

- ▶ Переход на другую форму (Form2) по нажатию на кнопку (Button1).

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 a = new Form2();
    a.ShowDialog();
}
```

- ▶ Подключить музыку (только в формате wav) при открытии формы.
- ▶ Допустим файл с именем ring (должен находиться внутри проекта, там где программный код.
- ▶ Скачиваем библиотеку winmm.dll.
- ▶ Пишем код.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace everyday
{
    public partial class Form5 : Form
    {
        // функция PlaySound, обеспечивающая воспроизведение wav-файлов, находится в библиотеке winmm.dll
        // подключим эту библиотеку
        [System.Runtime.InteropServices.DllImport("winmm.dll")]
        private static extern Boolean PlaySound(string lpszName, int hModule, int dwFlags);

        WMPLib.WindowsMediaPlayer WMP = new WMPLib.WindowsMediaPlayer();

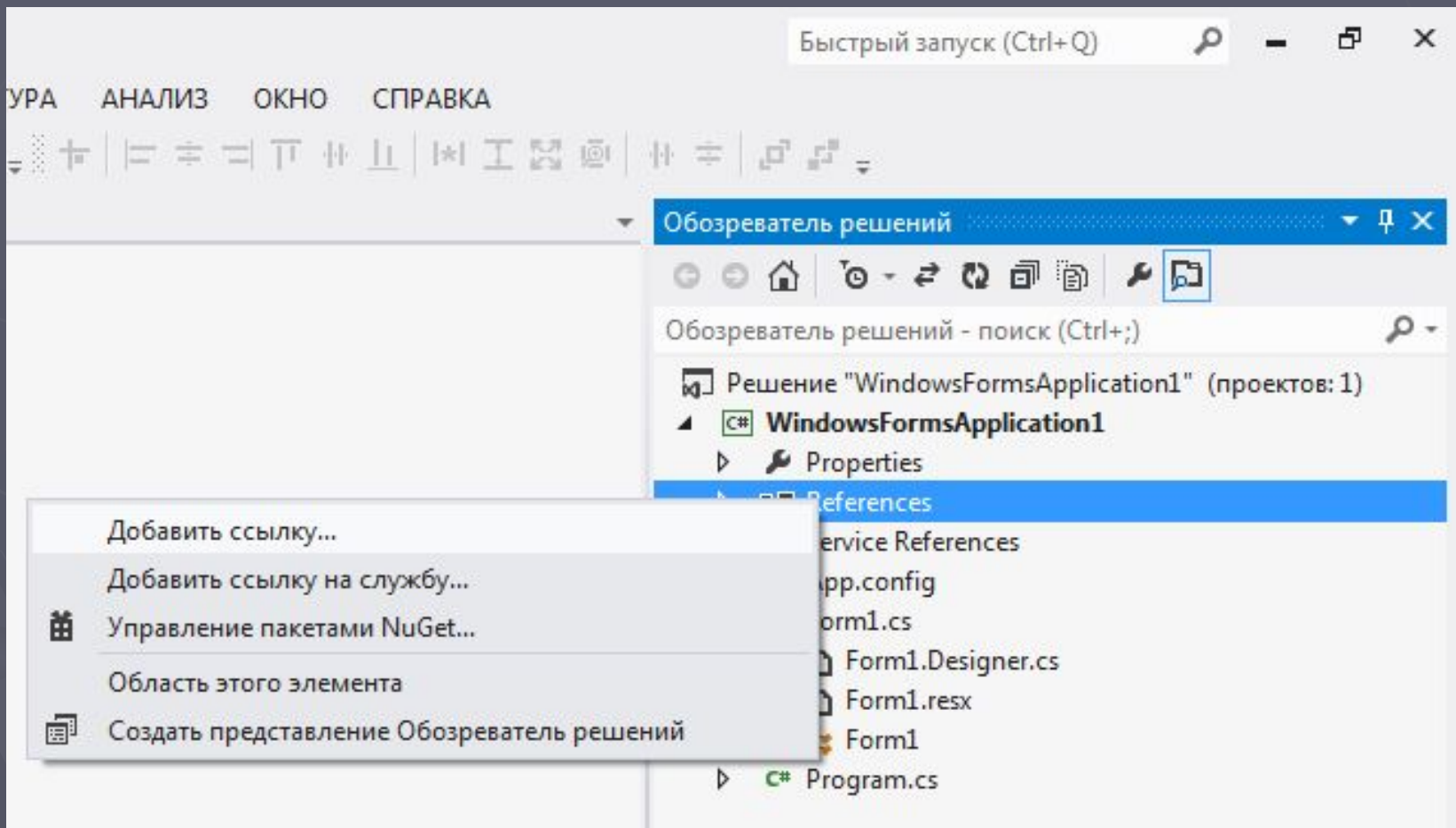
        public Form5()
        {
            InitializeComponent();
        }
        private void Form5_Load(object sender, EventArgs e)
        {
            WMP.URL = "ring.mp3";
            WMP.controls.play();
        }

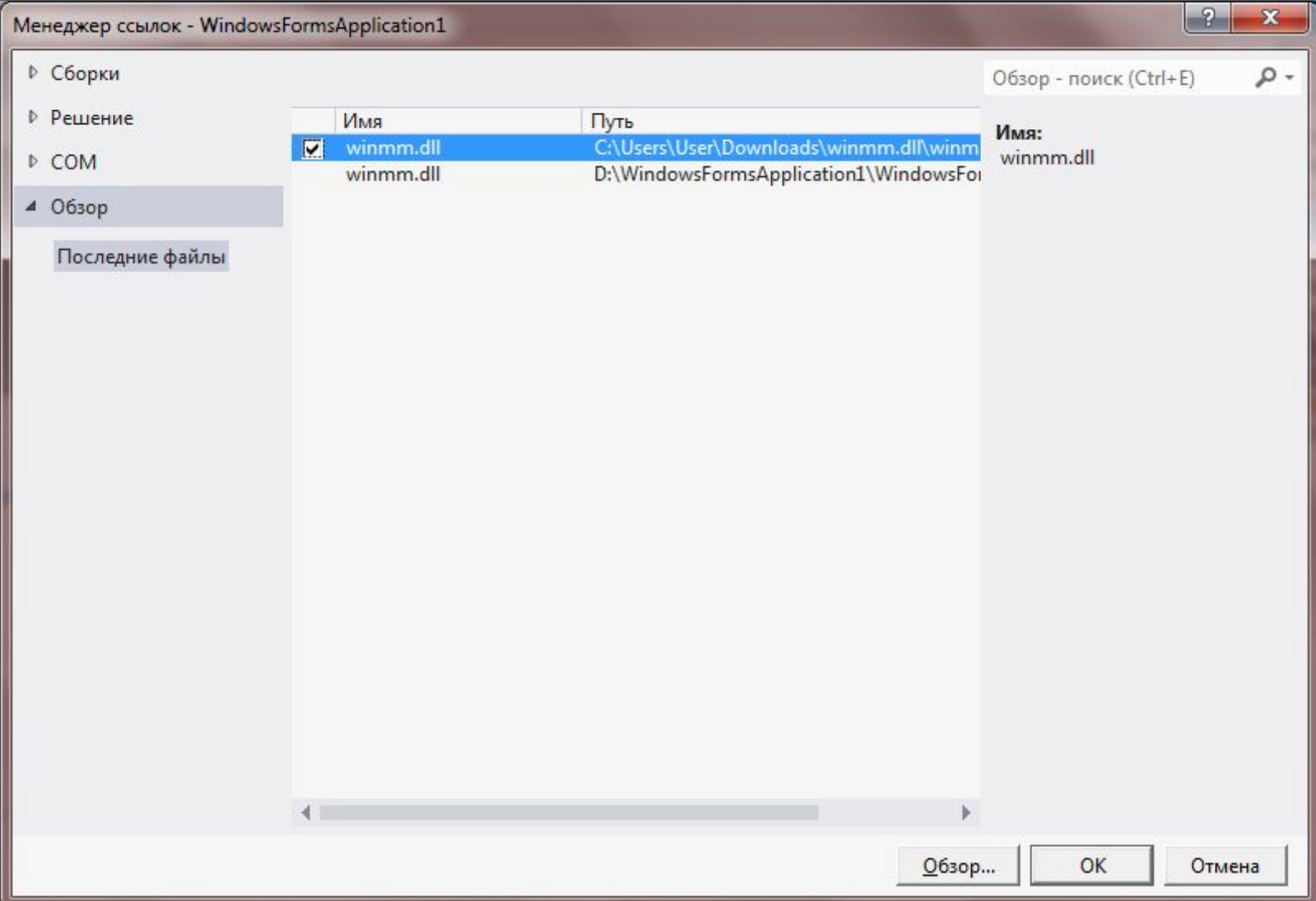
        public void button1_Click(object sender, EventArgs e)
        {
            WMP.controls.stop();
            WMP.close();
        }
    }
}
```

Скачать библиотеку можно здесь

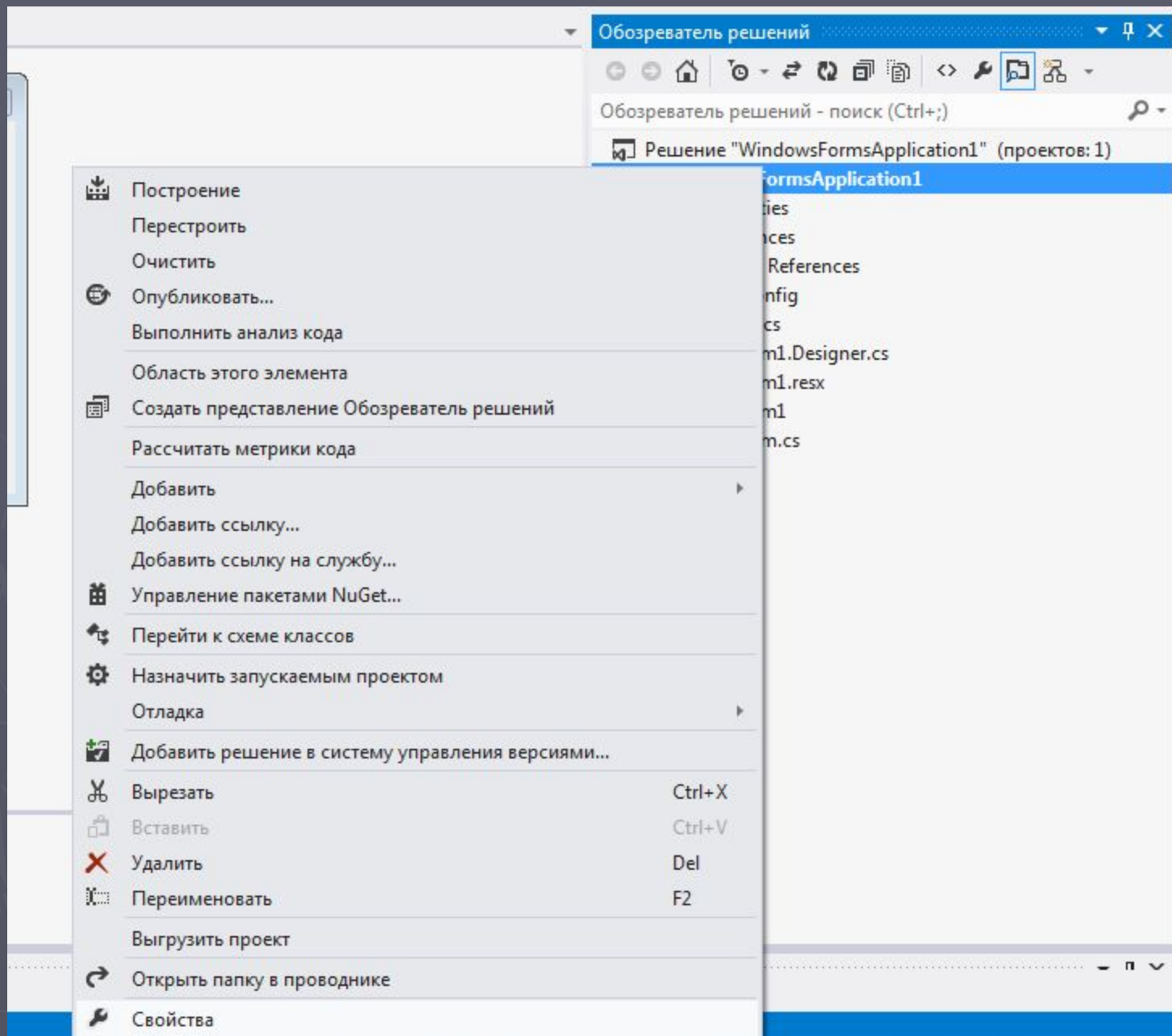
<http://www.dll.ru/files/winmm-dll.html>

► Подключаем библиотеку в Visual Studio.





