом GetChanges и извлечь из DataSet лишь изменённые данные. В результате уменьшится объём данных, передаваемых между процессами и серверами.

Классы DataSet со строгим контролем типов

Visual Studio.NET упрощает разработку приложений для доступа к данным, генерируя объекты DataSet со строгим контролем типов. Если у вас, предположим, имеется простая таблица Orders со столбцами CustomerID и CompanyName, то вам не потребуется писать код, аналогичный приведенному ниже.

```
DataSet ds;  
// Создаём и заполняем объект DataSet  
Console.WriteLine(ds.Tables["Customers"].Rows[0]["CustomerID"]);
```

Объект DataSet со строгим контролем типов — обыкновенный класс, создаваемый Visual Studio .NET и предоставляющий информацию о таблицах и столбцах посредством свойств.

Вместо этого можно написать такой код.

```
CustomersDataSet ds;  
// Создаём и заполняем объект DataSet  
Console.WriteLine(ds.Customers[0].CustomerID);
```

Практика

Создание локальной базы данных в MS Access

Запустите Ms Access и создайте новую базу данных. Переключите режим:



Вам предложат сохранить таблицу. Введите название таблицы:

Сохранение

Имя таблицы:

Ежедневник

OK Отмена

Задайте таблице поля, как показано ниже:

Ежедневник	
Имя поля	Тип данных
Код	Счетчик
Дата	Дата и время
Текст	Короткий текст

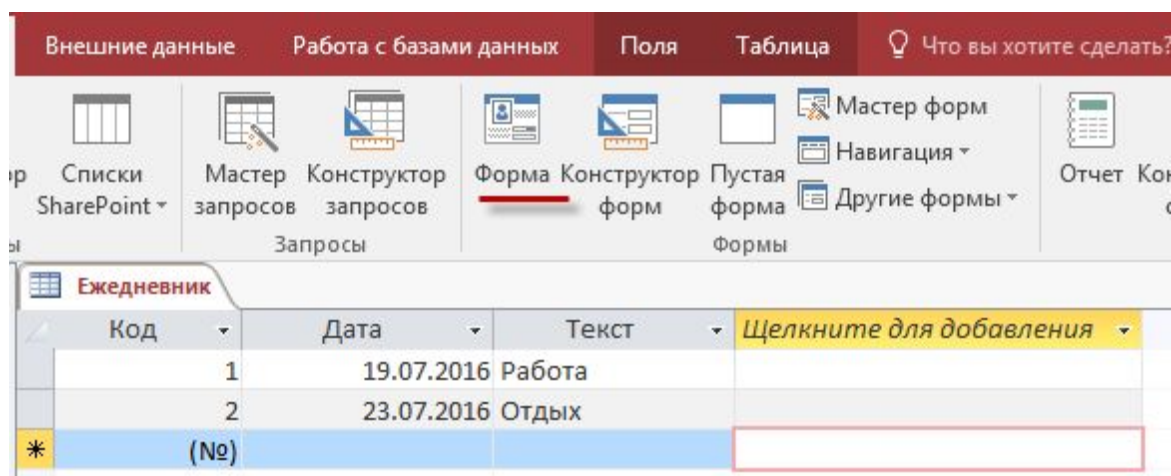
Нажмите на режим и сохраните получившуюся таблицу:



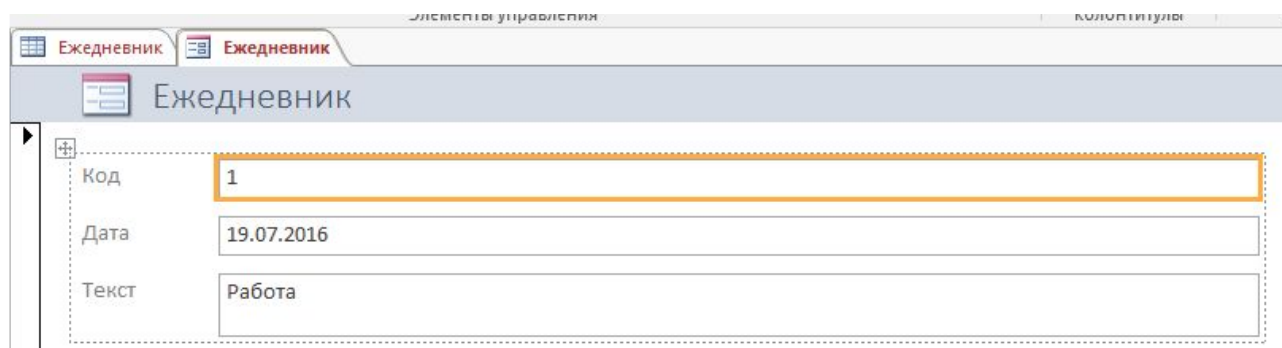
Введите в таблицу произвольные данные:

Ежедневник			
Код	Дата	Текст	Щелкните для добавления
1	19.07.2016	Работа	
2	23.07.2016	Отдых	
* (№)			

Для удобства ввода данных полезно создать форму. Выберите вкладку “Создание” и нажмите кнопку “Форма”:



Среда проанализирует вашу таблицу и создаст форму для ввода данных.



