О. Н. Шварцкоп

ЯЗЫКИ И СИСТЕМЫ ПРОГАММИРОВАНИЯ

Создание приложений для работы с базами данных

Учебно-методическое пособие

Челябинск
2023

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»

О. Н. Шварцкоп

ЯЗЫКИ И СИСТЕМЫ ПРОГАММИРОВАНИЯ

Создание приложений для работы с базами данных

Учебно-методическое пособие

Челябинск
2023

УДК 681.14(021)
ББК 32.973.2-018я73
Ш 33

Шварцкоп О.Н. Языки и системы программирования: создание приложений для работы с базами данных: учебно-методическое пособие / О.Н. Шварцкоп. – Челябинск: Изд-во ЗАО «Библиотека А. Миллера», 2023. – 34 с.

ISBN 978-5-93162-421-1

Учебно-методическое пособие содержит методические рекомендации для выполнения лабораторных работ по дисциплине «Языки и системы программирования» раздела «Создание приложений для работы с базами данных». Рассмотрены способы создания приложений для работы с базами данных, методы доступа к базам данных и применяемые в них программные интерфейсы. Также рассматривается интеграция приложений Visual C# с СУБД Microsoft Access.

Рекомендуется студентам, обучающимся по программе бакалавриата направления подготовки 44.03.04 Профессиональное обучение (по отраслям), профиль «Информатика и вычислительная техника».

Рецензенты:
Диденко Г.А., к.п.н., доцент ФГБОУ ВО «Южно-Уральский государственный медицинский университет» Министерства здравоохранения Российской Федерации
Гафарова Е.А., к.п.н., ЮУрГГПУ

ISBN 978-5-93162-421-1

Содержание

Пояснительная записка

Дисциплина «Языки и системы программирования» относится к модулю части, формируемой участниками образовательных отношений, Блока 1 «Дисциплины/модули» основной профессиональной образовательной программы по направлению подготовки 44.03.04 «Профессиональное обучение (по отраслям)» (уровень образования бакалавр).

Изучение дисциплины «Языки и системы программирования» основано на знаниях, умениях и навыках, полученных при изучении обучающимися дисциплин образовательной программы общего среднего образования.

Цель изучения дисциплины: введение в проблематику, связанную с изучением языков программирования высокого уровня, методов разработки алгоритмов и программ и особенности реализации языков программирования.

Задачи дисциплины:

1) освоение методов описания синтаксических конструкций языков программирования, знание классификации и эволюции языков программирования;

2) освоение основных концепций языков программирования на примере языка C# и формирование способности к изучению других алгоритмических и процедурных языков программирования высокого уровня;

3) освоение основных концепций объектно-ориентированного программирования;

4) изучение жизненного цикла программного обеспечения и понимание работ, выполняемых на каждом из его этапов.

Учебно-методическое пособие нацелено на оказание помощи студентам в формировании профессиональных компетенций в соответствии

с Федеральным государственным образовательным стандартом высшего образования. Компетенции с детализациями представлены в таблице 1.

Таблица 1 − Список компетенций с детализациями по дисциплине «Языки и системы программирования»

№ п/п	Код и наименование индикатора достижения компетенции	Образовательные результаты по дисциплине
1	ПК.7.1 Знать методы и средства представления проектной идеи для решения профессиональных задач	З.1 технологии разработки алгоритмов и программ; З.2 основы объектно- ориентированного подхода к программированию;
2	ПК.7.2 Уметь применять методы и средства представления проектной идеи для решения профессиональных задач.	У.1 ставить задачу и разрабатывать алгоритм ее решения; У.2 работать с современными системами программирования, включая объектно-ориентированные.
3	ПК.7.3 Владеть методами и средствами представления проектной идеи для решения профессиональных задач.	В.1 навыками разработки и отладки программ не менее чем на одном из алгоритмических процедурных языков программирования высокого уровня; В.2 языками процедурного и объектно-ориентированного программирования.

В учебно-методическом пособии представлены лабораторные работы по разделу дисциплины «Языки и системы программирования»: Введение в ADO.Net. Создание базы данных.

В учебно-методическом пособии в содержание каждой лабораторной работы включены цели изучения темы, краткие теоретические сведения по изучаемой теме, ход работы, задания для самостоятельной работы, контрольные вопросы для повторения изученного материала.

Тема 1. Создание простейшей базы данных

Лабораторная работа 1. Создание простейшей базы данных при помощи dataGridView

Цель работы: Изучение компонента dataGridView, создание простейшей базы данных.

Ход работы:

Разместим на форме следующие элементы (рис. 1):

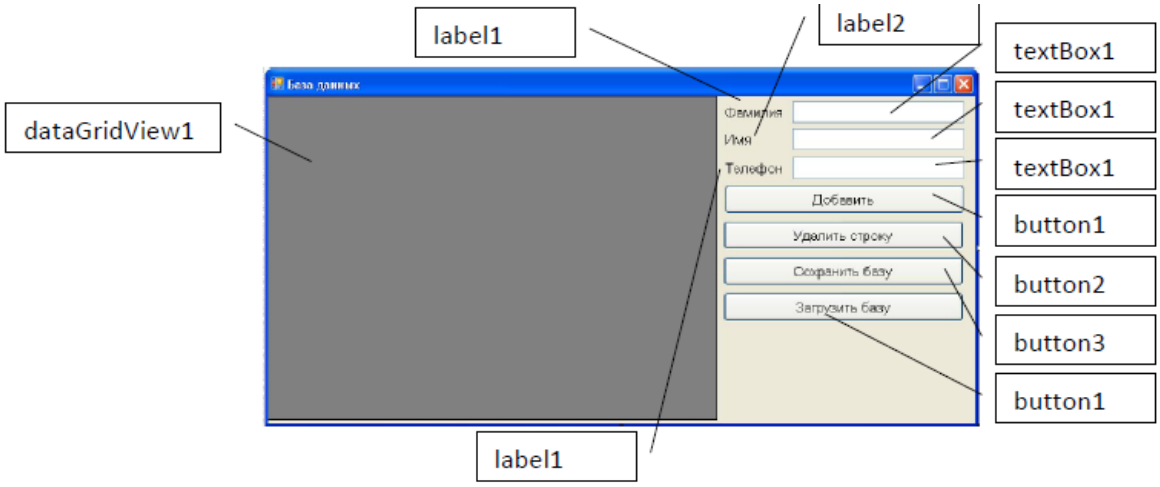


Рисунок 1. Элементы

Так как мы будем сохранять данные на диске, подключим пространство имен, отвечающее за эти операции:

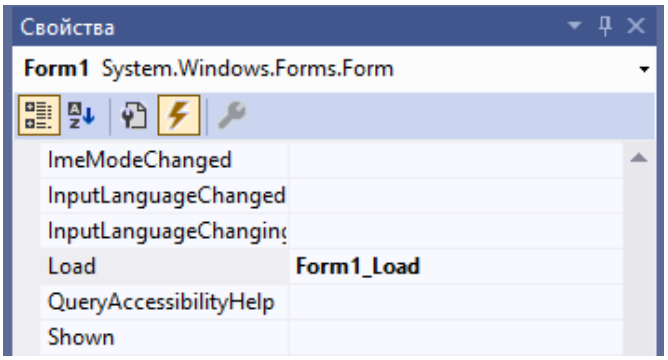
```
using System.Text;
using System.Windows.Forms;
using System.IO;
```

Опишем необходимые глобальные переменные:

```
public partial class Form1 : Form
    {
        int n, m,i,j;
        public Form1()
```

При старте программы у нас будет отображаться пустая таблица. Создадим событие, которое будет выполняться при загрузке формы. Выберем основную форму и перейдем в инспекторе объектов во вкладку

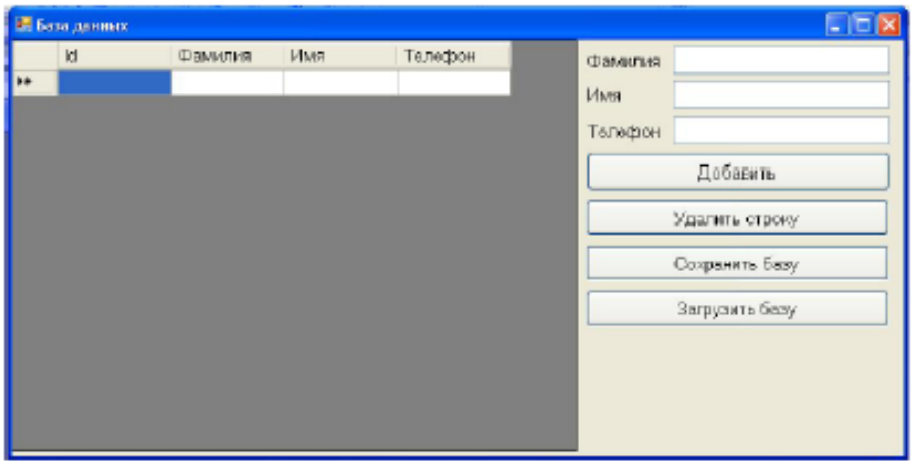
события. Напротив, слова Load дважды щелкнем мышкой и создадим это событие.