► Публикация приложения



- Приложение
- Построение
- События построения
- Отладка
- Ресурсы
- Службы
- Параметры
- Пути для ссылок
- Подписывание
- Безопасность
- Публикация**
- Анализ кода

Конфигурация: Н/Д Платформа: Н/Д

Место публикации

Местоположение каталога публикации (веб-узел, ftp-сервер или путь к файлу):

D:\ ...

URL-адрес каталога установки (если отличается от вышеприведенного):

...

Режим установки и параметры

- Приложение доступно только из сети
- Приложение будет доступно в автономном режиме (можно будет запустить из меню "Пуск")

- Файлы приложения...
- Необходимые компоненты...
- Обновления...
- Параметры...

Версия публикации

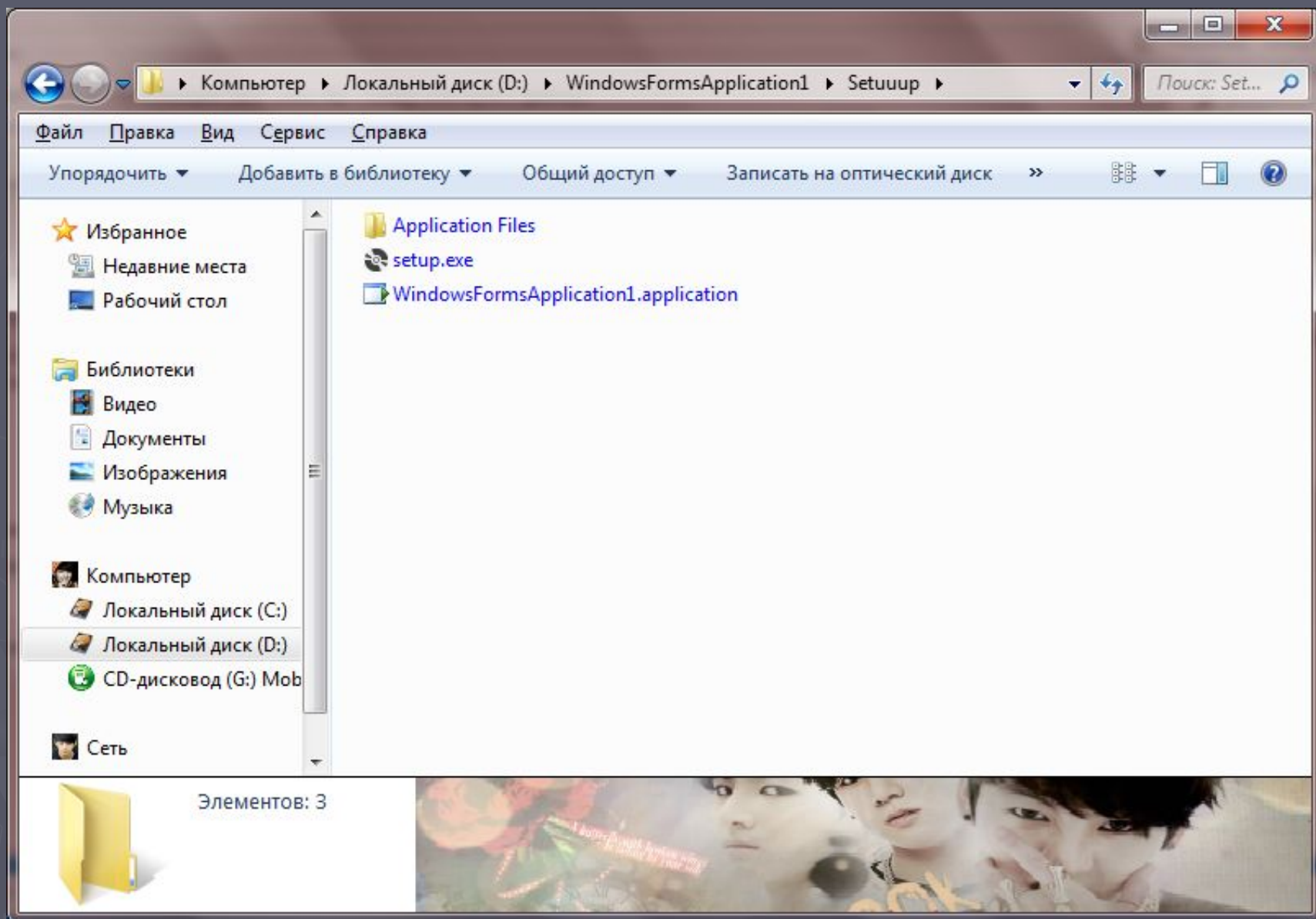
Основной номер: 1 Дополнительный номер: 0 Построение: 0 Редакция: 1

Автоматически увеличивать номер редакции после каждой публикации

- Мастер публикации...
- Опубликовать сейчас

Обозреватель

- Обозреватель
- Решение
- WindowsFormsApplication1
- Form1.cs



Установка приложения - Предупреждение о безопасности



Издатель не может быть проверен.
Вы уверены, что хотите установить данное приложение?



Имя:

WindowsFormsApplication1

От (наведите указатель на строку ниже, чтобы увидеть полное имя домена):

D:\WindowsFormsApplication1\Setuuup

Издатель:

Неизвестный издатель

Установить

Не устанавливать



Хотя приложения могут быть полезными, они потенциально могут повредить ваш компьютер. Если вы не доверяете этому источнику, не устанавливайте эту программу.

[Дополнительная информация...](#)

Приложения с базами данных

Лекция 2

Привязка данных средствами Windows Forms

- ▶ класс Binding

синхронизация между свойством элемента управления и свойством источника данных

класс BindingSource - > свойство DataSource


Сложная привязка данных


- ▶ Сложная привязка (complex data binding) – это привязка списочного источника данных целиком к элементу управления


Элемент управления DataGridView – отображает строки и столбцы данных в сетке, которые пользователь может изменить.

Элемент управления DataSet – представляет кэш данных в памяти

Элемент управления BindingSource – встраивает источник данных на форму и обеспечивает возможность навигации, фильтрации, сортировки и обновления.

 DataGridView


 DataSet

 BindingSource

 dataSet1

Список ошибок [Вывести](#)

 bindingSource1

 Наше первое приложение

DataGridView Задачи

Выберите источник данных:

[Правка столбцов...](#)

[Добавить столбец...](#)

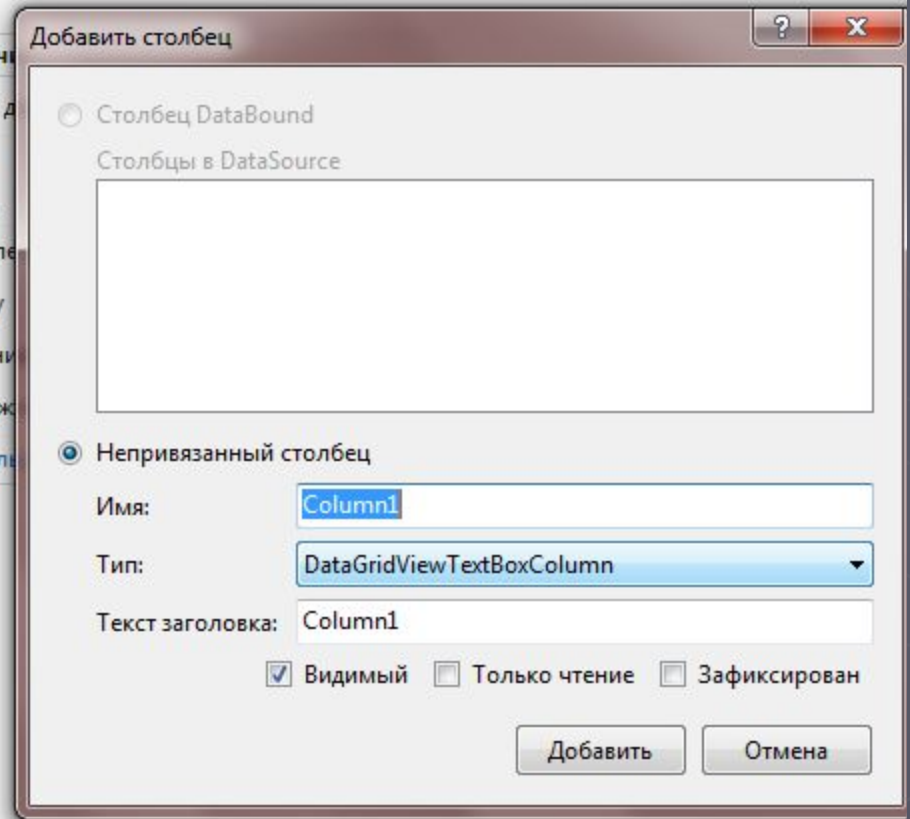
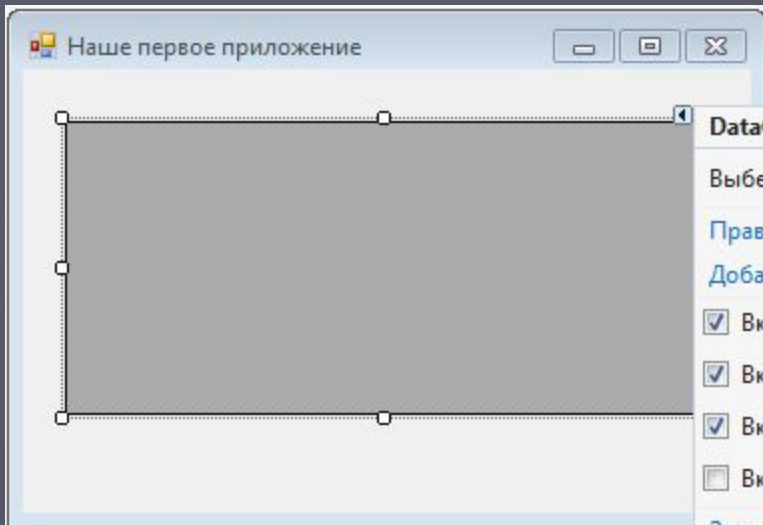
Включить добавление

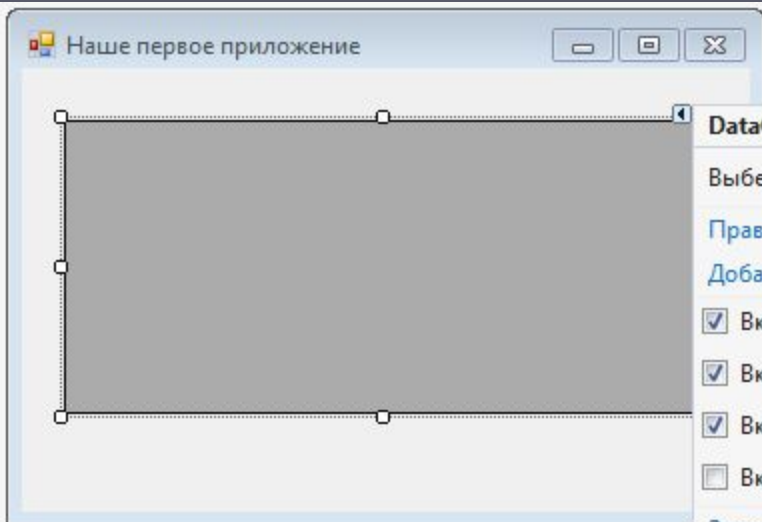
Включить правку

Включить удаление

Включить возможность изменения порядка столбцов

[Закрепить в родительском контейнере](#)