Закрываем базу данных и находим получившийся файл на диске.

Подключение к базе данных Access из C# в консоли. Прямой доступ

Создайте консольное приложение и скопируйте туда этот текст. Базу данных подключите к проекту или просто скопируйте её в папку bin. В примере база данных называется “Database5.accdb” и располагается в папке bin проекта. У вас она может называться по-другому. В этом случае поменяйте название в параметре Data Source переменной connectionString.

```

using System;
using System.Data.OleDb;
namespace AccessConsole
{
    class Program
    {
        static void Main(string[] args)
        {
            // Строка подключение
            string connectionString = @"Provider = Microsoft.ACE.OLEDB.12.0;Data Source = Database5.accdb";
            // Объект подключение к базе данных
            OleDbConnection connection = new OleDbConnection(connectionString);
            connection.Open();
            OleDbCommand command = new OleDbCommand("Select * from Ежедневник", connection);
            OleDbDataReader reader = command.ExecuteReader();
            while (reader.Read())//Считываем каждую строку
            {
                Console.WriteLine("{0}\t{1}\t{2}", reader.GetInt32(0), reader.GetDateTime(1), reader.GetString(2));
            }
            connection.Close();
            Console.ReadKey();
        }
    }
}

```

В этом примере используется прямой доступ к базе данных. База данных постоянно открыта, пока мы с ней работаем. Подключение к базе данных Access из C# в консоли. Автономный режим

Другой способ позволяет предварительно считать данные из базы данных в память компьютера. Внести изменения автономно, а потом обновить базу данных.

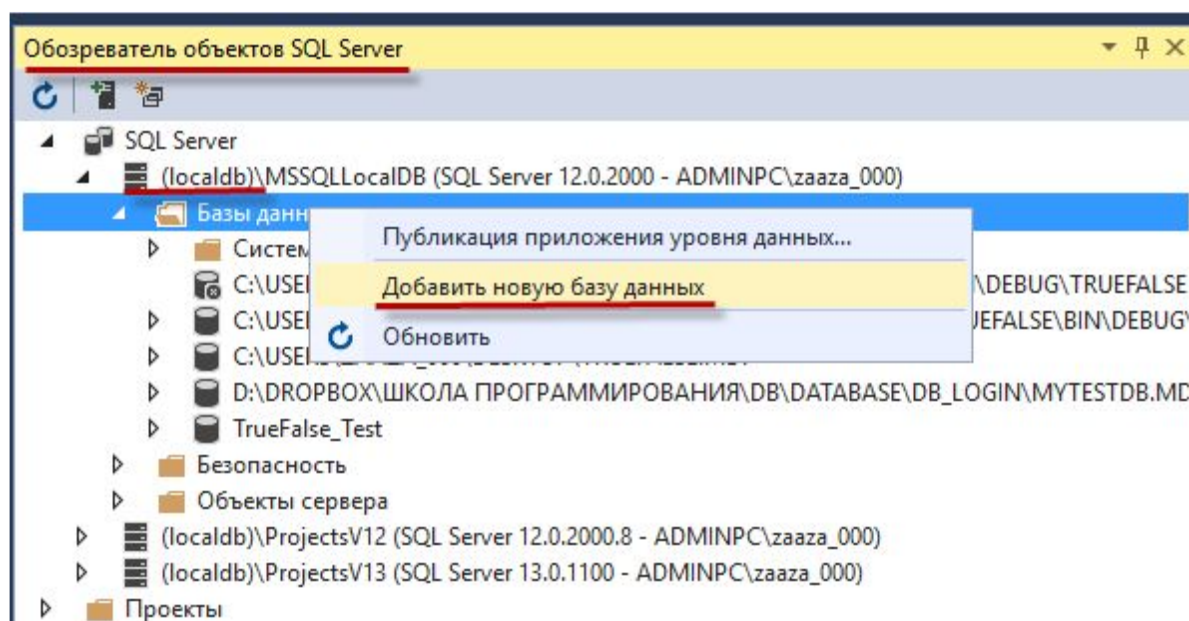
```

using System;
using System.Data;
using System.Data.OleDb;
namespace AccessConsoleDataSet
{
    class Program
    {
        static void Main(string[] args)
        {
            // Строка подключение
            string connectionString = @"Provider = Microsoft.ACE.OLEDB.12.0;Data Source = Database5.accdb";
            //Объект-промежуточный адаптер между базой данных и набором данных(dataSet)
            OleDbDataAdapter dataAdapter = new OleDbDataAdapter("Select * from Ежедневник", connectionString);
            DataSet dataSet = new DataSet();
            // Загружаем данные из базы данных
            dataAdapter.Fill(dataSet);
            foreach (DataRow dataRow in dataSet.Tables[0].Rows)
            {
                Console.WriteLine("{0}\t{1}\t{2}", (int)dataRow[0], (DateTime)dataRow[1], (string)dataRow[2]);
            }
            Console.ReadKey();
        }
    }
}

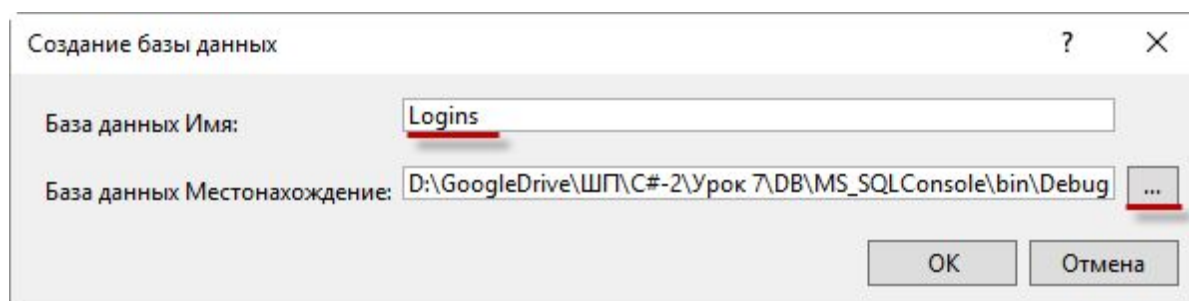
```