Запишем в него следующий код. В котором в пустую таблицу добавляются столбцы и подписываются их названия.

```
private void Form1_Load(object sender, EventArgs e)
        {
            // загрузка формы
             //создаём и добавляем столбцы
             dataGridView1.Columns.Add("ID", "Id");
             dataGridView1.Columns.Add("Fa", "Фамилия");
             dataGridView1.Columns.Add("Name", "Имя");
             dataGridView1.Columns.Add("Tel", "Телефон");
        }
```

При старте форма должна выглядеть следующим образом.



Пользователь может добавлять записи. Для этого он должен ввести данные в окна ввода и нажать кнопку «Добавить». Введенные строки можно будет удалять. Для этого нужно выделить строку целиком и нажать «Удалить строку».

Базу можно будет сохранять и читать из файла. Правда сохранение и чтение происходит из строго определенного файла. Выбор файла тоже можно подключить, используя диалоговые окна. В нашем проекте данные будут хранится в текстовом файле, в папке с исполняемым файлом проекта.

Создадим событие «Добавить».

```
private void button1_Click(object sender, EventArgs e)
{
 //Добавляем строку с номером rowNumber
 int rowNumber = dataGridView1.Rows.Add();
 //Заполняем ячейки
 dataGridView1.Rows[rowNumber].Cells["ID"].Value = rowNumber;
 dataGridView1.Rows[rowNumber].Cells[1].Value = textBox1.Text;
 dataGridView1.Rows[rowNumber].Cells[2].Value = textBox2.Text;
 // можно по номеру столбца а можно по идентефикатору
 dataGridView1.Rows[rowNumber].Cells["Tel"].Value=textBox3.Text;
 //Стираем поля ввода для новой информации
 textBox1.Text = "";
 textBox2.Text = "";
 textBox3.Text = "";
}
```

Создадим событие удалить строку. Хочу еще раз обратить внимание, чтобы удалить строку, ее нужно выделить. Выделение всей строки в DataGridView происходит только при нажатии на самый первый (левый) столбец. Но, многие пользователи не знают о выше приведенном способе и часто выделяют строку, просто нажимая на любую ячейку.

```
private void button2_Click(object sender, EventArgs e)
  {
   foreach (DataGridViewRow row in dataGridView1.SelectedRows)
       {
         dataGridView1.Rows.Remove(row);
       }

  }
```

Поэтому такой способ может не подойти, нам теперь нужно отловить не выделенную строку, а выделенную ячейку. Это мы можем сделать, например, следующим образом:

```
foreach (DataGridViewCell cell in dataGridView1.SelectedCells)
{
   dataGridView1.Rows.RemoveAt(cell.RowIndex);
}
```

Запись в файл и StreamWriter.

Для записи в текстовый файл используется класс StreamWriter. Свою функциональность он реализует через следующие методы:

1. Close: закрывает записываемый файл и освобождает все ресурсы.

2. Flush: записывает в файл оставшиеся в буфере данные и очищает буфер.

3. Write: записывает в файл данные простейших типов, как int, double, char, string и т.д.

4. WriteLine: также записывает данные, только после записи добавляет в файл символ окончания строки.

Создадим событие «Сохранить базу»:

```
private void button3_Click(object sender, EventArgs e)
{
   n = dataGridView1.RowCount;  // определяем число строк
   m = dataGridView1.ColumnCount; // определяем число столбцов
   //Создаём или перезаписываем существующий файл
   StreamWriter sw = File.CreateText("data.txt");
   //Записываем текст в поток файла
   sw.WriteLine(Convert.ToString(n));
   sw.WriteLine(Convert.ToString(m));
   for (i = 0; i < n; i++)
    {
       for (j = 0; j < m; j++)
        {
         sw.WriteLine(Convert.ToString(dataGridView1[j,i].Value));
        }
    }
    //Закрываем файл
    sw.Close();
 }
```

Чтение из файла и StreamReader

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла. Среди его методов можно выделить следующие:

1. Close: закрывает считываемый файл и освобождает все ресурсы

2. Peek: возвращает следующий доступный символ, если символов больше нет, то возвращает -1

3. Read: считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: Read(char[] array, int index,

int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов.

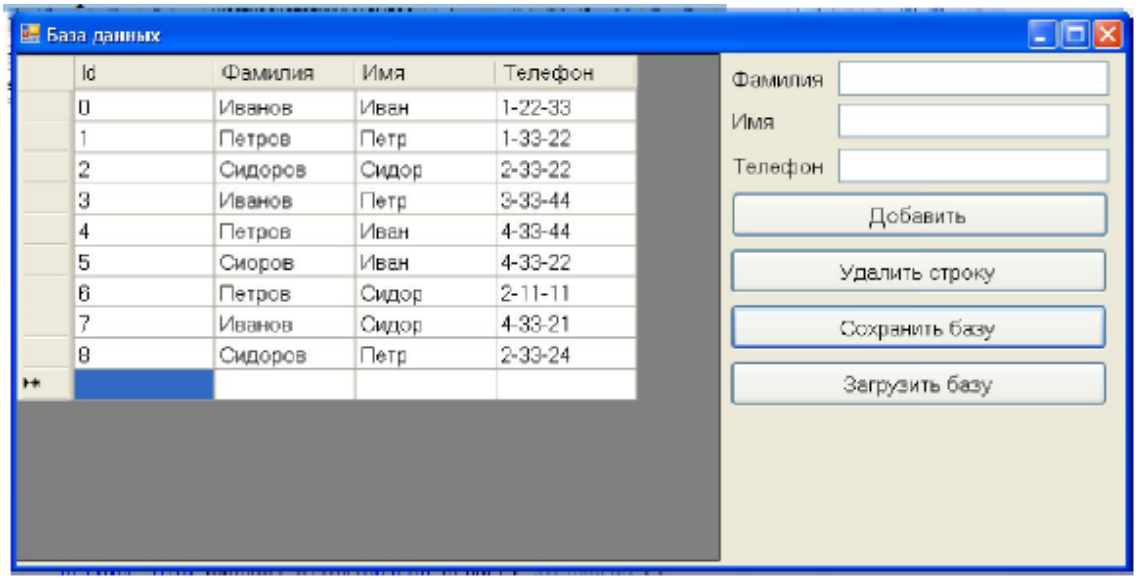
4. ReadLine: считывает одну строку в файле.

5. ReadToEnd: считывает весь текст из файла.

Создадим событие «Загрузить базу».

```
private void button4_Click(object sender, EventArgs e)
{
    //Создаем файловую переменную для чтения данных
    StreamReader f = new StreamReader("data.txt");
    //читаем первые две строки из файла
    n = Convert.ToInt32(f.ReadLine());
    m = Convert.ToInt32(f.ReadLine());
    //перебираем строки
    for (i = 0; i < n-1; i++)
    {
    int rowNumber = dataGridView1.Rows.Add();
    dataGridView1.Rows[rowNumber].Cells["ID"].Value=f.ReadLine();
    dataGridView1.Rows[rowNumber].Cells[1].Value = f.ReadLine();
    dataGridView1.Rows[rowNumber].Cells[2].Value = f.ReadLine();
    dataGridView1.Rows[rowNumber].Cells["Tel"].Value=f.ReadLine();
    }
    //Закрываем файл
    f.Close();
}
```

Программа в рабочем состоянии должна выглядеть так:



Тема 2. Подключение к базе данных

Лабораторная работа 2. Организация доступа к данным и работа с объектом DataReader

Цель работы: Изучение классов, предоставляющих службы доступа к данным и получение навыков по использованию компонентов ADO.NET.