- DataGridView Задачи
- Выберите источник данных
- Правка столбцов...
- Добавить столбец...
- Включить добавление
- Включить правку
- Включить удаление
- Включить возможность
- Закрепить в родителе

Добавить столбец

Столбец DataBound

Столбцы в DataSource

Непривязанный столбец

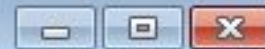
Имя:

Тип:

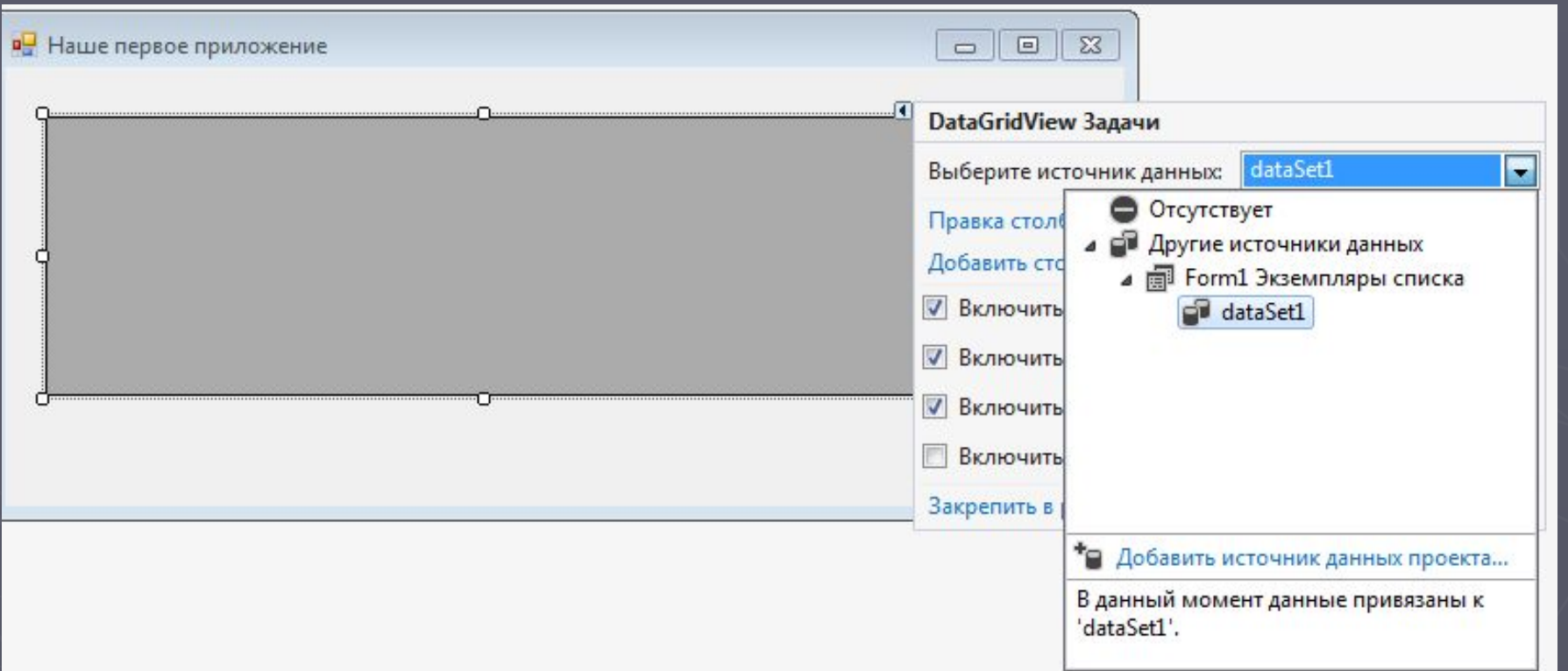
Текст заголовка:

- DataGridViewButtonColumn
- DataGridViewCheckBoxColumn
- DataGridViewComboBoxColumn
- DataGridViewImageColumn
- DataGridViewLinkColumn
- DataGridViewTextBoxColumn

Наше первое приложение



| | Column1 | Column2 | Column3 | Column4 |
|---|---------|--------------------------|--------------------------|----------------------|
| * | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="text"/> |



Создаем класс компилируем

```
namespace STUD1
{
    public class Student
    {
        private int id;

        public int Id
        {
            get { return id; }
            set { id = value; }
        }
        private string surname;

        public string Surname
        {
            get { return surname; }
            set { surname = value; }
        }
        private string gender = "M";

        public string Gender
        {
            get { return gender; }
            set { gender = value; }
        }
        private DateTime birthDate;
    }
}
```


Свойства

bindingSource1 System.Windows.Forms.BindingSource

⊕ (ApplicationSettings)

| | |
|-------------------|-----------------------|
| (Name) | bindingSource1 |
| AllowNew | True |
| DataMember | |
| DataSource | STUD1.Student |
| Filter | |
| GenerateMember | True |
| Modifiers | Private |
| Sort | |

- Отсутствует
- Другие источники данных
- Источники данных проекта
 - Student**

 [Добавить источник данных проекта...](#)

Выбор источника данных проекта создает экземпляр на форме и привязывает к этому экземпляру через новый источник BindingSo...

Student.cs

Form1.Designer.cs

Form1.cs

Form1.cs [Конструктор]

Отчет о преобразовании

| | Id | Surname | Gender | BirthDate | Course | Group | Scholarship |
|---|----|---------|--------|-----------|--------|-------|--------------------------|
| * | | | | | | | <input type="checkbox"/> |

bindingSource1

- ▶ Каждая колонка, получаемая из источника данных, вызовет добавление соответствующей колонки в control-e.
- ▶ Названия колонок источника отобразятся в заголовках колонок.
- ▶ Если пользователь щелкнет по заголовку колонки, строки будут автоматически отсортированы.

Текущая (выбранная) ячейка

Заголовки столбцов (ColumnHeaders)

`.RowIndex`

Заголовки строк (RowHeaders)

`.ColumnIndex`

| | ProductName | QuantityPerUnit | UnitPrice | UnitsInStock | UnitsOnOrder |
|---|------------------------------|---------------------|-----------|--------------|--------------|
| 0 | Chai | 10 boxes x 20 bags | 18.0000 | 39 | 0 |
| | Chang | 24 - 12 oz bottles | 19.0000 | 17 | 40 |
| | Aniseed Syrup | 12 - 550 ml bottles | 10.0000 | 13 | 70 |
| | Chef Anton's Cajun Seasoning | 48 - 6 oz jars | 22.0000 | 53 | 0 |
| | Chef Anton's Gumbo Mix | 36 boxes | 21.3500 | 0 | 0 |
| 5 | Grandma's Boysenberry Spread | 12 - 8 oz jars | 25.0000 | 120 | 0 |

Подключение к базе данных Microsoft Access

► Допустим у нас имеется некоторая база данных, созданная в СУБД Microsoft Access. Файл базы данных имеет имя «db1.mdb».

► Путь к файлу базы данных:

E:\Programs\C_Sharp\WindowsFormsApplication1\db1.mdb

База данных имеет одну таблицу с именем «**Tovar**».

Порядок выполнения

- ▶ Запустить MS Visual Studio. Создать приложение Windows Forms Application.
- ▶ Вызов мастера подключения.
Для доступа к файлу базы данных необходимо подключить его к приложению. Это осуществляется путем Вид-Другие окна-Источники данных-Добавить новый источник данных.

| | | |
|----|----------------------------------|----------------|
| <> | Код | Ctrl+Alt+0 |
| 🏗 | Конструктор | Shift+F7 |
| 🔍 | Обозреватель решений | Ctrl+Alt+L |
| 👥 | Командный обозреватель | Ctrl+\, Ctrl+M |
| 🏗 | Обозреватель архитектуры | Ctrl+\, Ctrl+R |
| 🏗 | Обозреватель объектов SQL Server | Ctrl+\, Ctrl+S |
| 📁 | Окно закладок | Ctrl+K, Ctrl+W |
| 📊 | Иерархия вызовов | Ctrl+Alt+K |
| 👤 | Классы | Ctrl+Shift+C |
| 📄 | Окно определения кода | Ctrl+Shift+V |
| 🔍 | Обозреватель объектов | Ctrl+Alt+J |
| 🚫 | Список ошибок | Ctrl+\, E |
| 🖨 | Вывод | Alt+2 |
| 🌐 | Ресурсы | Ctrl+Shift+E |
| 🏠 | Начальная страница | |
| 🧰 | Панель элементов | Ctrl+Alt+X |
| | Результаты поиска | |
| | Другие окна | |
| | Панели инструментов | |
| 🖥 | Во весь экран | Shift+Alt+BВВД |
| 📄 | Все окна | Shift+Alt+M |

| | | |
|---|--|----------------|
| 📄 | Окно команд | Ctrl+Alt+A |
| 📄 | Источники данных | Shift+Alt+D |
| 🌐 | Веб-браузер | Ctrl+Alt+R |
| 🔍 | Инспектор страниц | |
| 🚀 | Загрузить тестовые запуски | |
| 🏗 | Обозреватель слоев | |
| 📄 | Обозреватель управления исходным кодом | |
| 🏗 | Обозреватель моделей UML | Ctrl+\, Ctrl+U |
| 🔍 | Инструментальные средства для обработки данных | |
| 🖨 | Консоль диспетчера пакетов | |
| 📄 | Структура документа | Ctrl+Alt+D |
| 🕒 | Журнал | |
| 🕒 | Ожидаящие изменения | |
| 🔧 | Окно свойств | Alt+BВВД |
| 🏗 | Обозреватель серверов | Ctrl+Alt+S |
| 📄 | Список задач | Ctrl+\, T |
| 📄 | F# Interactive | Ctrl+Alt+F |
| 🔍 | Анализ кода | |
| 📄 | Результаты метрик кода | |

Обозреватель серверов Панель элементов

Form1.cs [Конст

Наше пе

Добавить новый источник данных

С проектом в настоящий момент не связаны источники данных. Добавьте новый источник данных, а затем создайте привязки к данным, перетащив элементы из окна на формы или существующие элементы управления.

[Добавить новый источник данных...](#)



dataSet1 bin

► **Выбор типа источника данных.**

В результате откроется окно мастера для подключения к источнику данных которое изображено на рис. 2.

В окне необходимо выбрать один из четырех возможных вариантов подключения к источнику данных. В MSVS существует четыре типа подключения к источникам данных:

- Database – подключение к базе данных и выбор объектов базы данных;
- Service – открывает диалоговое окно Add ServiceReference позволяющее создать соединение с сервисом, который возвращает данные для вашей программы;
- Object – позволяет выбрать объекты нашего приложения, которые в дальнейшем могут быть использованы для создания элементов управления (controls) с привязкой к данным;
- Share Point – позволяет подключиться к сайту SharePoint и выбрать объекты для вашей программы.

В нашем случае выбираем элемент Database и продолжаем нажатием на кнопке Next.

Data Source Configuration Wizard



Choose a Data Source Type

Where will the application get data from?



Lets you connect to a database and choose the database objects for your application.

< Previous

Next >

Finish

Cancel

Рисунок. 2. Выбор типа подключения из которого приложение будет получать данные

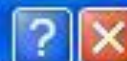
➤ **Выбор модели подключения к базе данных.**

Следующий шаг – выбор модели подключения к базе данных (рис. 3).