Создание локальной базы данных в Visual Studio Community 2015

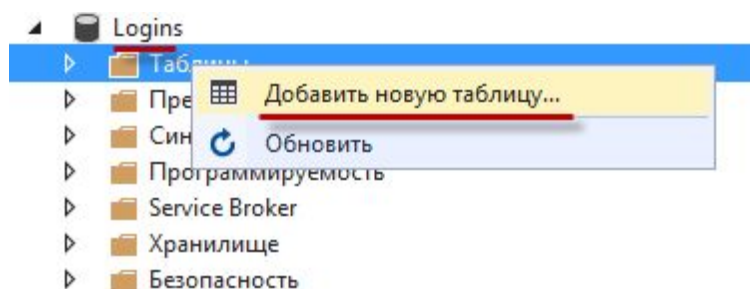
Запустите Visual Studio и выберите “Обозреватель объектов SQL Server”. Далее выберите “сервер баз данных” и добавьте новую базу данных.




Выберите месторасположение и имя базы данных.



Добавьте в созданной базе данных новую таблицу.



В новой таблице добавьте три поля intId, nvcLogin и nvcPassoword.

Обновить		Файл скрипта:	dbo.Table.sql*
	Имя	Тип данных	Допустимы значения
	intId	int	<input type="checkbox"/>
	nvcLogin	nvarchar(50)	<input checked="" type="checkbox"/>
	nvcPassword	nvarchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Для поля `intId` установите свойство Идентификатор в `True`.

Свойства

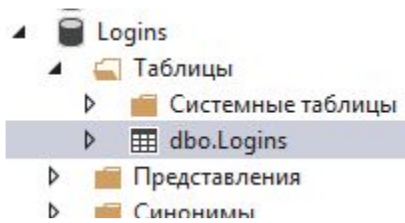
intId Столбец

(Имя)	intId
Допускает значения NULL	False
Значение или привязка по умолчанию	
Набор столбцов	False
Не для репликации	False
Описание	
Параметры сортировки	
Первичный ключ	True
Разреженный	False
<input checked="" type="checkbox"/> Спецификации вычисляемого столбца	
<input checked="" type="checkbox"/> Спецификация идентификатора	True
(Идентификатор)	True
Начальное значение идентификатора	1
Шаг приращения идентификатора	1
<input checked="" type="checkbox"/> Спецификация полнотекстового поиска	False
Тип данных	int
Файловый поток	False
Является столбцом ROWGUID	False

У вас должен получиться вот такой скрипт:

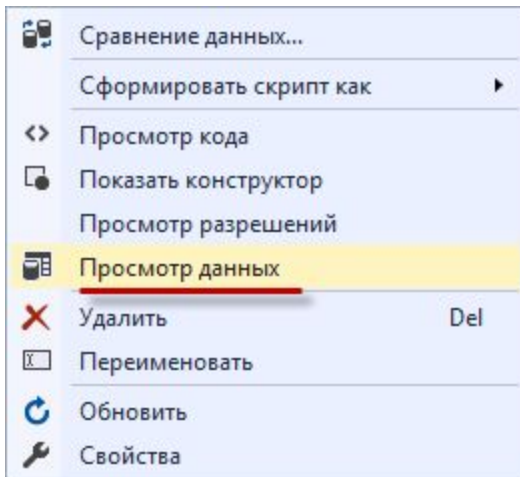
```
CREATE TABLE [dbo].[Logins] (
    [intId] INT IDENTITY (1, 1) NOT NULL,
    [nvcLogin] NVARCHAR (50) NULL,
    [nvcPassword] NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([intId] ASC)
);
```

Нажимаем кнопку “Обновить” и “Обновить базу данных”. В папке таблиц появится новая таблица.