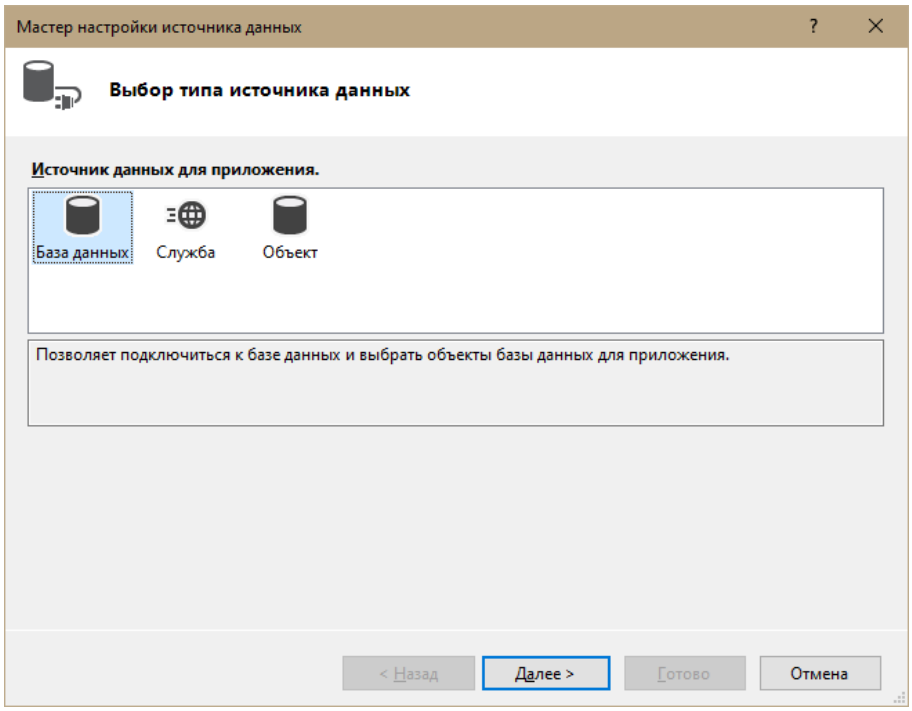
Ход работы:

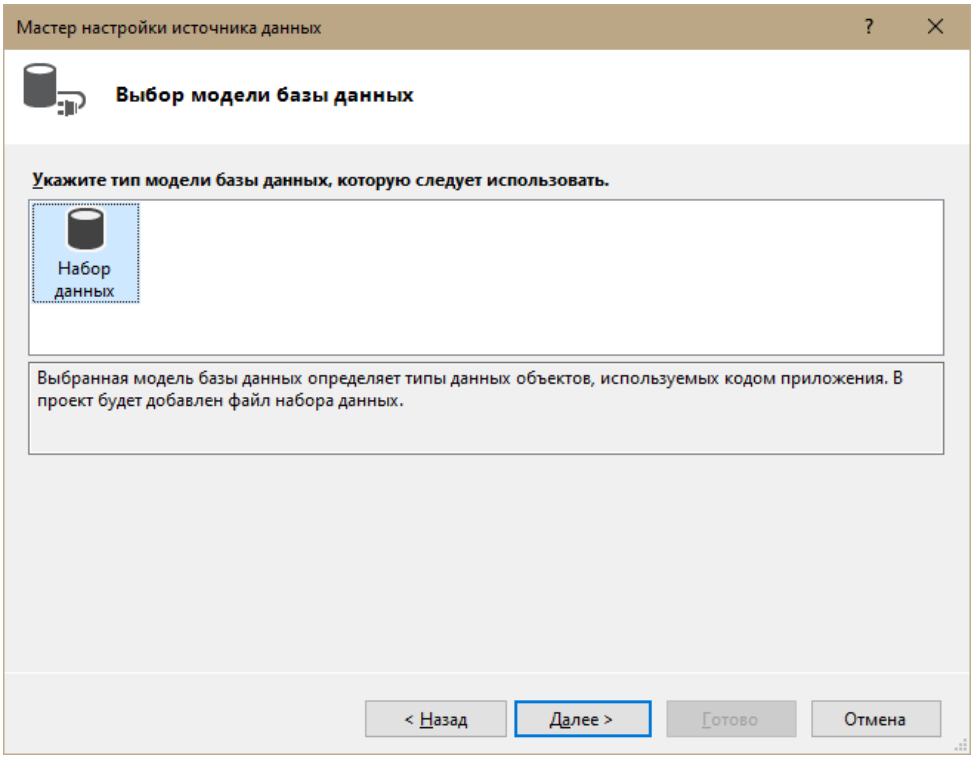
Создание нового соединения.

Прежде всего, следует создать соединение с БД при помощи окна Server Explorer.

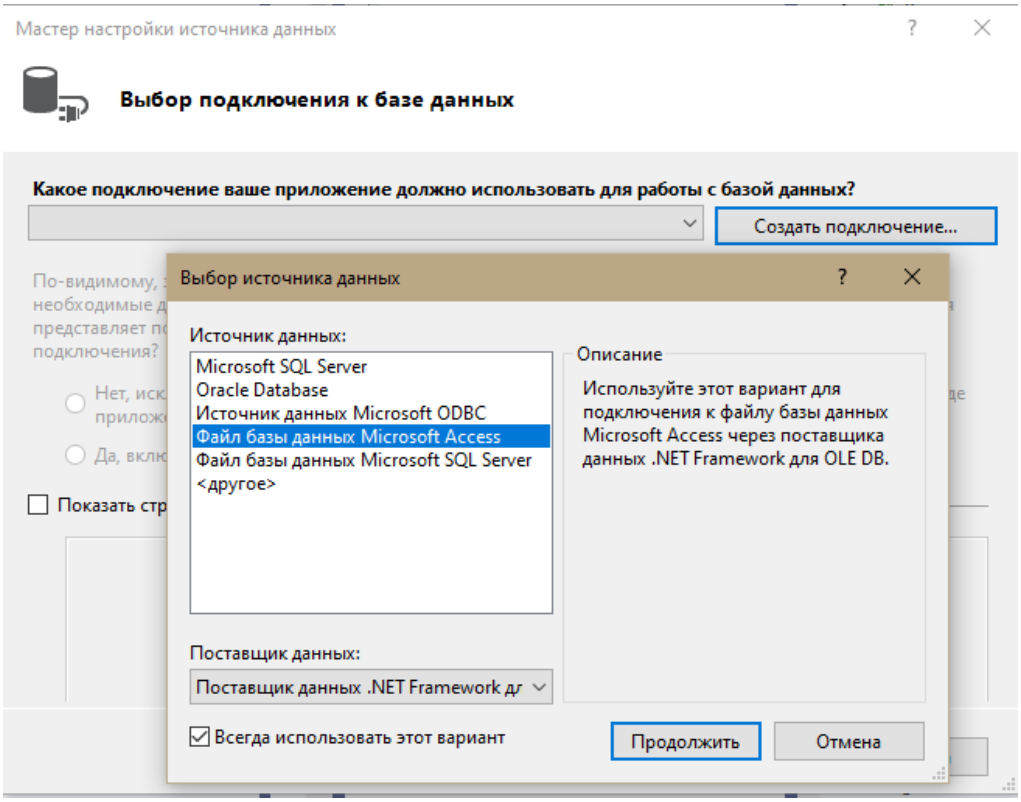
1. Создайте в Visual Studio новое приложение Windows Forms. Назовите его **WinBD.**

2. В окне **Источники данных** щелкните на **«Добавить новый источник данных…»** – при первом добавлении подключения откроется диалоговое окно **Выбор типа источника данных.** Если вместо диалогового окна **Выбор типа источника данных** появится диалоговое окно **Добавить соединение,** щелкните кнопку **Изменить,** в случае необходимости в изменении источника данных.





3. В диалоговом окне **Выбор подключения к базе данных** щелкните на **Создать подключение…** и выберите источник данных, к которому хотите подключиться - **Файл базы данных Microsoft Access,** а также провайдер данных для подключения. Обратите внимание, как нужный провайдер данных автоматически заполнился при указании источника данных.



4. **В окне** Добавить подключение **в поле** Имя файла базы данных **щелкните кнопку** Обзор...