Система предлагает выбор одного из двух вариантов:

- – модели данных на основе набора данных (**Dataset**);
- – модели данных Entity, что означает, что система может сгенерировать модель данных из базы данных которой могут выступать сервера баз данных Microsoft SQL Server, Microsoftt SQL Server Compact 3.5 или Microsoft SQL Server Database File, либо создать пустую модель как отправную точку для визуального проектирования концептуальной модели с помощью панели инструментов.

В нашем случае выбираем тип модели данных **DataSet**.



Choose a Database Model

What type of database model do you want to use?



Dataset



Entity Data
Model

The database model you choose determines the types of data objects your application code uses. A dataset file will be added to your project.

< Previous

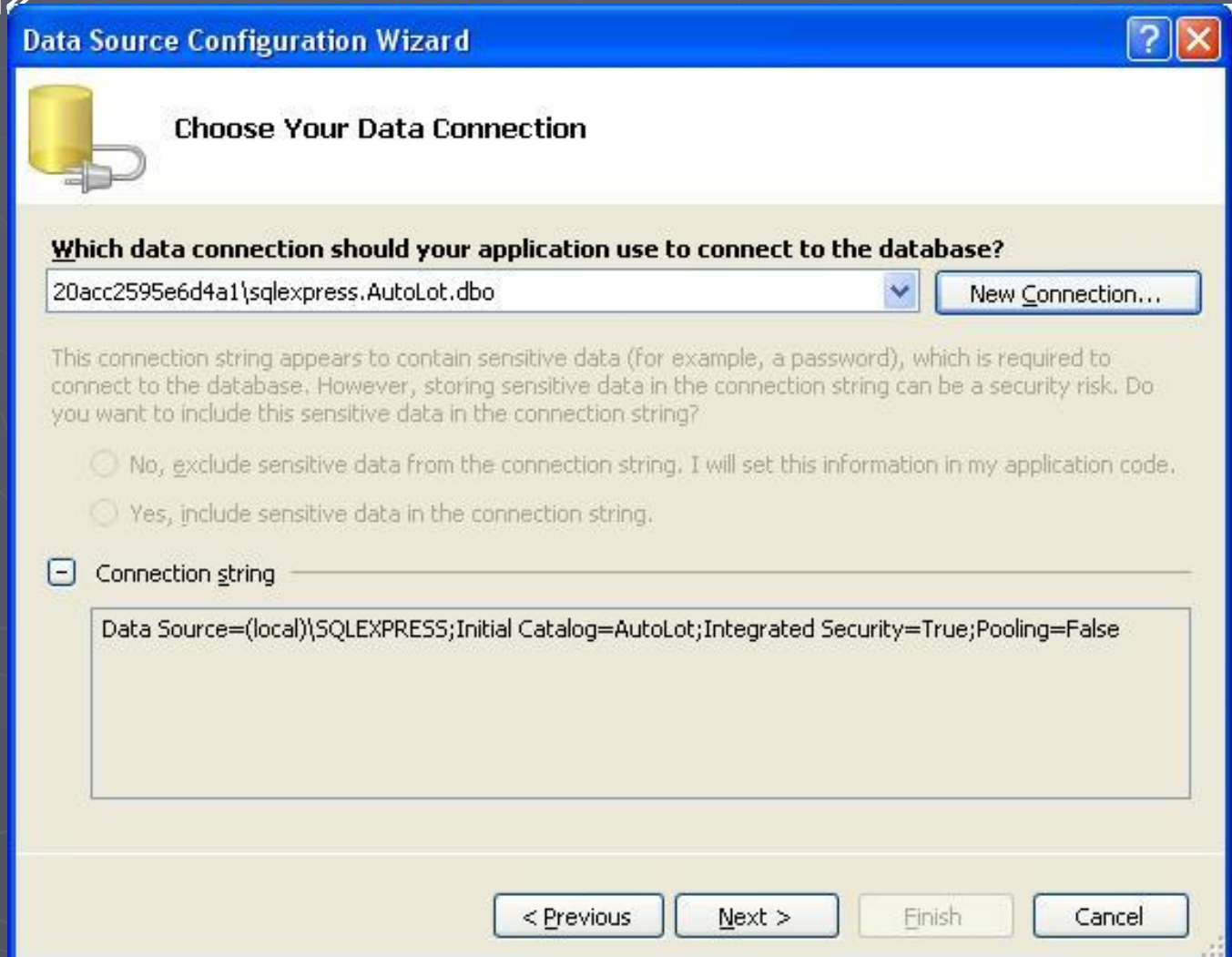
Next >

Finish

Cancel

Задание соединения с БД.

Следующим шагом мастера (рис. 4) есть выбор соединения данных которое должно использоваться приложением для соединения с базой данных. Для создания нового соединения необходимо выбрать кнопку «New Connection»



В результате откроется окно «**Add Connection**» (рис. 5) в котором нужно добавить новое соединение Microsoft Access и выбрать маршрут к файлу базы данных.

В нашем случае поле «**Data source**» уже содержит нужный нам тип соединения «**Microsoft Access Database File (OLE DB)**».



Если нужно выбрать другую базу данных, то для этого используется кнопка «Change...», которая открывает окно, изображенное на рисунке 6.



В окне на рисунке 6 системой Microsoft Visual Studio будет предложено следующие виды источников данных:

Microsoft Access Database File – база данных Microsoft Access;

Microsoft ODBC Data Source – доступ к базе данных с помощью программного интерфейса ODBC (Open Database Connectivity);

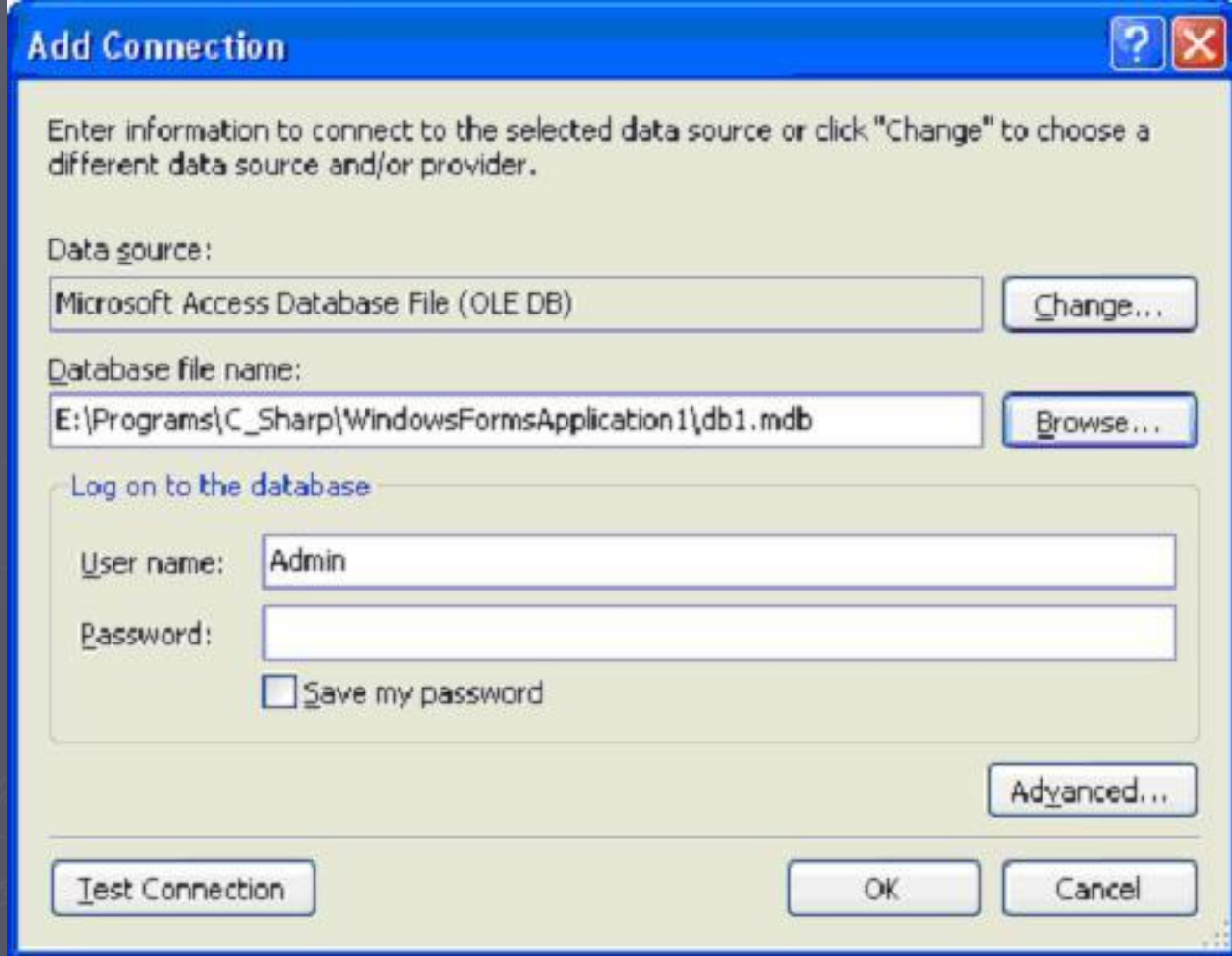
Microsoft SQL Server;

Microsoft SQL Server Compact 3.5;

Microsoft SQL Server Database File;

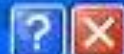
Oracle Database – база данных Oracle.

- ▶ Нажимаем кнопку «Browse...» и в открывшемся окне (рис. 7) «Add Connection» выбираем маршрут к файлу базы данных «db1.mdb». Целесообразно размещать файл базы данных в каталоге содержащем исполняемый модуль приложения.
- ▶ Для проверки правильности установленного соединения можно воспользоваться кнопкой «Test Connection».



- ▶ После нажатия на кнопке ОК система сгенерирует строку «**Connection string**» (рис. 8) который в дальнейшем будет использован для программного подключения к базе данных.
- ▶ Кликаем на «**Next**» для продолжения работы мастера.

Data Source Configuration Wizard



Choose Your Data Connection

Which data connection should your application use to connect to the database?

db1.mdb



New Connection...

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- No, exclude sensitive data from the connection string. I will set this information in my application code.
- Yes, include sensitive data in the connection string.