Префикс dbo - означает DataBase Owner - владелец базы данных. Этот префикс отделяет пользовательские таблицы от системных таблиц.

Далее щёлкните правой кнопкой мыши на таблице и выберите "Просмотр данных".



Заполните таблицу данными. intId заполнится автоматически.

dbo.Table [Данные] X dbo.Table [Конструктор] Pro			
Максимальное количество строк: 100			
	intId	nvcLogin	nvcPassword
	1	Admin	Admin
	2	Ivanov	123456
	3	Sidorov	654321
▶*	NULL	NULL	NULL

Создаем консольный проект, в котором подключимся к базе данных. Но для начала щёлкните на имени базы данных правой кнопкой мыши, выберите свойства и из свойств скопируйте куда-нибудь строку подключения.

Сервер	(localdb)\MSSQLLocalDB
Срок хранения	2
Строка подключения	Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Logins;Integrated Security=True;Connect Timeout=30;Enc
Уровень совместимости	120

Подключение к базе данных с использованием C# (ADO.NET)

Для начала продемонстрируем, как можно подключиться к базе данных и вывести её содержимое на экран консольного окна:

```
Таблицы в dataSet
Table
Содержимое таблицы Logins
1      Admin   Admin
2      Ivanov  123456
3      Sidorov 654321
```

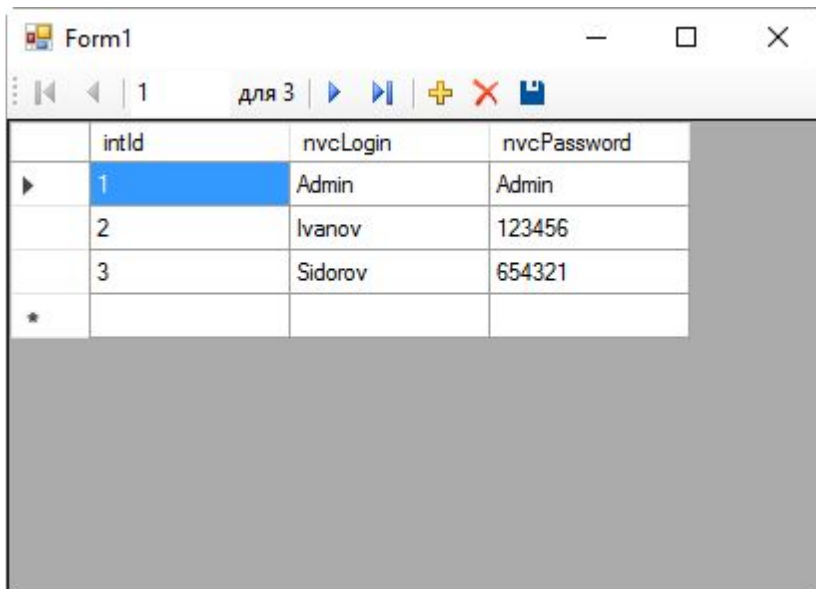
Код в качестве connectionString вставьте строку подключения, которую вы сохранили на предыдущем шаге.

```
using System;
using System.Data;
using System.Data.SqlClient;
namespace MS_SQLConsole
{
    class Program
    {
        static void Main(string[] args)
        {
            // Строка подключения
            string connectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=D:\TEMP\LOGINS.MDF;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=F
alse";
            // Объект-промежуточный адаптер между базой данных и набором данных (dataSet)
            SqlDataAdapter dataAdapter = new SqlDataAdapter("Select * from Logins", connectionString);
            DataSet dataSet = new DataSet();
            // Загружаем данные из базы данных
            dataAdapter.Fill(dataSet);
            // Выведем названия таблиц
            Console.WriteLine("Таблицы в dataSet");
            foreach (var table in dataSet.Tables)
            {
                Console.WriteLine(table);
            }
            // Выведем содержимое таблицы Logins
            Console.WriteLine("Содержимое таблицы Logins");
            // В DataSet таблица называется Table
            foreach (DataRow dataRow in dataSet.Tables["Table"].Rows)
            {
                Console.WriteLine("{0}\t{1}\t{2}", (int)dataRow[0], (string)dataRow[1], (string)dataRow[2]);
                Console.ReadKey();
            }
        }
    }
}
```


Подключение к базе данных из Windows Forms

Давайте посмотрим, как подключиться к базе данных из Windows Forms. Что самое примечательное, для этого нам не придётся писать ни одной строчки кода!