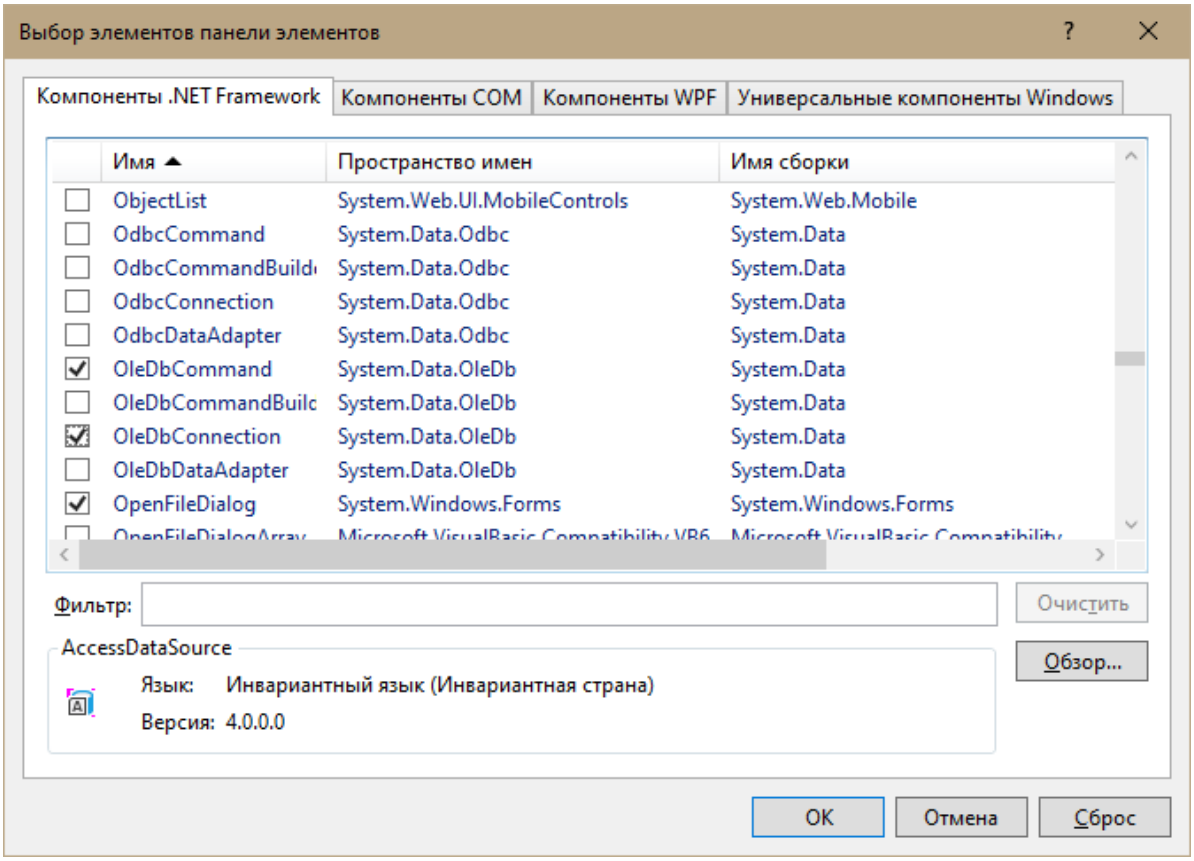
5. Вставьте любую базу данных, которая у Вас есть.

6. Щелкните кнопку **Проверить подключение**, чтобы проверить соединение, — должно появиться сообщение об успешной проверке. Если такого сообщения вы не получили, вернитесь к пункту 5 и убедитесь, что БД выбрана правильно.

7. Щелкните кнопку ОК — в окне **Выбор подключения к базе данных** появится новое соединение.

Создание и настройка объекта *DataCommand*

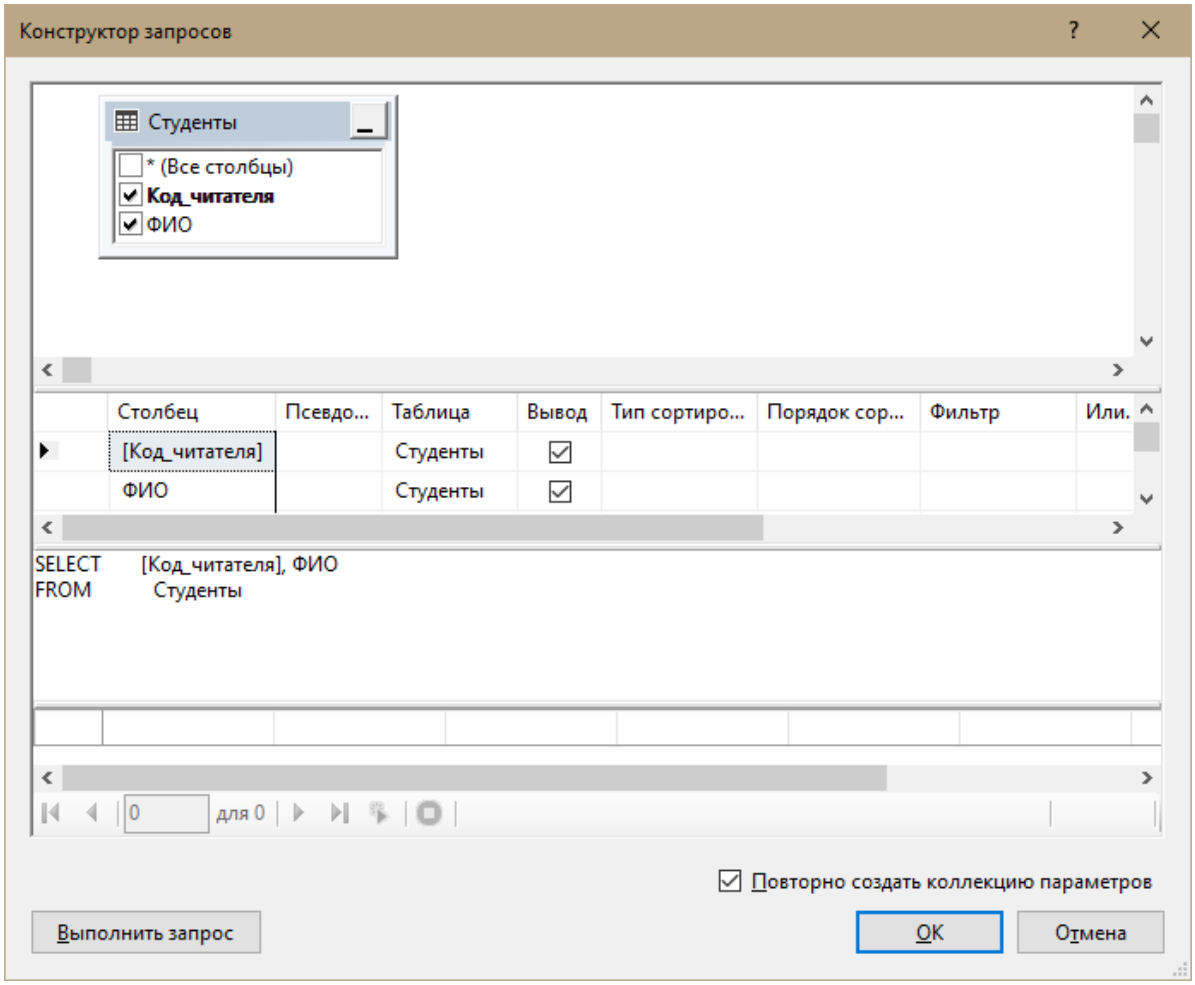
1. Добавьте на Панель инструментов компоненты **OleDbCotnmand** и **OleDbConnection. Для** этого щелкните правой кнопкой мыши на Панели инструментов и в контекстном меню выберите команду **Выбрать элементы.** В списке компонентов NET отметьте нужные компоненты и нажмите ОК.



2. Перетащите из Панель инструментов на форму экземпляр класса **OleDbConnection** — к приложению добавится новый объект с именем **oleDbConnectionl.** В свойстве **ConnectionString** этого объекта укажите в выпадающем списке свою базу данных.

3. Перетащите из Панель инструментов на форму экземпляр класса **OleDbCommand** — к приложению добавится новый объект **OleDbCotnmand** с именем **oleDbCommandl.**

4. Присвойте свойству **Connection** объекта **oleDbCommandl** значение oleDbConnectionl, выбрав его в списке **Существует**. Для свойства **CommandText** укажите SELECT * FROM Ваша таблица.