[-] Connection string

```
Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=E:\Programs\C_Sharp\WindowsFormsApplication1\db1.mdb
```

< Previous

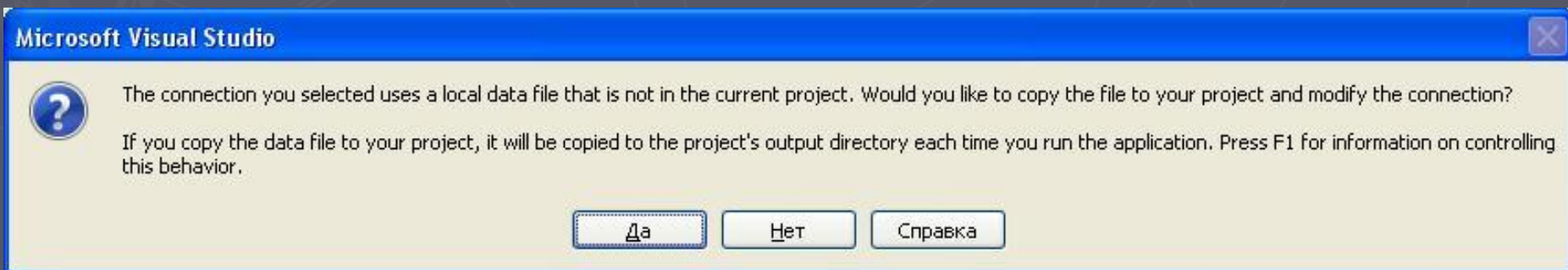
Next >

Finish

Cancel

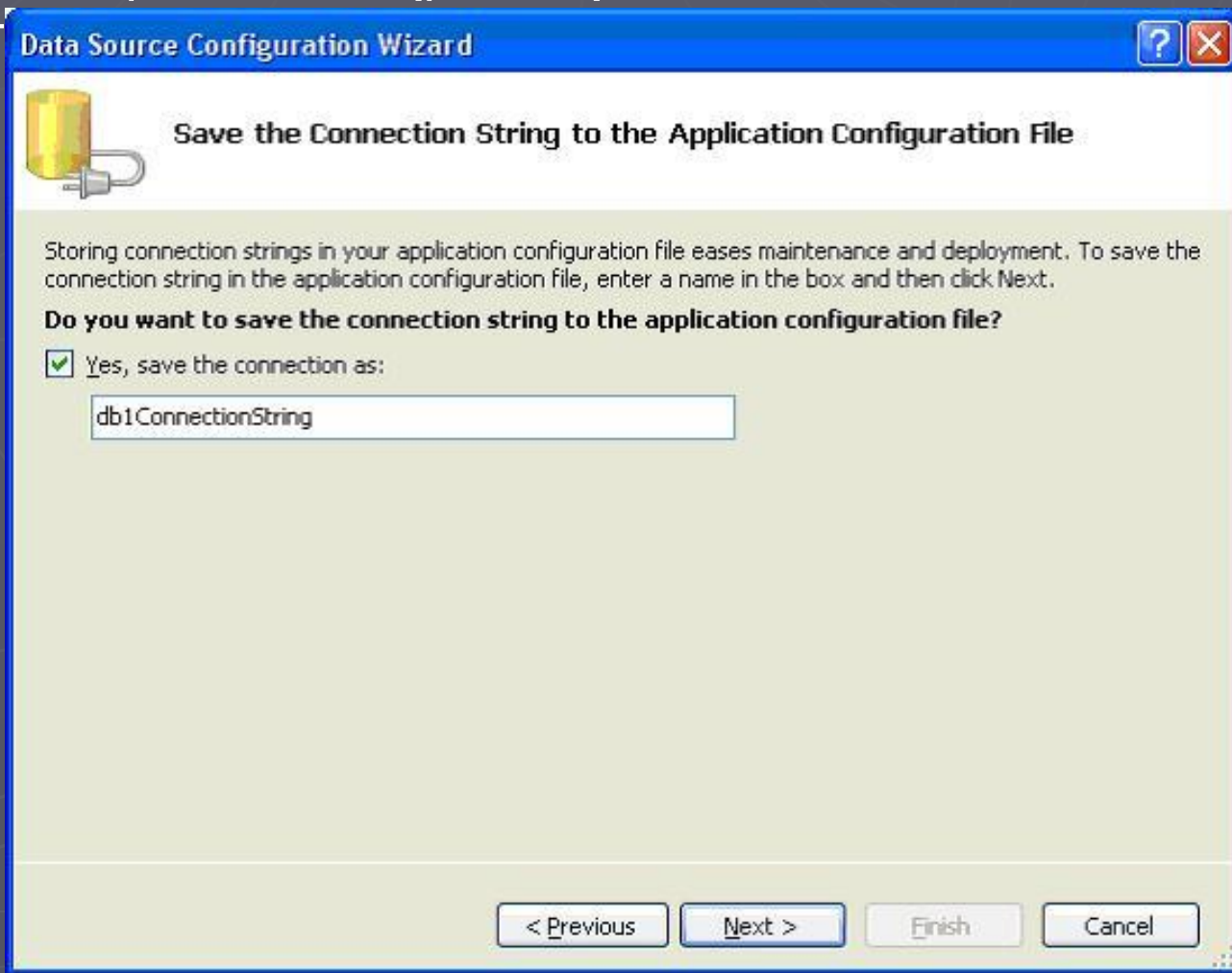
Рис. 8. Строка Connection string

► После выбора **Next** система выдаст информационное окно следующего вида (рис. 9). Если выбрать «**Да**», то файл базы данных «db1.mdb» будет копироваться в выходной каталог приложения каждый раз при его запуске в среде MS Visual Studio. Как правило, это каталог, содержащий основные модули приложения (например Program.cs , Form1.cs и другие).



► Формирование конфигурационного файла приложения.

После выбора кнопки «Next» мастера откроется следующее окно, в котором предлагается сохранить строку соединения в конфигурационный файл приложения (рис. 10). Ничего не изменяем, оставляем все как есть.

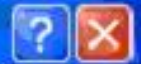


► Выбор объектов базы данных для использования в программе.

Последнее окно мастера (рисунок 11) предлагает выбрать список объектов (таблиц, запросов, макросов, форм и т.д.), которые будут использоваться в наборе данных. Как правило выбираем все таблицы базы данных. В нашем примере база данных содержит всего одну таблицу с именем **Tovar**.

После выбора кнопки «**Finish**» заканчиваем работу с мастером подключения. Теперь база данных подключена к приложению и будет автоматически подключаться при его запуске или при его проектировании в MS Visual Studio.

Data Source Configuration Wizard



Choose Your Database Objects

Which database objects do you want in your dataset?

- Tables
 - Tovar
 - ID_Tovar
 - Name
 - Price
 - Count
 - Views

DataSet name:

db1DataSet

< Previous

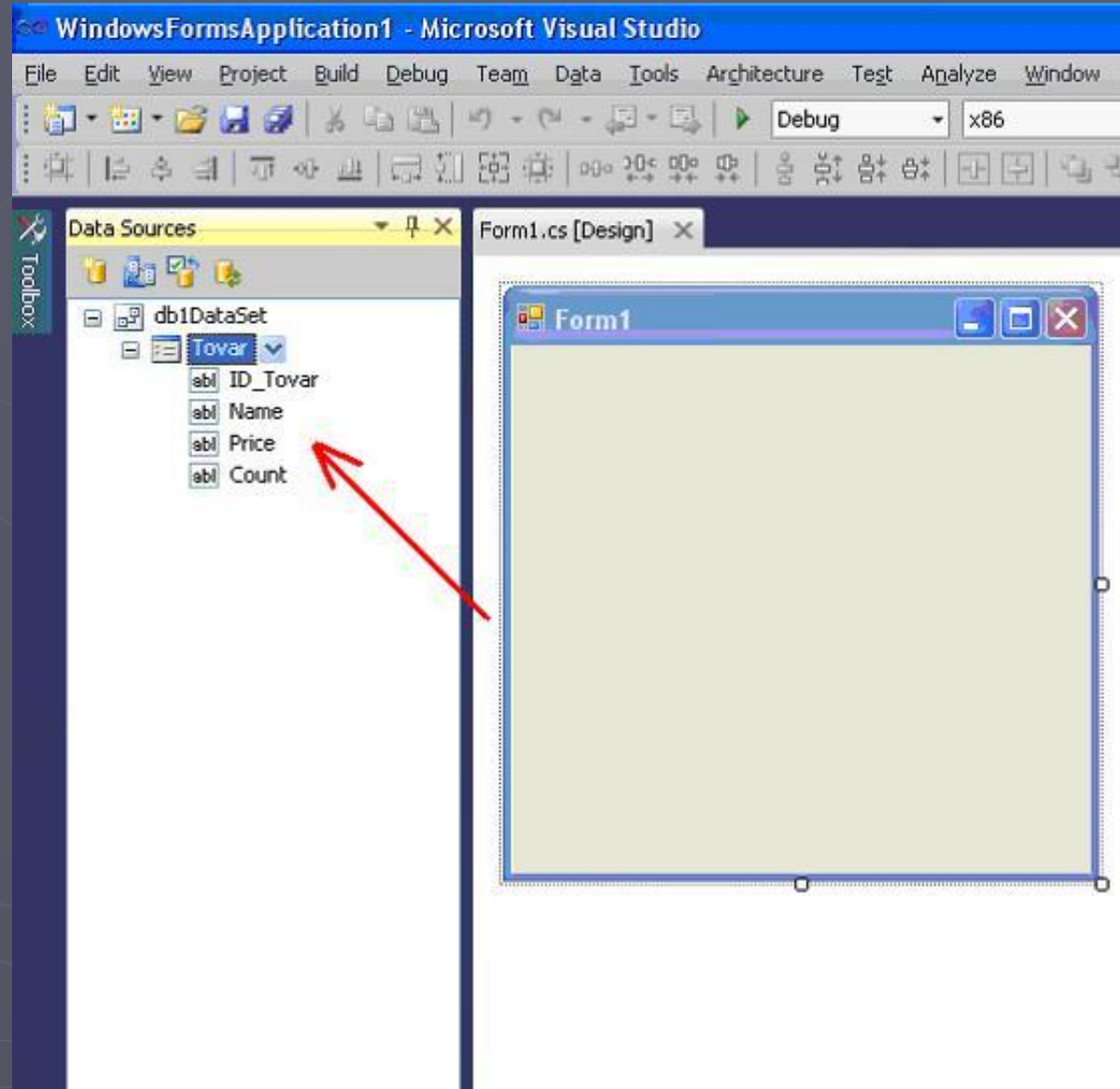
Next >

Finish

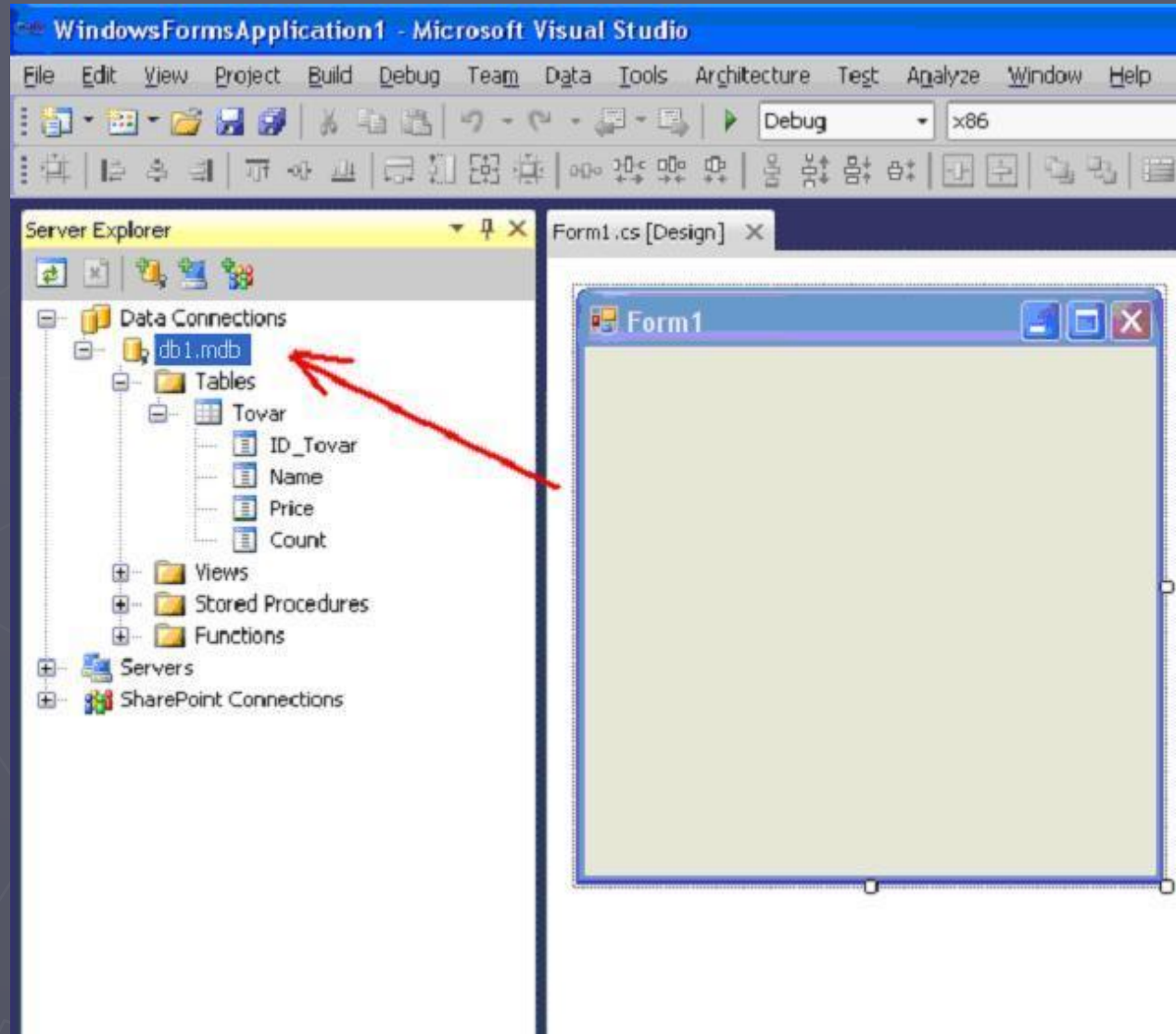
Cancel

Что же изменилось в программе после выполнения мастера?