Для этого воспользуемся созданной базой данных Logins:



	intId	nvcLogin	nvcPassword
▶	1	Admin	Admin
	2	Ivanov	123456
	3	Sidorov	654321
*			

Для начала создайте проект Windows Forms под названием Logins. После создания приложения необходимо подключить источник данных, при этом будет создан набор данных со строгой типизацией.

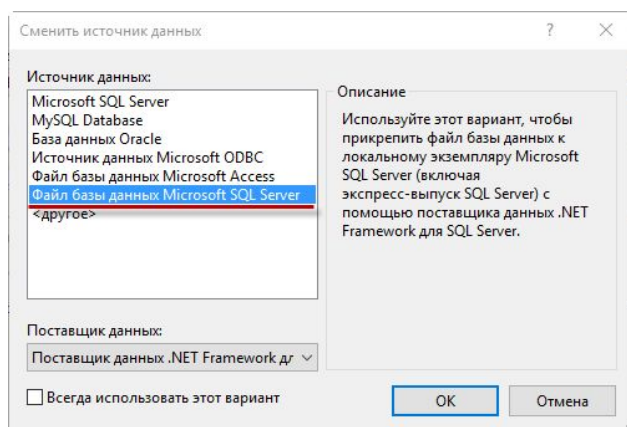
Создание источника данных (DataSet)

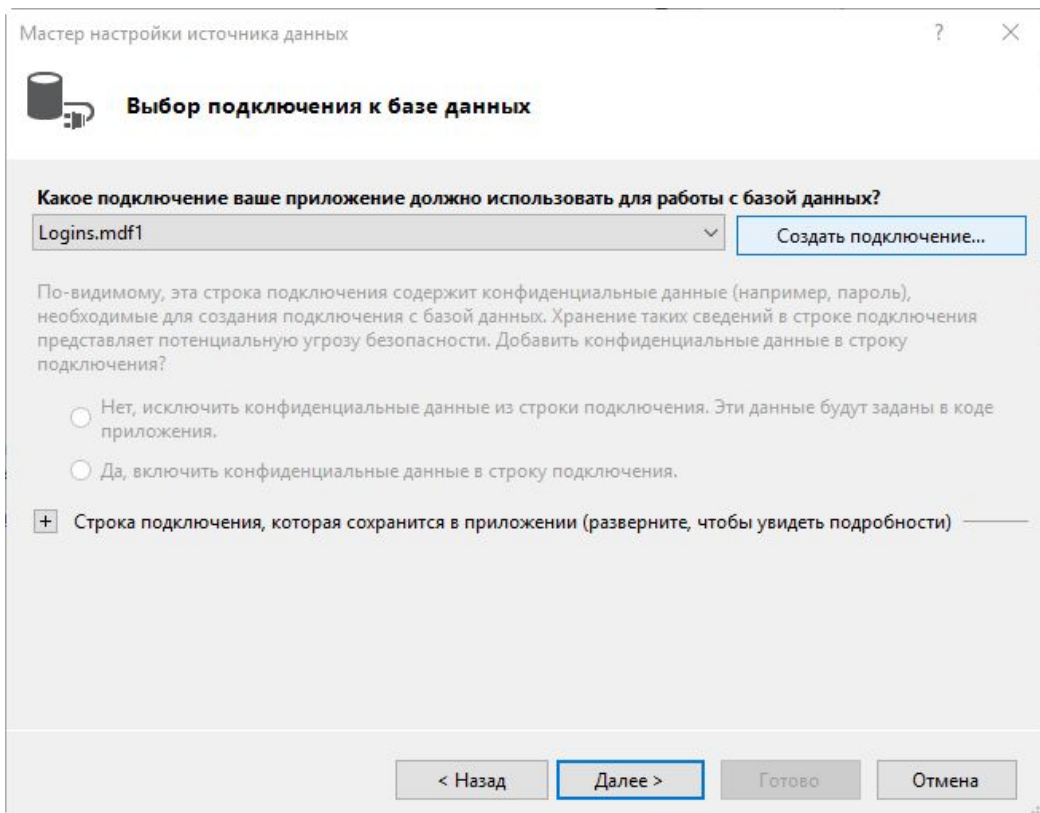
Выведите панель “Источники данных” (“Вид-Другие окна-Источники данных”) или Ctrl+Alt+D. Щёлкните на “Добавить новый источник данных”



Выберите “Базы данных-Набор данных-Создать подключение...”

В окне “Добавить подключение” напротив поля “Источник данных” нажмите кнопку “Изменить...” и выберите “Файл базы данных Microsoft SQL Server”. Нажмите “ОК”.