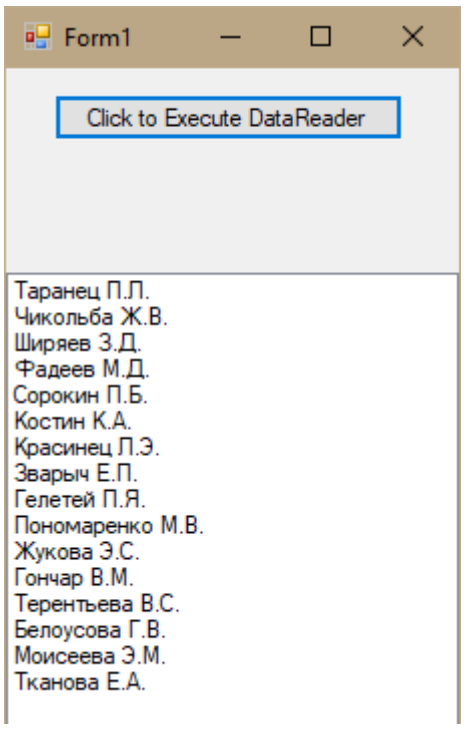


5. Перетащите из Панели инструментов на форму элементы управления **Button** (в верхнюю часть) и **ListBox** (в середину). Для **Buttonl** задайте свойству **Text** – «Click to Execute DataReader». Увеличьте ширину кнопки для оптимального размещения надписи. Для **ListBox** свойство **Dock** установите **Bottom.**

6. В окне дизайнера дважды щелкните объект **buttonl,** чтобы создать обработчик по умолчанию для события **buttonl.Click** и открыть его в редакторе кода. Добавьте в обработчик следующий код, извлекающий объект **DataReader** и заполняющий элемент управления **ListBox**:

```
System.Data.OleDb.OleDbDataReader myReader;
string CustomerString;
oleDbConnection1.Open();
myReader = oleDbCommand1.ExecuteReader();
while (myReader.Read()) {
CustomerString = myReader[1].ToString() + " "
+ myReader[2].ToString(); // Данная команда
берет данные из указанной таблицы в пункте 4.
myReader[1].ToString()- берет данные из 1
столбца
myReader[2].ToString() - берет данные из 2
столбца.
Если у вас в таблице только один столбец, то
вы используете myReader[n].ToString()один
раз, где n - номер столбца.
Если у вас больше столбцов, то вы прибавляете
столько myReader[n].ToString() сколько вам
нужно. Например, у вас 3 столбца: Фамилия,
Имя, Отчество. В коде вы должны написать:
CustomerString = myReader[1].ToString() + " "
+ myReader[2].ToString()+ " " +
myReader[3].ToString();
listBox1.Items.Add(CustomerString);
}
myReader.Close();
oleDbConnection1.Close();
```

7. Сохраните и протестируйте приложение. По щелчку кнопки элемент управления **ListBox** должен заполниться данными указанных вами столбцов**.**



8. Объекты **DataReader** быстро извлекают данные, но не позволяют модифицировать содержимое БД.

Лабораторная работа 3. Извлечение и обновление данных с помощью объектов DataAdapter и DataSet

Цель работы: Изучение классов, предоставляющих службы доступа к данным и получение навыков по использованию компонентов ADO.NET.

Ход работы:

В лабораторной работе вы реализуете доступ к данным для чтения и записи с помощью объектов **DataAdapter** и **DataSet.**

Вы должны заполнить объект **DataSet** с помощью объектов **DataAdapter,** связать **DataSet** с элементом управления **DataGrid** и обновить БД модифицированными данными.

Реализация доступа к БД для чтения и записи

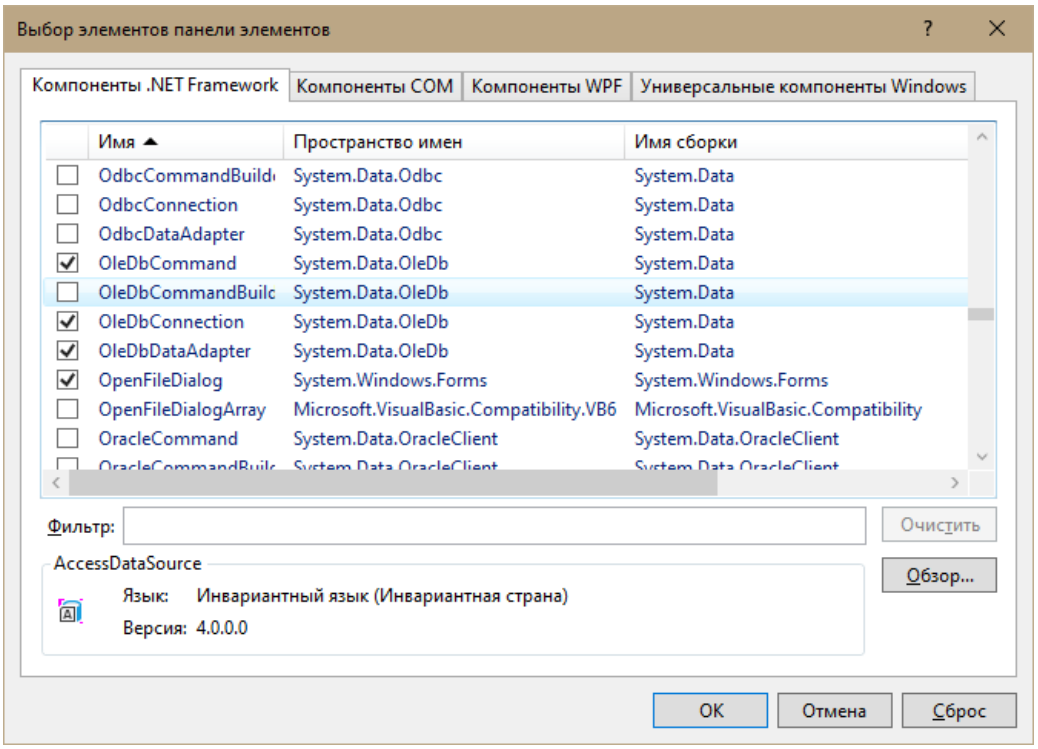
1. Перетащите из Панели элементов на поверхность формы элемент управления **Button** и установите свойству **Text** – «Click here for Exercise *2*». Увеличьте ширину кнопки для оптимального размещения надписи.

2. Дважды щелкните элемент управления **button2** чтобы создать обработчик по умолчанию для события **Button2. Click,** и добавьте к нему следующий код:

```
Form2 Exercise2 = new Form2();
Exercise2.Show();
```

3. В меню **Проект** выберите команду **Добавить форму Windows** и щелкните **Добавить,** чтобы добавить новую форму.

4. В этой версии Visual Studio объекты **DataAdapter** были удалены из Панели элементов, так что добавьте **OleDbDataAdapter** обратно в Панель элементов (см. предыдущее упражнение).



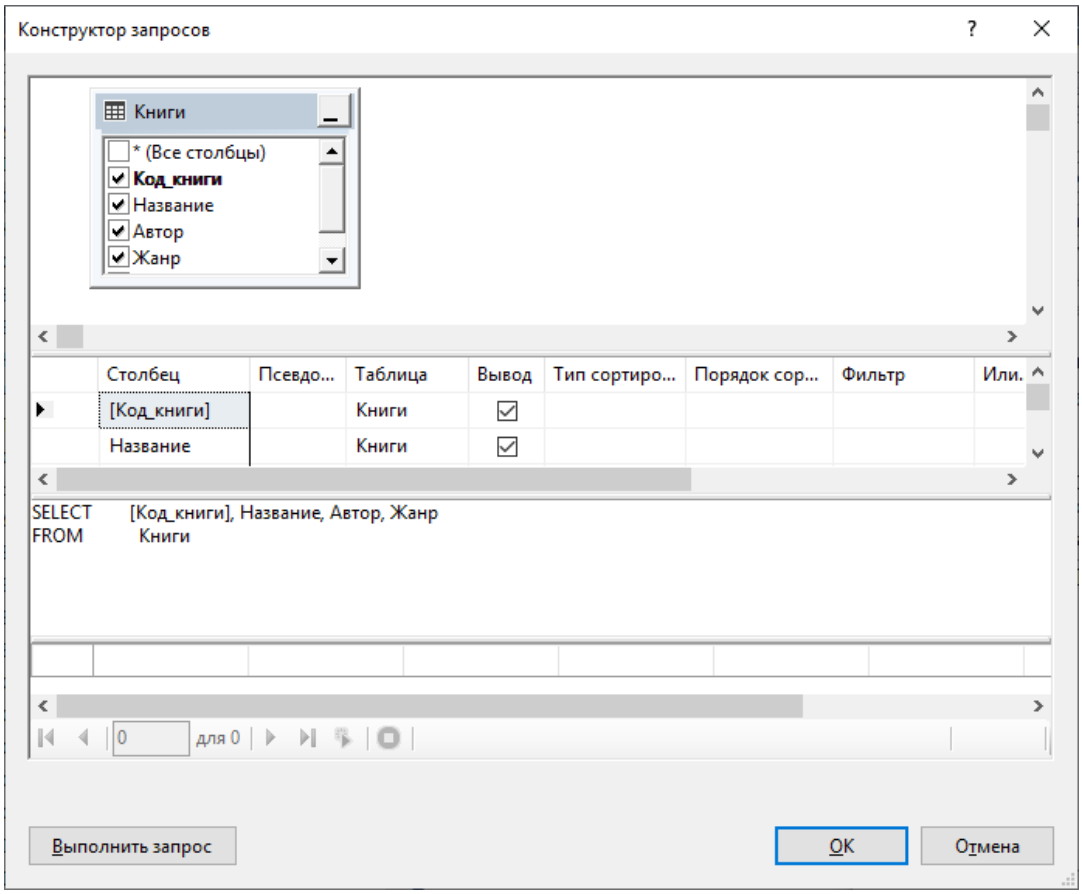
5. Перетащите объект **OleDbDataAdapter** на форму **Form2,** чтобы запустить **Мастер настройки адаптера данных.**