Если выбрать панель **Data Source** (рисунок 12), то можно увидеть, как подключен набор данных с именем **db1DataSet** в котором есть таблица с именем **Tovar**.



Точно также можно увидеть изменения в панели **Server Explorer** (рисунок 13), где появилась база данных «**db1.mdb**» с таблицей **Tovar** и ее полями. Приложение может подключать не только одну, но и несколько баз данных.



► Подключение методов оперирования базой данных.

Для того, чтобы использовать методы, которые будут работать с базой данных **MS Access**, необходимо подключить пространство имен **System.Data.OleDb**.

Для этого в коде основной формы (Form1.cs) добавляем следующую строку:

```
using System.Data.OleDb;
```

На этом этапе подключение к базе данных **db1.mdb** выполнено. Дальнейшими шагами есть создание программного кода для оперирования данными в базе данных.

Вывод таблицы базы данных Microsoft Access в компоненте dataGridView

- ▶ Пусть имеется база данных, созданная в приложении Microsoft Access. Имя файла базы данных "mydb.mdb". Файл размещается на диске по следующему пути:
- ▶ Путь к файлу базы данных:
`C:\Programs\C_Sharp\WindowsFormsApplication1\mydb.mdb`
- ▶ База данных имеет несколько таблиц, одна из которых имеет название "Order".
- ▶ Задача состоит в том, чтобы с помощью средств языка C# осуществить подключение к базе данных и вывести таблицу с именем «Order» на форму.

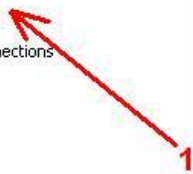
► Чтение строки подключения **Connection String**.

После подключения базы данных MS Access к нашему приложению, мы получаем строку подключения **Connection String**, которую будем использовать в нашем приложении.

Чтобы получить корректную строку подключения к базе данных, нужно выделить базу данных в панели **Server Explorer (mydb.mdb)** и в окне **“Properties”** скопировать значение свойства **“Connection String”** (рис. 2, красное выделение). Следует учесть, что **‘\’** в строке на C# нужно заменить на **‘\\’** согласно синтаксису языка.

Server Explorer

- Data Connections
 - mydb.mdb
- Servers
- SharePoint Connections



```
Form1.cs x Form1.cs [Design]  
WindowsFormsApplication1.Form1 Form1()  
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
  
namespace WindowsFormsApplication1  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
    }  
}
```

Error List

0 Errors 0 Warnings 0 Messages

| Description | File | Line | Column | Project |
|-------------|------|------|--------|---------|
|-------------|------|------|--------|---------|

Solution Explorer

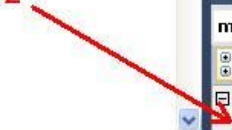
- Solution 'WindowsFormsApplication1' (1 project)
 - WindowsFormsApplication1
 - Properties
 - References
 - app.config
 - Form1.cs
 - mydb.mdb
 - mydbDataSet.xsd
 - mydbDataSet.Designer.cs
 - mydbDataSet.xsc
 - mydbDataSet.xss
 - Program.cs

Properties

mydb.mdb Connection

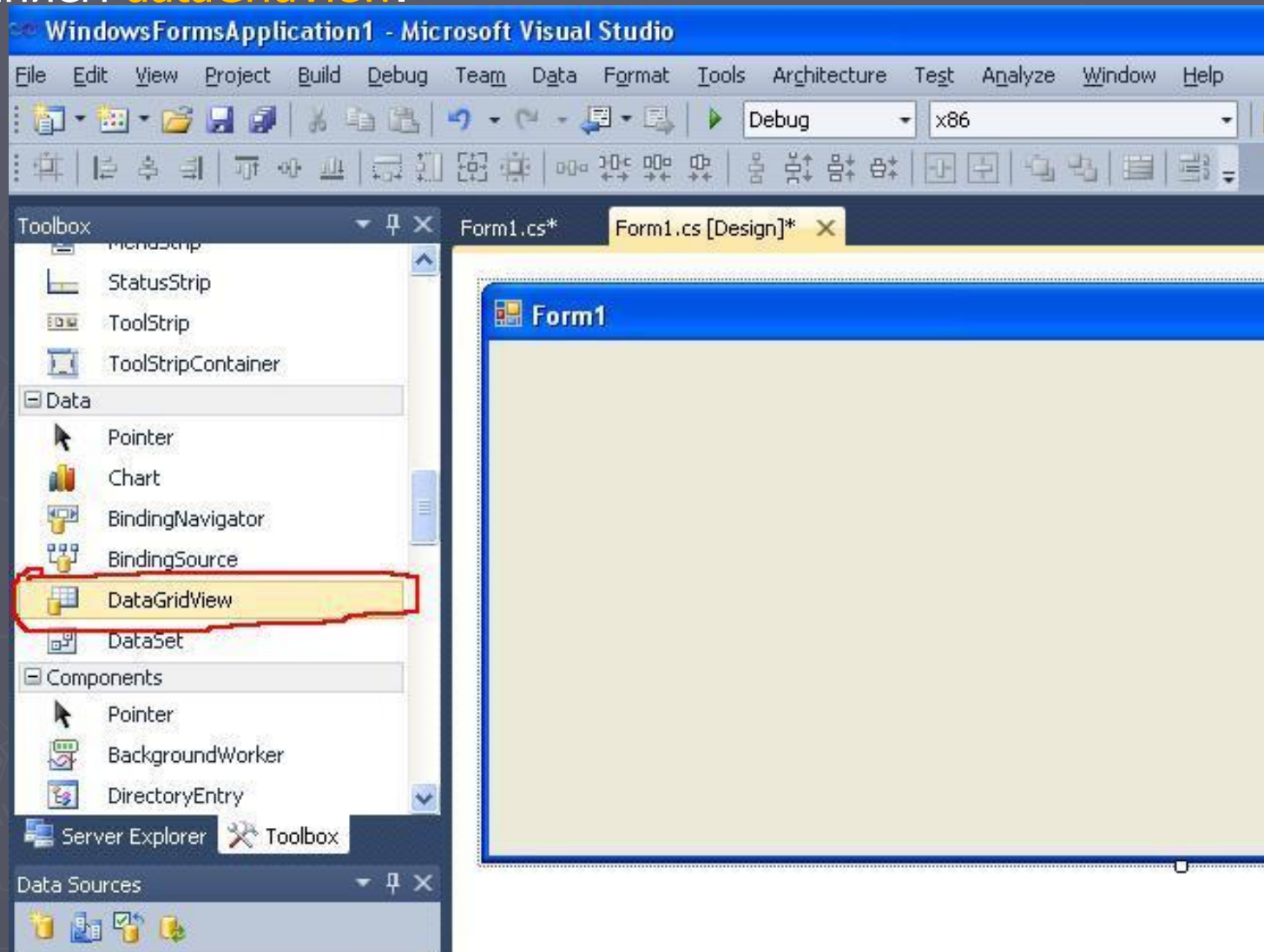
| Connection | |
|-------------------|--|
| Connection String | Provider=Microsoft.Jet.OleDB.4.0;Data Source=.;User ID=;Password=; |
| Provider | .NET Framework Data Provider for Microsoft Jet |
| State | Closed |

Connection String



► Размещение компонента типа `dataGridView`.

Выносим на форму компонент `dataGridView` (рис. 3), представляющий компонент-таблицу, в которой будет выведена наша таблица "Order" из базы данных. Получаем объект-переменную под названием `dataGridView`.



Размещение компонента dataGridView на форме изображено на рисунке 4.



- **Добавление переменных SQL-запроса и строки подключения к базе данных.**
- В программный код Form1.cs формы вводим дополнительные переменные **CmdText** и **ConnectionString**.
- Переменная **CmdText** будет содержать строку SQL-запроса для вывода всех записей таблицы "**Order**".
- Переменная **ConnectionString** представляет собой строку подключения к базе данных (см. п. 2). Общий вид программного кода класса формы следующий:

```
public partial class Form1 : Form
{
    public string CmdText = "SELECT * FROM [Order]";
    public string ConnString = "Provider=Microsoft.Jet.OLEDB.4.0;
DataSource=C:\\Programs\\C_Sharp\\WindowsFormsApplication1\\mydb.mdb";
    public Form1()
    {
        InitializeComponent();
    }
}
```

► Подключение пространства имен OleDb.

► В **Microsoft Visual Studio** взаимодействие с файлом данных **Microsoft Access** осуществляется с помощью поставщика данных **OLE DB** или **ODBC**. Поставщик данных **OLE DB** обеспечивает доступ к данным, находящимся в любом хранилище данных, если оно поддерживает классический протокол OLE DB на основе технологии COM. Этот поставщик состоит из типов, которые определены в пространстве имен **System.Data.OleDb**.

► В последующих шагах мы будем использовать методы из этого пространства имен. Поэтому, вначале файла **Form1.cs** добавим:

```
using System.Data.OleDb;
```

➤ Создание объекта типа **OleDbDataAdapter**.

➤ В конструкторе формы после вызова.

```
InitializeComponent();
```

➤ Добавляем строку создания объекта типа **OleDbDataAdapter**:

```
OleDbDataAdapter dA = new OleDbDataAdapter(CmdText, ConnString);
```

➤ Объект типа **OleDbDataAdapter** организует пересылку наборов данных с вызываемым процессом. Адаптеры данных содержат набор из четырех внутр. объектов команд: чтения, вставки, изменения и удаления информации.

➤ конструктор содержит параметрами строку запроса на языке SQL (переменная **CmdText**) и строку подключения к базе данных (переменная **ConnString**). Таким образом, после выполнения данного кода, объект адаптера уже связан с нашей базой данных.

➤ Создание объекта набора данных DataSet.

- После создания адаптера данных (**OleDbDataAdapter**) создаем объект типа **DataSet** (набор данных):

```
DataSet ds = new DataSet();
```

- Объект **DataSet** представляет собой буфер для хранения данных из базы. Этот буфер предназначен для хранения структурированной информации, представленной в виде таблиц, поэтому вложенным объектом DataSet является **DataTable**. Внутри одного объекта DataSet может храниться **несколько загруженных таблиц** из базы данных, помещенных в соответствующие объекты DataTable.

➤ Всякая таблица состоит из **столбцов** и **строк**. Для работы с ними предназначены специальные объекты - **DataColumn** и **DataRow**. Между таблицами могут быть связи - здесь они представлены объектом **DataRelation**. Наконец, в таблицах есть первичные и вторичные ключи - объект **Constraint** со своими двумя подклассами **UniqueConstraint** и **ForeignKeyConstraint** описывают их.

```
➤ DataSet dsTours = new DataSet();
➤ DataTable dtTours = dsTours.Tables.Add("Туры");
➤ DataColumn dcPrice = dtTours.Columns.Add("Цена",
typeof(Decimal));
➤ DataColumn dcPriceNDS = dtTours.Columns.Add("Цена с НДС",
typeof(Decimal));
dcPriceNDS.Expression = "Цена*0.15+Цена";
➤ DataRow myRow = dtTours.NewRow();
➤ myRow["Цена"] = 25000;
➤ dtTours.Rows.Add(myRow);
```

► Заполнение таблицы "Order" на основе SQL-запроса.