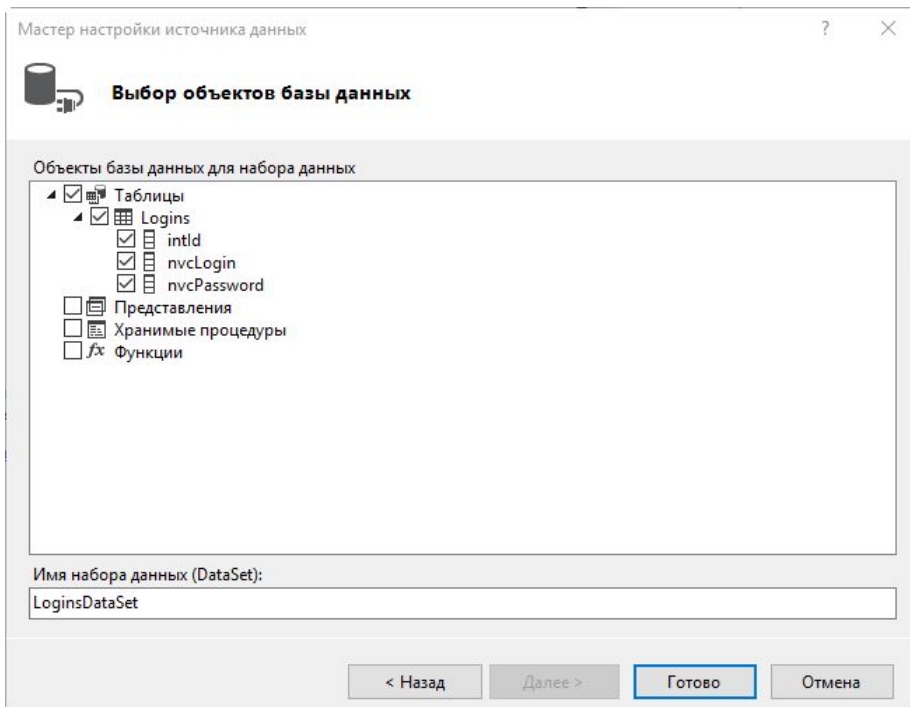




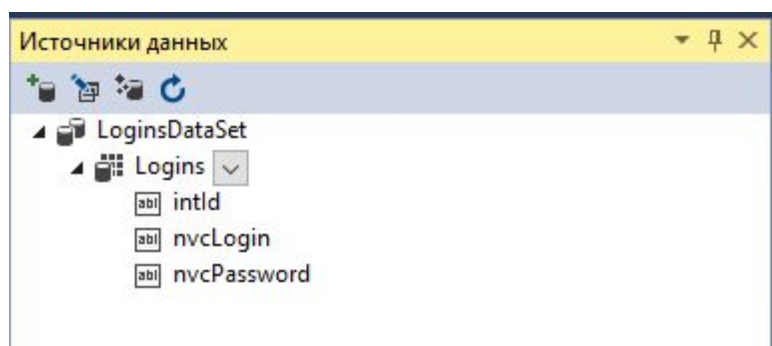
На вопрос, нужно ли копировать в выходной каталог, ответьте "Нет".

Создается подключение и вам предложат сохранить строку подключения в файле проекта. Нажмите "Далее".

Выберите Таблицу Logins и все поля таблицы.

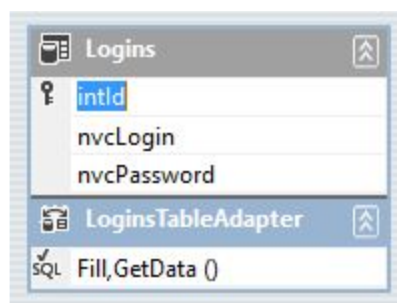


В итоге будет создан так называемый строго типизированный DataSet, который визуально можно увидеть в Источнике данных