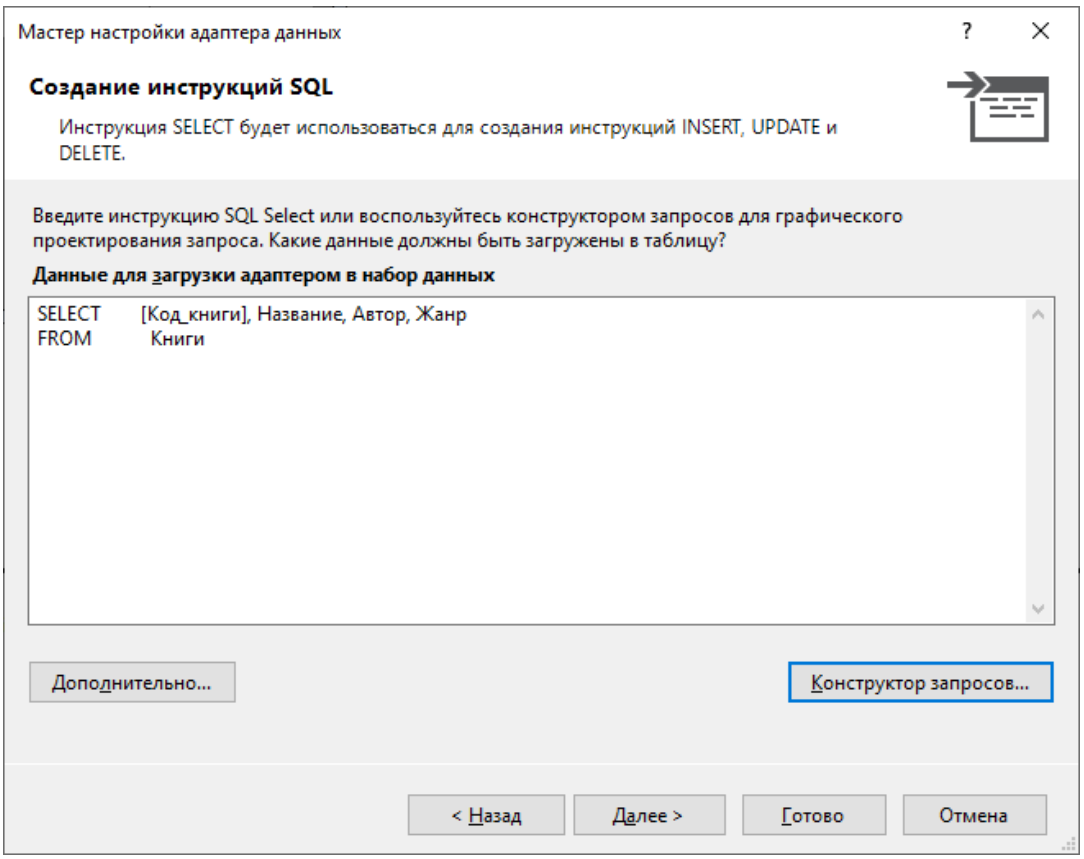
6. На странице **Выбор подключения баз данных** выберите подключение к базе данных **Студенческая библиотека_в2** (или создайте новое подключение в случае необходимости).

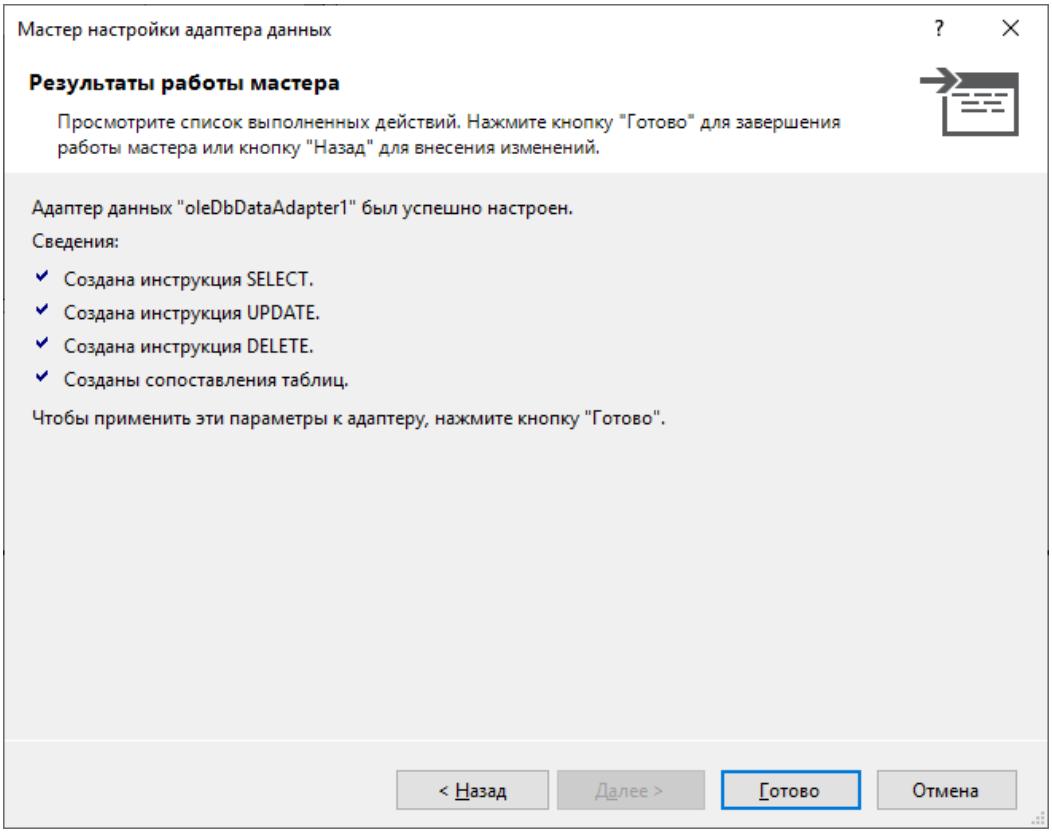
7. На странице **Выбор типа команды** оставьте настройку по умолчанию **Использовать инструкции SQL** и щелкните Далее.

8. На странице **Создание инструкций SQL** введите следующее предложение нажмите на **Конструктор запросов**… Правой кнопкой

откройте контекстное меню и выберите пункт **Добавить таблицу…** Выберите таблицу Книги, после галочками отметьте все столбцы таблицы. Нажмите ОК.