► Следующая команда – это заполнение набора данных (переменная **ds**) значениями записей из базы данных на основе **SQL-запроса**, содержащегося в адаптере данных **dA** с помощью метода **Fill()**:

```
dA.Fill(ds, "[Order]");
```

Визуализация данных в dataGridView1.

На данный момент данные из таблицы "Order" считаны в объекте `ds` (типа `DataSet`), представляющем собой набор данных.

Для их отображения необходимо чтобы свойство `DataSource` компонента `dataGridView1` ссылалось на первую таблицу (в нашем случае одна таблица) набора данных `ds`. Программный код этой операции имеет следующую реализацию:

```
*****.Tables ["Order"]  
*****.Tables [ 0 ]
```

```
dataGridView1.DataSource = ds.Tables[0].DefaultView;
```

После этого данные из таблицы "Order" отобразятся на форме (рис. 5).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public string CmdText = "SELECT * FROM [Order]";
        public string ConnString =
"Provider=Microsoft.Jet.OLEDB.4.0;DataSource=C:\\Programs\\C_Sharp\\WindowsFormsApplic
ation1\\mydb.mdb";
        public Form1()
        {
            InitializeComponent();
            OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CmdText, ConnString);
            // создаем объект DataSet
            DataSet ds = new DataSet();
            // заполняем таблицу Order
            // данными из базы данных
            dataAdapter.Fill(ds, "[Order]");
            dataGridView1.DataSource = ds.Tables[0].DefaultView;
        }
    }
}
```

Form1

| | IdOrder | NameOfSender | NameOfRecipier | DateOfSending | DateOfReceipt | IdTariff |
|---|---------|--------------|----------------|---------------|---------------|----------|
| ▶ | 2 | Jonson J.K. | Kempbell L.K. | 15.05.2015 | 18.09.2015 | 1 |
| | 3 | Kurt D.S. | Rudolf S.V. | 17.05.2015 | 19.05.2015 | 2 |
| * | | | | | | |

Общая схема взаимодействия между объектами.



Таким образом, можно выводить на форму любую таблицу базы данных. Условия выведения данных из базы данных задаются в строке SQL-запроса в переменной CmdText.





| | Код тура | Название | Цена | Информация |
|---|----------|----------|-------|---------------|
| ▶ | 1 | Кипр | 25000 | В стоимость |
| | 2 | Греция | 32000 | В августе и с |
| | 3 | Таиланд | 30000 | Не включая |
| | 4 | Италия | 26000 | Завтрак в от |
| | 5 | Франция | 27000 | Дополнител |
| * | | | | |

Программное изменение определенной строки и программное удаление определенной строки

```
DataSet dsTours = new DataSet();
DataTable dtTours = dsTours.Tables.Add("Туры");

dataGridView1.DataSource = dsTours.Tables["Туры"].DefaultView;

DataRow myRow = dtTours.Rows[4];
myRow.BeginEdit();
myRow["Код тура"] = 5;
myRow["Название"] = "Турция";
myRow["Цена"] = "27000";
myRow["Информация"] = "Осенние скидки с 15 октября";
myRow.EndEdit();

DataRow myRow2 = dtTours.Rows[0];
dtTours.Rows.Remove(myRow2);
myRow2.Delete();
```

Вывод двух связанных таблиц данных в один элемент DataGridView

В базе данных есть таблица Туристы , которая связана с другими таблицами. Было бы удобно выводить эту таблицу в элемент DataGridView вместе с другими таблицами, а также выводить связанные записи этой таблицы.

```
14 public partial class Form1 : Form
15 {
16     string commandText = "SELECT [Код туриста], Фамилия, Имя, Отчество FROM Туристы";
17
18     string connectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="+
19         @"D:\ВМИ\For ADO\BDTur_firm.accdb";
20
21     string commandText2 = "SELECT [Код туриста], [Серия паспорта], Город, Страна, Телефон, Индекс " +
22         "FROM [Информация о туристах]";
23
24     public Form1()
25     {
26         InitializeComponent();
27
28         //создаем соединение////////////////////////////////////
29         OleDbConnection conn = new OleDbConnection(connectionString);
30         OleDbCommand myCommand = new OleDbCommand();
31         myCommand.Connection = conn;
32         myCommand.CommandText = commandText;
33
34         //создаем объект DataAdapter и в его свойстве SelectCommand устанавливаем запрос
35         OleDbDataAdapter dataAdapter = new OleDbDataAdapter();
36         dataAdapter.SelectCommand = myCommand;
37
38         //открываем соединение
39         conn.Open();
40
41         //создаем объект
42         DataSet ds = new DataSet();
43         //В объекте DataSet здесь будут храниться две таблицы - главная и связанная с ней дочерняя.
44         //Поэтому воспользуемся свойством TableMappings для занесения в него первой таблицы «Туристы»
45         dataAdapter.TableMappings.Add("Table", "Туристы");
46         dataAdapter.Fill(ds);
```

```
47 //добавляем объекты OleDbDataAdapter и OleDbCommand для таблицы «Информация о туристах»
48 OleDbCommand myCommand2 = new OleDbCommand();
49 myCommand2.Connection = conn;
50 myCommand2.CommandText = commandText2;
51 OleDbDataAdapter dataAdapter2 = new OleDbDataAdapter();
52
53 //связываем OleDbDataAdapter2 со второй командой и отобразим «Информацию о туристах» на его таблицу
54 dataAdapter2.SelectCommand = myCommand2;
55 dataAdapter2.TableMappings.Add("Table", "Информация о туристах");
56
57 //заполняем объект DataSet данными из второй таблицы.
58 dataAdapter2.Fill(ds);
59 //Получаем объект DataSet с двумя таблицами.Теперь можно выводить 1 из этих таблиц на форму, или две сразу.
60
61 //Но связь между таблицами еще не создана.
62 //Для конфигурирования отношения по полю «Код туриста» создадим два объекта DataColumn:
63 DataColumn dcTouristsID = ds.Tables["Туристы"].Columns["Код туриста"];
64 DataColumn dcInfoTouristsID = ds.Tables["Информация о туристах"].Columns["Код туриста"];
65
66 //создаем DataRelation, передаем ему название отношения между таблицами и 2 объекта DataColumn
67 DataRelation dataRelation = new DataRelation("Дополнительная информация", dcTouristsID, dcInfoTouristsID);
68
69 //Добавляем созданный объект отношения к объекту DataSet:
70 ds.Relations.Add(dataRelation);
71
72 //Создаем объект DataViewManager, отвечающий за отображение DataSet в объекте DataGridView:
73 DataViewManager dsview = ds.DefaultViewManager;
74
75 //Присваиваем свойству DataSource объекта DataGridView созданный объект DataViewManager:
76 dataGridView1.DataSource = dsview;
77
78 //сообщаем объекту DataGridView, какую таблицу считать главной и отображать на форме:
79 dataGridView1.DataMember = "Туристы";
80
81 //Закрываем соединение:
82 conn.Close();
83 }
84 }
85 }
```


Запускаем приложение. У каждой строки таблицы появился знак «+», говорящий о том, что у данной записи имеются дочерние записи. Для перехода на дочернюю запись нажимаем на «+» и нажимаем на ссылку «Дополнительная информация». Окно приложения приобретает следующий вид. Для возвращения на родительскую запись нажимаем на кнопку со стрелкой.

| | Код туриста | Фамилия | Имя | Отчество |
|-----|-------------|-----------|--------------|--------------|
| ▶ ⊕ | 1 | Иванов | Василий | Степанович |
| ⊕ | 2 | Николаев | Олег | Валентинович |
| ⊕ | 3 | Андреева | Инна | Вячеславовна |
| ⊕ | 4 | Волков | Антон | Павлович |
| ⊕ | 5 | Кириллова | Ольга | Михайловна |
| ⊕ | 6 | test | (не определе | (не определе |
| * | | | | |

| Туристы: Код туриста: 2 Фамилия: Николаев Имя: Олег Отчество: Валентинович | | | | | | |
|--|-------------|-------------|--------|--------|---------|--------|
| | Код туриста | Серия паспо | Город | Страна | Телефон | Индекс |
| ▶ * | 2 | TE 1562487 | Ростов | Россия | 3216547 | 120035 |
| * | | | | | | |

Вывод связанных таблиц данных в два элемента DataGridView

- ▶ Наиболее часто встречаемая задача при разработке приложений, связанных с базами данных, - это одновременный вывод двух таблиц на форму, причем при перемещении по записям главной таблицы в дочерней автоматически отображаются связанные записи.
- ▶ Добавим на форму еще один компонент DataGridView, в котором будут отображаться связанные записи.
- ▶ добавим следующий код:

```
dataGrid2.DataSource = dsview;  
dataGrid2.DataMember = "Туристы.Дополнительная информация";
```

The screenshot shows a Windows application window titled "Form1" with two DataGridView controls. The top DataGridView displays a list of tourists with columns for ID, Surname, Name, and Patronymic. The bottom DataGridView displays detailed information for the selected tourist (ID 3), including passport series, city, country, phone, and index.

| Код туриста | Фамилия | Имя | Отчество |
|-------------|-----------|--------------|--------------|
| 1 | Иванов | Василий | Степанович |
| 2 | Николаев | Олег | Валентинови |
| 3 | Андреева | Инна | Вячеславовн |
| 4 | Волков | Антон | Павлович |
| 5 | Кириллова | Ольга | Михайловна |
| 6 | test | (не определе | (не определе |

| Код туриста | Серия паспорта | Город | Страна | Телефон | Индекс |
|-------------|----------------|----------|--------|---------|--------|
| 3 | ИП 6548243 | Оренбург | Россия | 685472 | 870054 |

Выведем теперь все записи клиентов, имена которых начинаются на "О":

```
string CommandText = "SELECT Фамилия, Имя, Отчество  
FROM Туристы where Имя like 'O%'";
```

Запросы примеры



| | Фамилия | Имя | Отчество |
|---|-----------|-------|--------------|
| ▶ | Николзев | Олег | Валентинович |
| | Кириллова | Ольга | Михаиловна |
| * | | | |

Изменить фамилию туриста с номером 3

- ▶ **CmdText** = "UPDATE Туристы " + "SET Фамилия = 'Сергеева' WHERE [Код туриста] = 3";

Вставить в таблицу Туристы новую строку

- ▶ **CmdText** = "INSERT " +
"INTO Туристы ([Код туриста], Фамилия,
Имя, Отчество) " +
"VALUES (6, 'Тихомиров', 'Андрей',
'Борисович')";

Удалить туриста под номером 4

- ▶ **CmdText** = "DELETE FROM Туристы WHERE [Код туриста] = 4";
-

Очистить всю таблицу Туристы

- ▶ **CmdText** = "DELETE FROM Туристы";
-

Подсчитать количество строк в таблице Туры

- ▶ **CmdText** = "SELECT COUNT (*) FROM Туры";
-

Найти максимальную цену тура

- ▶ **CmdText** = "SELECT MAX (Цена) FROM Туры";
-

Найти минимальную цену тура

- ▶ **CmdText** = "SELECT MIN (Цена) FROM Туры";
-

Найти среднее значение цен

- ▶ **CmdText** = "SELECT AVG (Цена) FROM Туры";

Ручное подключение к базе данных