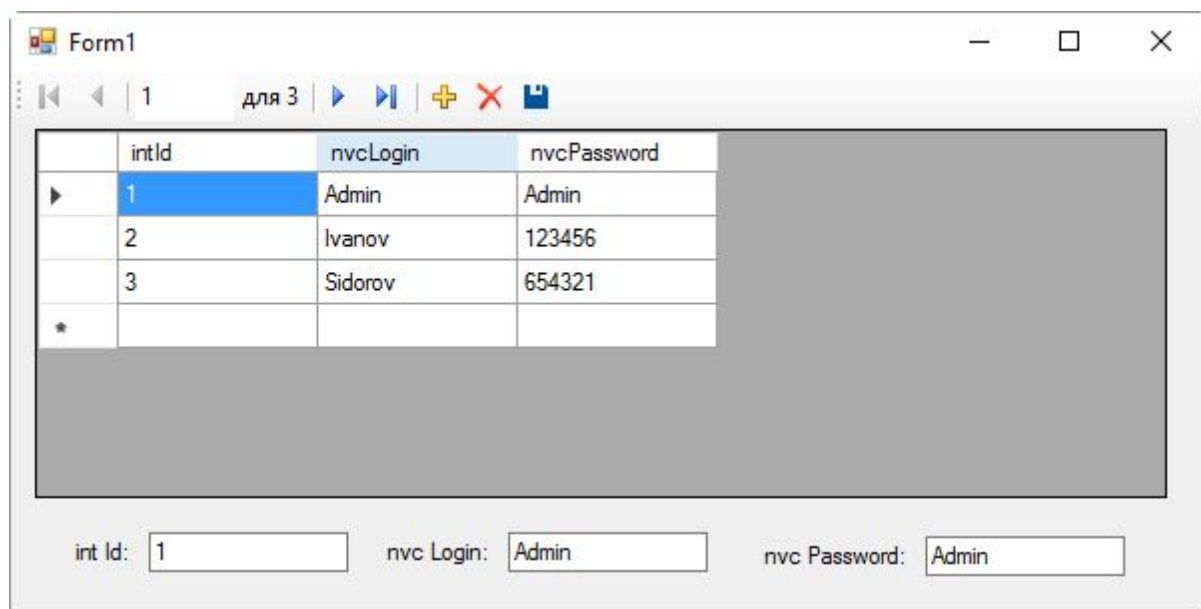


Типизированный DataSet - это класс, который описывает источник данных созданный автоматически.

Если щёлкнуть на LoginsDataSet.xsd, то VS покажет графическое отображение этого класса.



Для завершения нам достаточно перетащить таблицу и все поля из источника данных на форму. При этом VS автоматически сгенерирует код для работы с базой данных.



Написание ежедневника с использованием базы данных (Windows Forms)

Разработка базы данных

В нашем случае это довольно простая база, которая будет содержать всего три поля id, Text и Date.

- id - идентификационный номер записи целое число;
- Text - строка с заметкой;
- Date - дата заметки.

Обновить	Файл скрипта:	dbo.Table.sql*		
	Имя	Тип данных	Допустимы значения NULL	По умолчанию
	intId	int	<input type="checkbox"/>	
	nvcText	nvarchar(MAX)	<input checked="" type="checkbox"/>	
	dtDate	datetime	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Скрипт для создания базы данных:

```
CREATE TABLE [dbo].[Diary]
(
    [intId] INT NOT NULL PRIMARY KEY IDENTITY,
    [nvcText] NVARCHAR(MAX) NULL,
    [dtDate] DATETIME NULL
)
```

Нажимаем “Обновить”.

Выбираем нашу таблицу и в контекстном меню щёлкаем на “Посмотреть данные”

Заполним таблицу произвольными данными.

	intId	nvcText	dtDate
	1	Работа	21.07.2016 0:00:...
	2	Работа	22.07.2016 0:00:...
►*	NULL	NULL	NULL

И нажмём кнопку “Обновить”.

Переходим к нашему приложению. Создадим класс, который будет считывать данные. Конечно проще, если данные будут считываться сразу же в том формате, в котором нам необходимо.

Подключение с использованием DataSet

Загрузите проект, созданный на предыдущем уроке. Удалите из него ранее созданную таблицу grid и добавьте в загрузчик создание собственной таблицы.

```
// TODO: данная строка кода позволяет загрузить данные в таблицу "diaryDBDataSet.DiaryTable". При
// необходимости она может быть перемещена или удалена.
this.diaryTableTableAdapter.Fill(this.diaryDBDataSet.DiaryTable);
DataGridView grid = new DataGridView();
grid.Parent = tabPage1;
grid.AutoGenerateColumns = false;
grid.DataSource = diaryTableBindingSource ;
grid.Left = 350;
grid.Width = 300;
grid.Top = 100;
DataGridViewTextBoxColumn tbColumn = new DataGridViewTextBoxColumn();
tbColumn.DefaultCellStyle.NullValue = "Пусто";
grid.Columns.Add(tbColumn);
grid.Columns[0].DataPropertyName = "nvcMemo";
CalendarColumn calColumn = new CalendarColumn();
calColumn.DefaultCellStyle.NullValue = "01.01.01";
grid.Columns.Add(calColumn);
grid.Columns[1].DataPropertyName = "dtDate";
```

Добавим пункт меню в меню File Update database. И добавим обработчик события.

```
private void tsmiUpdateDatabase_Click(object sender, EventArgs e)
{
    diaryTableTableAdapter.Update(diaryDBDataSet.DiaryTable);
}
```

Для обновления данных используем созданный системой объект TableAdapter и передадим ему таблицу из объекта diaryDBDataSet.

Обработчик Сохранить изменится на:

```
private void tsmiSaveXML_Click(object sender, EventArgs e)
{
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.Filter = "Файл xmlMemo|*.xmlMemo";
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        diaryDBDataSet.WriteXml(dlg.FileName);
    }
}
```

Обработчик загрузки тоже изменится:

```
private void tsmiLoadXML_Click(object sender, EventArgs e)
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "Файл xmlMemo[*].xmlMemo";
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        diaryDBDataSet.Clear();
        diaryDBDataSet.ReadXml(dlg.FileName);
    }
}
```

Примеры работы с базой данных

Во всех этих примерах используется строка подключения, сформированная средой VS и подключаемая как свойство настроек проекта:

```
string connectionString = Properties.Settings.Default.DiaryDBConnectionString;
```

Вы должны заменить её на свою строку подключения.