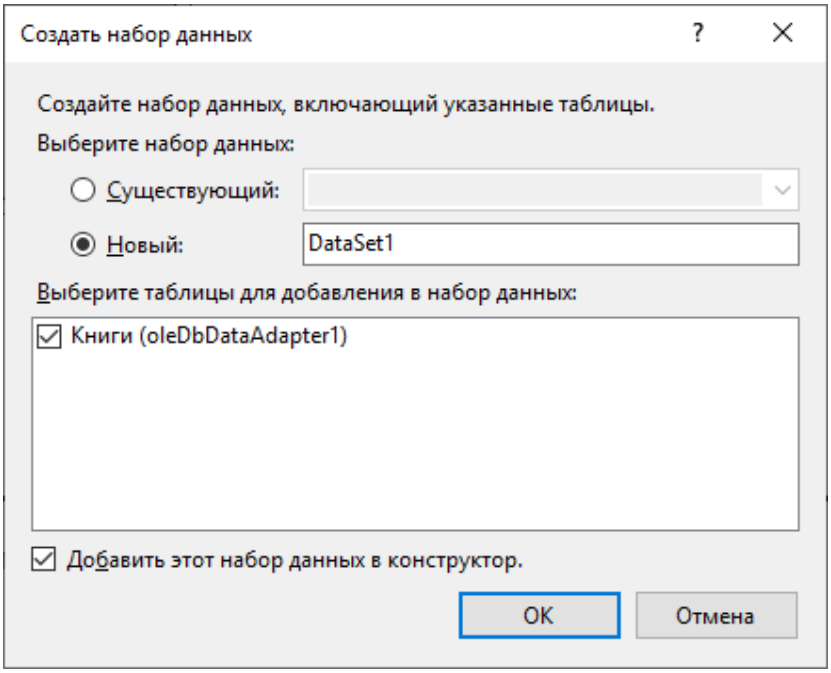

Мастер настройки адаптера данных
Создание инструкций SQL
Инструкция SELECT будет использоваться для создания инструкций INSERT, UPDATE и DELETE.
Введите инструкцию SQL Select или воспользуйтесь конструктором запросов для графического проектирования запроса. Какие данные должны быть загружены в таблицу?
Данные для загрузки адаптером в набор данных
SELECT [Код_книги], Название, Автор, Жанр
FROM Книги
Дополнительно...
Конструктор запросов...
< Назад
Далее >
Готово
Отмена

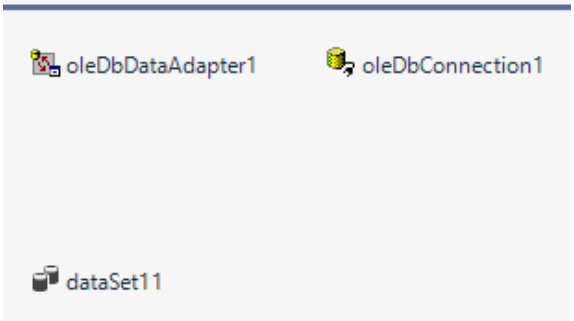

Мастер настройки адаптера данных
Результаты работы мастера
Просмотрите список выполненных действий. Нажмите кнопку "Готово" для завершения работы мастера или кнопку "Назад" для внесения изменений.
Адаптер данных "oleDbDataAdapter1" был успешно настроен.
Сведения:
Создана инструкция SELECT.
Создана инструкция UPDATE.
Создана инструкция DELETE.
Созданы сопоставления таблиц.
Чтобы применить эти параметры к адаптеру, нажмите кнопку "Готово".
< Назад
Далее >
Готово
Отмена

9. Щелкните Готово для завершения мастера и добавления экземпляра настроенного **OleDbDataAdapter** к форме.

10. Генерируйте строго типизированный **DataSet,** основанный на настроенном адаптере, для чего правой кнопкой мыши нажмите на **oleDbDataAdapter1** и в контекстном меню выберите **Создать набор данных** в меню Data.

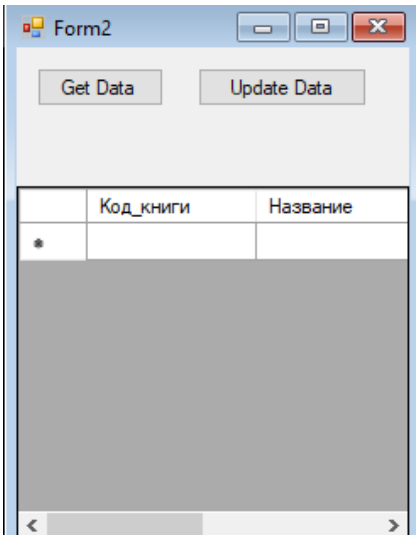
11. Щелкните OK для создания нового DataSet и добавления его к проекту. Обратите внимание на имя нового объекта - **dataSet11.**





12. Перетащите из Панели инструментов на форму **Form2** две кнопки и **DataGridView.** Установите для свойств, перечисленных в таблице, указанные в таблице значения:

Объект	Свойство	Значение
button1	Text	Get Data
button2	Text	Update Data
DataGridView	DataSource	dataSet11
	Dock	Bottom
	DataMember	Книги



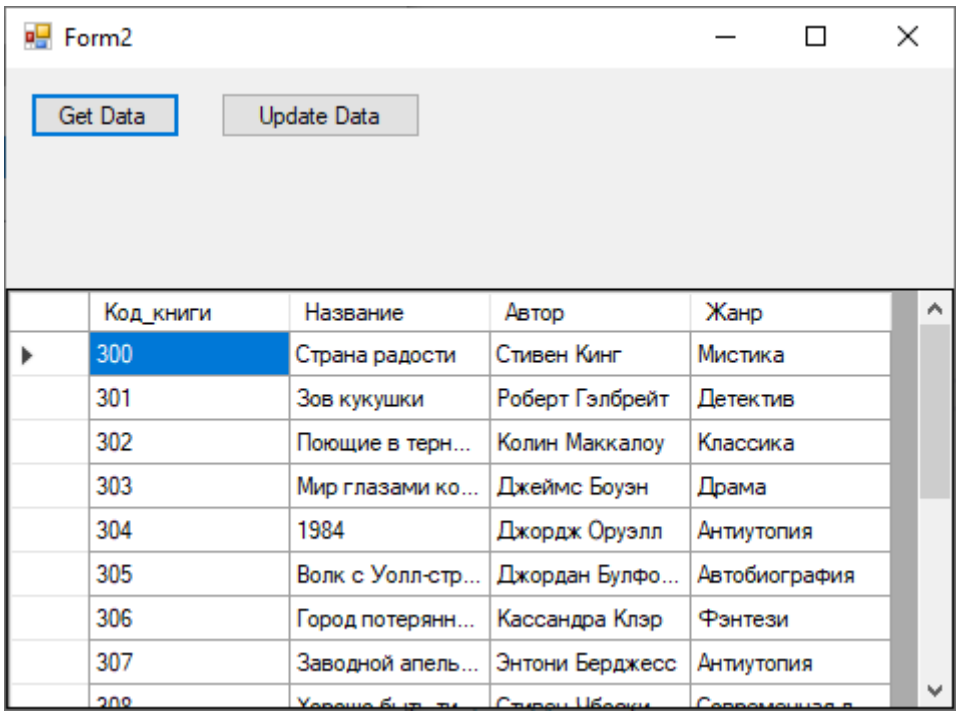
14. В окне конструктора дважды щелкните элемент управления **buttonl,** чтобы вызвать редактор кода с обработчиком по умолчанию для события **buttonl.Click.** Добавьте к нему следующий код:

```
oleDbDataAdapter1.Fill(dataSet11.Книги);
```

15. Аналогичным образом реализуйте обработчик события **button2.Click,** добавив следующий код:

```
oleDbDataAdapter1.Update(dataSet11);
```

16. Сохраните и протестируйте приложение. Открыв первую форму приложения, щелкните кнопку с надписью «Click here for Exercise 2», чтобы открыть созданную в этом форму. По щелчку кнопки с надписью «Get Data» в элемент управления **DataGrid** будут загружены данные.



Лабораторная работа 4. Использование объектов DataView

Цель работы: изучение объекта **DataView** для сортировки и фильтрации данных.

Ход работы:

1. Добавьте на форму два элемента **label** и свойству **Text** задайте значения *Сортировка* и *Фильтрация* соответственно.