```
private void btnCommand2_Click(object sender, EventArgs e)
{
    OleDbConnectionStringBuilder builder = new OleDbConnectionStringBuilder();
    builder.Provider = "Microsoft.ACE.OLEDB.12.0";
    builder.DataSource = @"Contacts.accdb";
    OleDbConnection con = new OleDbConnection(builder.ConnectionString); //задаем строку подключения
    con.Open(); //открываем соединение с базой данных
    OleDbCommand cmd = con.CreateCommand(); //создаем пустую команду
    cmd.CommandText = "SELECT MAX(Name) FROM Contacts"; //запускаем команду
    string result = cmd.ExecuteScalar().ToString(); //выполняет запрос и возвращает результат 1 ячейку
    //записываем ячейку в переменную
    //отключаемся от базы данных
    con.Close();
}
```

Параметризированный запрос

The screenshot shows a Windows application window titled "Form1" with three distinct sections for database operations:

- Пример UPDATE:** Contains two text input fields labeled "Введите код туриста" and "Введите фамилию туриста", followed by a blue button labeled "Обновить".
- Пример DELETE:** Contains one text input field labeled "Введите код туриста для удаления" and a blue button labeled "Удалить".
- Пример INSERT:** Contains four text input fields labeled "Введите код туриста", "Введите фамилию туриста", "Введите имя туриста", and "Введите отчество туриста", followed by a blue button labeled "Добавить".

Изменение Update

```
34 private void btnCommand2_Click(object sender, EventArgs e)
35 {
36     OleDbConnectionStringBuilder builder = new OleDbConnectionStringBuilder();
37     builder.Provider = "Microsoft.ACE.OLEDB.12.0";
38     builder.DataSource = @"Contacts.accdb";
39     OleDbConnection conn = new OleDbConnection(builder.ConnectionString);
40     conn.Open();//открываем соединение
41     try
42     {
43         string Family = Convert.ToString(this.TextBox1.Text); //в форме вользователь вводит фамилию в TextBox1
44         int TouristID = int.Parse(this.TextBox2.Text);//в форме вользователь вводит номер туриста в TextBox2
45
46
47         OleDbCommand myCommand = conn.CreateCommand(); //создаем оболочку для команды
48
49         //задаем команду - изменить фамилию туриста на ЧЧЧЧ под номером ЧЧЧЧ
50         //@-означает параметр
51         myCommand.CommandText = "UPDATE Туристы SET Фамилия = @Family WHERE [Код туриста] = @TouristID";
52         //задаем первый параметр - тип данных текст ---длина не больше 50 символов
53         myCommand.Parameters.Add("@Family", OleDbType.VarChar, 50);
54         //присваиваем значение параметру из переменной
55         myCommand.Parameters["@Family"].Value = Family;
56         myCommand.Parameters.Add("@TouristID", OleDbType.Integer, 4);
57         myCommand.Parameters["@TouristID"].Value = TouristID;
58
59         int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery(); //возвращает количество измененных строк
60         if (UspeshnoeIzmenenie != 0) //если хотя бы одна строка изменена
```

```
60     if (UspeshnoeIzmenenie != 0) //если хотя бы одна строка изменена
61     {
62         MessageBox.Show("Изменения внесены", "Изменение записи");
63     }
64     else
65     { MessageBox.Show("Не удалось внести изменения", "Изменение записи"); }
66
67     conn.Close(); //разорвать подключение
68 }
69 catch (Exception ex)
70 {
71     MessageBox.Show(ex.ToString());
72 }
73 finally
74 {
75     conn.Close(); //при ошибке база данных все равно будет корректна отключена }
76
77 }
78 }
```


Вставка Insert

```
34 private void btnCommand1_Click(object sender, EventArgs e)
35 {
36     OleDbConnectionStringBuilder builder = new OleDbConnectionStringBuilder();
37     builder.Provider = "Microsoft.ACE.OLEDB.12.0";
38     builder.DataSource = @"Contacts.accdb";
39     OleDbConnection conn = new OleDbConnection(builder.ConnectionString);
40     conn.Open();//открываем соединение
41     try
42     {
43         int TouristID = int.Parse(this.TextBox3.Text);
44         string Family = Convert.ToString(this.TextBox4.Text);
45         string FirstName = Convert.ToString(this.TextBox5.Text);
46         string MiddleName = Convert.ToString(this.TextBox6.Text);
47
48         OleDbCommand myCommand = conn.CreateCommand();
49         myCommand.CommandText = "INSERT INTO " +
50             "Туристы ([Код туриста], Фамилия, Имя, Отчество) " +
51             "VALUES (@TouristID, @Family, @FirstName, @MiddleName)";
52         //Параметр - номер туриста
53         myCommand.Parameters.Add("@TouristID", OleDbType.Integer, 4);
54         myCommand.Parameters["@TouristID"].Value = TouristID;
55         //Параметр - фамилия
56         myCommand.Parameters.Add("@Family", OleDbType.VarChar, 50);
57         myCommand.Parameters["@Family"].Value = Family;
58         //Параметр - Имя
59         myCommand.Parameters.Add("@FirstName", OleDbType.VarChar, 50);
60         myCommand.Parameters["@FirstName"].Value = FirstName;
```

```
61 //Параметр - Отчество
62 myCommand.Parameters.Add("@MiddleName", OleDbType.VarChar, 50);
63 myCommand.Parameters["@MiddleName"].Value = MiddleName;
64
65 int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery(); //возвращает количество измененных строк
66 if (UspeshnoeIzmenenie != 0) //если хотя бы одна строка изменена
67 {
68     MessageBox.Show("Изменения внесены", "Изменение записи");
69 }
70 else
71 { MessageBox.Show("Не удалось внести изменения", "Изменение записи"); }
72
73 conn.Close(); //разорвать подключение
74 }
75 catch (Exception ex)
76 {
77     MessageBox.Show(ex.ToString());
78 }
79 finally
80 {
81     conn.Close(); //при ошибке база данных все равно будет корректна отключена }
82
83 }
84 }
```



```
34 private void btnCommand3_Click(object sender, EventArgs e)
35 {
36     OleDbConnectionStringBuilder builder = new OleDbConnectionStringBuilder();
37     builder.Provider = "Microsoft.ACE.OLEDB.12.0";
38     builder.DataSource = @"Contacts.accdb";
39     OleDbConnection conn = new OleDbConnection(builder.ConnectionString);
40     conn.Open();//открываем соединение
41     try
42     {
43         int TouristID = int.Parse(this.TextBox7.Text);
44
45         OleDbCommand myCommand = conn.CreateCommand();
46         myCommand.CommandText = "DELETE FROM Туристы" +
47             "WHERE [Код туриста] = @TouristID";
48         //параметр - номер туриста
49         myCommand.Parameters.Add("@TouristID", OleDbType.Integer, 4);
50         myCommand.Parameters["@TouristID"].Value = TouristID;
51
52         int UspeshnoeIzmenenie = myCommand.ExecuteNonQuery(); //возвращает количество измененных строк
53         if (UspeshnoeIzmenenie != 0) //если хотя бы одна строка изменена
54         {
55             MessageBox.Show("Изменения внесены", "Изменение записи");
56         }
57         else
58         { MessageBox.Show("Не удалось внести изменения", "Изменение записи"); }
59
60         conn.Close(); //разорвать подключение
61     }
62     catch (Exception ex)
63     {
64         MessageBox.Show(ex.ToString());
65     }
66     finally
67     {
68         conn.Close(); //при ошибке база данных все равно будет корректна отключена }
69     }
70 }
71 }
```

Удаление Delete

Form1

Пример UPDATE

Пример INSERT

Пример DELETE

| Таблица - dbo.Туристы | | Сводка | | |
|-----------------------|-------------|-----------|---------|--------------|
| | Код туриста | Фамилия | Имя | Отчество |
| | 1 | Иванов | Василий | Степанович |
| | 2 | Николаев | Олег | Валентинович |
| ▶ | 3 | Сергеева | Инна | Вячеславовна |
| | 4 | Волков | Антон | Павлович |
| | 5 | Кириллова | Ольга | Михаиловна |
| * | NULL | NULL | NULL | NULL |

Form1

Пример UPDATE

Введите код туриста

Введите фамилию туриста

Обновить

Пример INSERT

Введите код туриста

Введите фамилию туриста

Введите имя туриста

Введите отчество туриста

Добавить

Пример DELETE

4

Удалить

Таблица - dbo.Туристы Сводка

| | Код туриста | Фамилия | Имя | Отчество |
|---|-------------|-----------|---------|--------------|
| ▶ | 1 | Иванов | Василий | Степанович |
| | 2 | Николаев | Олег | Валентинович |
| | 3 | Сергеева | Инна | Вячеславовна |
| | 5 | Кириллова | Ольга | Михаиловна |
| | 6 | Тихомиров | Андрей | Борисович |
| * | NULL | NULL | NULL | NULL |

Form1

Пример UPDATE

Введите код туриста

Введите фамилию туриста

Обновить

Пример INSERT

7

Иванов

Иван

Иванович

Добавить

Пример DELETE

4

Удалить

Таблица - dbo.Туристы Сводка

| | Код туриста | Фамилия | Имя | Отчество |
|---|-------------|-----------|---------|--------------|
| ▶ | 1 | Иванов | Василий | Степанович |
| | 2 | Николаев | Олег | Валентинович |
| | 3 | Сергеева | Инна | Вячеславовна |
| | 5 | Кириллова | Ольга | Михаиловна |
| | 6 | Тихомиров | Андрей | Борисович |
| | 7 | Иванов | Иван | Иванович |
| * | NULL | NULL | NULL | NULL |

Form1

Введите код туриста

Запуск

This screenshot shows a standard Windows application window titled "Form1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is white and contains a text box on the left with the placeholder text "Введите код туриста" (Enter tourist code) and a button on the right labeled "Запуск" (Start).

Form1

2 Николаев

Запуск

This screenshot shows the same "Form1" window after some data has been entered. The text box now contains the number "2", and the label "Николаев" is visible next to it. The "Запуск" button remains on the right side of the form.


```
private void btnCommand3_Click(object sender, EventArgs e)
{
    OleDbConnectionStringBuilder builder = new OleDbConnectionStringBuilder();
    builder.Provider = "Microsoft.ACE.OLEDB.12.0";
    builder.DataSource = @"Contacts.accdb";
    OleDbConnection conn = new OleDbConnection(builder.ConnectionString);
    conn.Open();//открываем соединение

    try
    {
        OleDbCommand myCommand = conn.CreateCommand();
        myCommand.CommandType = CommandType.StoredProcedure;
        myCommand.CommandText = "SELECT @LastName FROM Туристы WHERE [Код туриста] = @TouristID";
        int TouristID = int.Parse(this.TextBox1.Text);
        myCommand.Parameters.Add("@TouristID", OleDbType.Integer, 4);
        myCommand.Parameters["@TouristID"].Value = TouristID;

        myCommand.Parameters.Add("@LastName", OleDbType.VarChar, 60);
        myCommand.Parameters["@LastName"].Direction = ParameterDirection.Output;
        myCommand.ExecuteScalar();
        label1.Text = Convert.ToString(myCommand.Parameters["@LastName"].Value);

        conn.Close(); //разорвать подключение
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
    finally
    {
        conn.Close(); //при ошибке база данных все равно будет корректна отключена
    }
}
```

```
private void btnCommand3_Click(object sender, EventArgs e)
{
    OleDbConnectionStringBuilder builder = new OleDbConnectionStringBuilder();
    builder.Provider = "Microsoft.ACE.OLEDB.12.0";
    builder.DataSource = @"Contacts.accdb";
    OleDbConnection conn = new OleDbConnection(builder.ConnectionString);
    conn.Open();//открываем соединение
try
{
    int TouristID = int.Parse(this.TextBox1.Text);

    OleDbCommand myCommand = conn.CreateCommand();
    myCommand.CommandText = "SELECT Фамилия FROM Туристы WHERE [Код туриста] = @TouristID";

    myCommand.Parameters.Add("@TouristID", OleDbType.Integer, 4);
    myCommand.Parameters["@TouristID"].Value = TouristID;

    string result = myCommand.ExecuteScalar().ToString();
    label1.Text = result;

    conn.Close(); //разорвать подключение
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString());
}
finally
{
    conn.Close(); //при ошибке база данных все равно будет корректна отключена
}
}
```