Создание таблицы

```
private void tsmiCreateTable_Click(object sender, EventArgs e)
{
    string connectionString = Properties.Settings.Default.DiaryDBConnectionString;
    System.Data.SqlClient.SqlConnection connection = new
    System.Data.SqlClient.SqlConnection(connectionString);
    System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand(
        "CREATE TABLE[dbo].[Diary2]"
        +"("
        +"[intId] INT NOT NULL PRIMARY KEY IDENTITY,"
        +"[nvcText] NVARCHAR(MAX) NULL,"
        +"[dtDate] DATETIME NULL)", connection);
    connection.Open();
    command.ExecuteReader();
    connection.Close();
}
```

Удаление таблицы

```
private void tsmiDeleteTable_Click(object sender, EventArgs e)
{
    string connectionString = Properties.Settings.Default.DiaryDBConnectionString;
    System.Data.SqlClient.SqlConnection connection = new
    System.Data.SqlClient.SqlConnection(connectionString);
    System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand("DROP
    TABLE Diary2", connection);
    connection.Open();
    command.ExecuteNonQuery();
    connection.Close();
}
```

Обновление базы данных

```
private void tsmiUpdateDatabase_Click(object sender, EventArgs e)
{
    string connectionString = Properties.Settings.Default.DiaryDBConnectionString;
    System.Data.SqlClient.SqlConnection connection = new
    System.Data.SqlClient.SqlConnection(connectionString);
    System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand("Select * from
    DiaryTable", connection);
    connection.Open();
    foreach (var el in memoFile.Memos)
    {
        string commandStr = "Update DiaryTable Set nvcMemo=" + el.Text + " where intId=" + el.Id;
        command.CommandText = commandStr;
        System.Data.SqlClient.SqlDataReader reader = command.ExecuteReader();
    }
    connection.Close();
}
```

Вставка записи в базу данных

```
private void tsmiInsertInDatabase_Click(object sender, EventArgs e)
{
    string connectionString = Properties.Settings.Default.DiaryDBConnectionString;
    System.Data.SqlClient.SqlConnection connection = new
    System.Data.SqlClient.SqlConnection(connectionString);
    System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand("INSERT
    INTO DiaryTable (nvcMemo, dtDate) VALUES ('Work','01.01.2017')", connection);
    connection.Open();
    command.ExecuteNonQuery();
    connection.Close();
}
```

Удаление базы данных

```
private void tsmiDeleteDatabase_Click(object sender, EventArgs e)
{
    string connectionString = Properties.Settings.Default.DiaryDBConnectionString;
    System.Data.SqlClient.SqlConnection connection = new
    System.Data.SqlClient.SqlConnection(connectionString);
    System.Data.SqlClient.SqlCommand command = new
    System.Data.SqlClient.SqlCommand("DROP DATABASE DiaryDB", connection);
    connection.Open();
    connection.Close();
}
```

Пример использования DataSet для получения данных из служб Интернета

Разместите на форме DataGridView с именем gridInfo и кнопку GetData.

Подключите веб-службу <http://www.cbr.ru/dailyinfowebsew/dailyinfo.asmx>

На обработчик кнопки добавьте следующий код:

```
private void btnGetData_Click(object sender, EventArgs e)
{
    DataSet dataSetInfo = new DataSet();
    var info = new ru.cbr.www.DailyInfo();
    dataSetInfo = info.BiCurBacket();
    gridInfo.DataSource = dataSetInfo.Tables[0];
}
```

Домашняя работа

1. Создать игру “Верю-Не верю” с использованием MS SQL LocalDB.
 - а) Первая программа, которая позволяет работать со списком вопросов;
 - б) *Вторая программа, которая позволяет играть. Задаётся 5 вопросов и подсчитывается, сколько человек ответило правильно, а сколько неправильно.

Дополнительные материалы

1. [DataGridView. Новый контрол в составе Framework 2.0](#)
2. [РАБОТА С БАЗАМИ ДАННЫХ НА ЯЗЫКЕ C#. ТЕХНОЛОГИЯ ADO .NET. Учебное пособие](#)
3. [SQL Server 2014 Express LocalDB](#)

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. “Программирование с использованием Windows Forms”, Петцольд Чарльз, 2006 г.
2. “Visual C# 2012 на примерах”, Зиборов В.В., БВХ-Петербург, 2013
3. “Язык программирования C# 5.0 и платформа .NET 4.5”. Эндрю Троелсен, Питер 2013 г.
4. Привязка данных в Windows Forms. Программирование клиентских приложений обработки данных на платформе .NET, Брайан Нойес, Бином, 2009
5. Transact-SQL. Наиболее полное руководство в подлиннике. Михаил Фленов, “БХВ-Петербург”, 2006
6. [MSDN](#)