2. Рядом с соответствующими элементами **label** расположите два элемента **TextBox.**

3. Для каждого элемента установите значения свойств:

Объект	Name	Text
textBox1	SortTextBox	Название
textBox2	FilterTextBox	Жанр='Поэма'

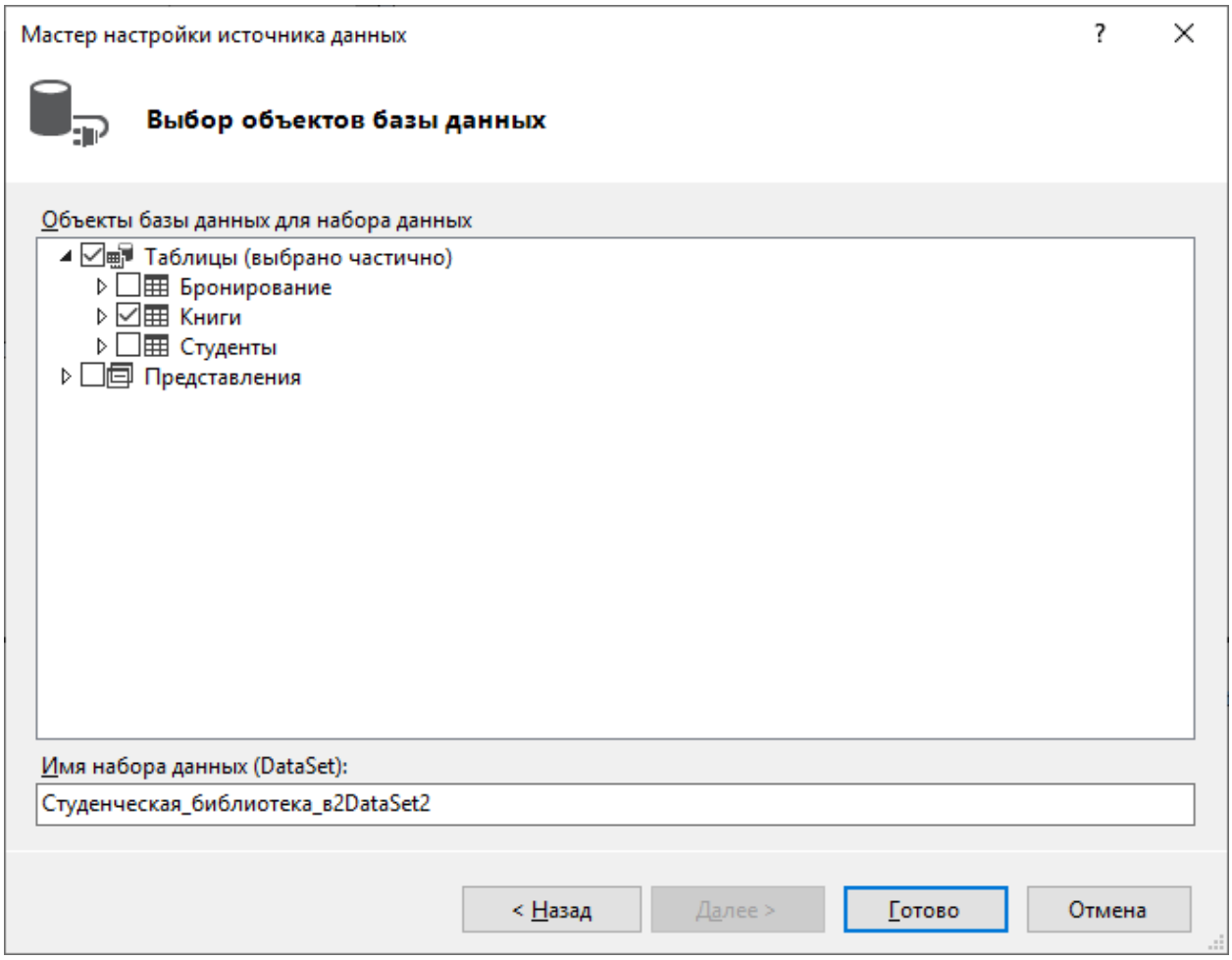
5. Создайте новый источник данных, выбрав Добавить новый источник данных в меню Data.

6. Выберите База данных и щелкните Далее.

7. Выберите допустимое подключение к своей базе данных.

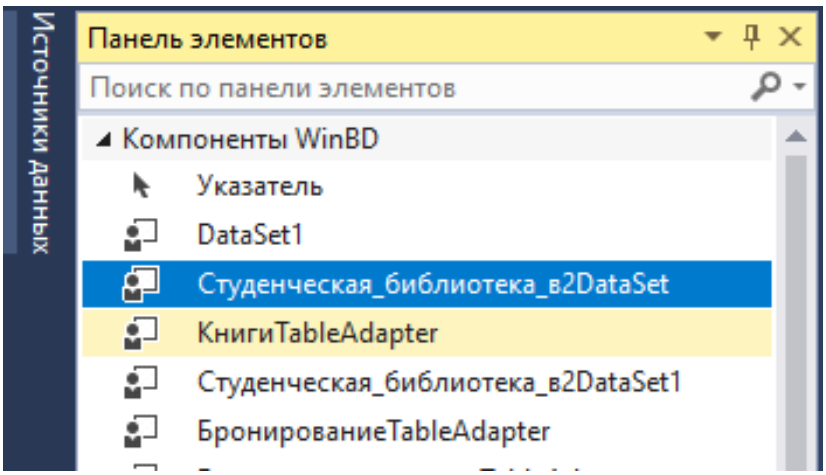
8. Выбирайте значения по умолчанию, пока не появится страница **Выбор объектов базы данных**.

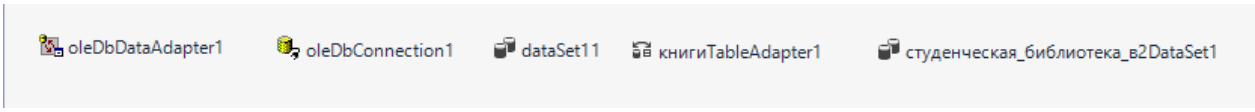
9. Выберите таблицу Книги и щелкните Готово.



10. Постройте проект.

11. Найдите в Панели инструментов компоненты **КнигиTableAdapter** и **Студенческая_библиотека_в2DataSet** и перетащите их на форму.





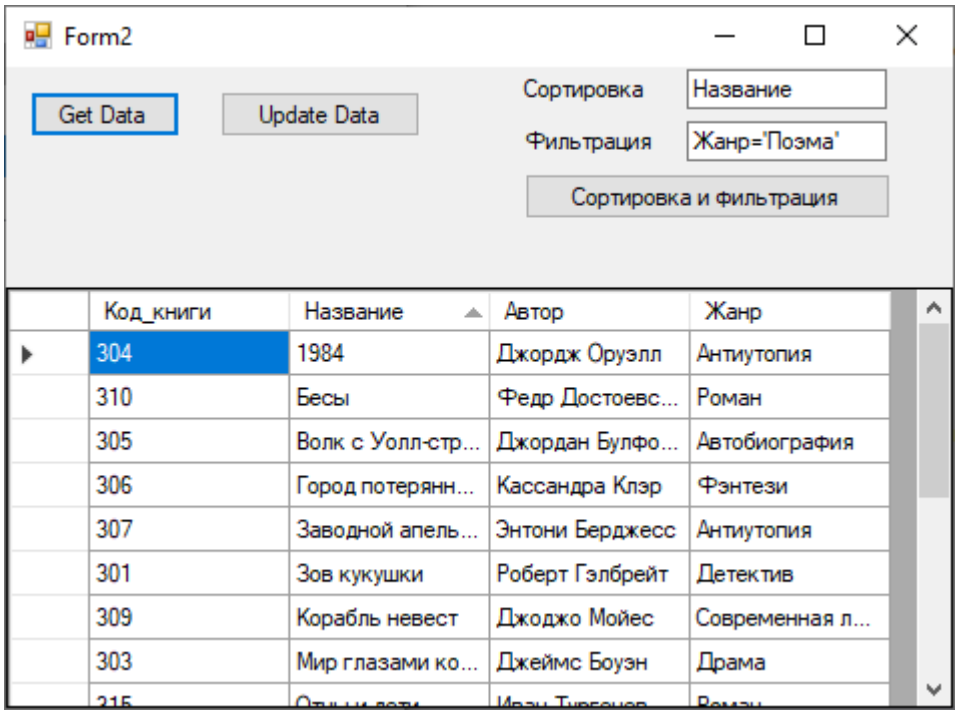
12. Создайте объект DataView для таблицы *Книги,* указав перед обработчиком события button1_Click следующий код:

```
DataView КнигиDataView;
```

13. Замените существующий код в обработчике события button1_Click на следующий:

```
КнигиTableAdapter1.Fill(студенческая_библиотека_в2D
ataSet1.Книги);
// Настройка объекта DataView
КнигиDataView = new
DataView(студенческая_библиотека_в2DataSet1.Книги);
// Настройка dataGridView для отображения данных
dataGridViewl.DataSource = КнигиDataView;
// Присвоения исходного порядка сортировки
КнигиDataView.Sort = "Название";
```

14. Постройте и выполните приложение. По нажатию кнопки «Get Data» должны загрузиться данные, отсортированные по столбцу Фамилия **(по тому столбцу, который вы указали в коде).**

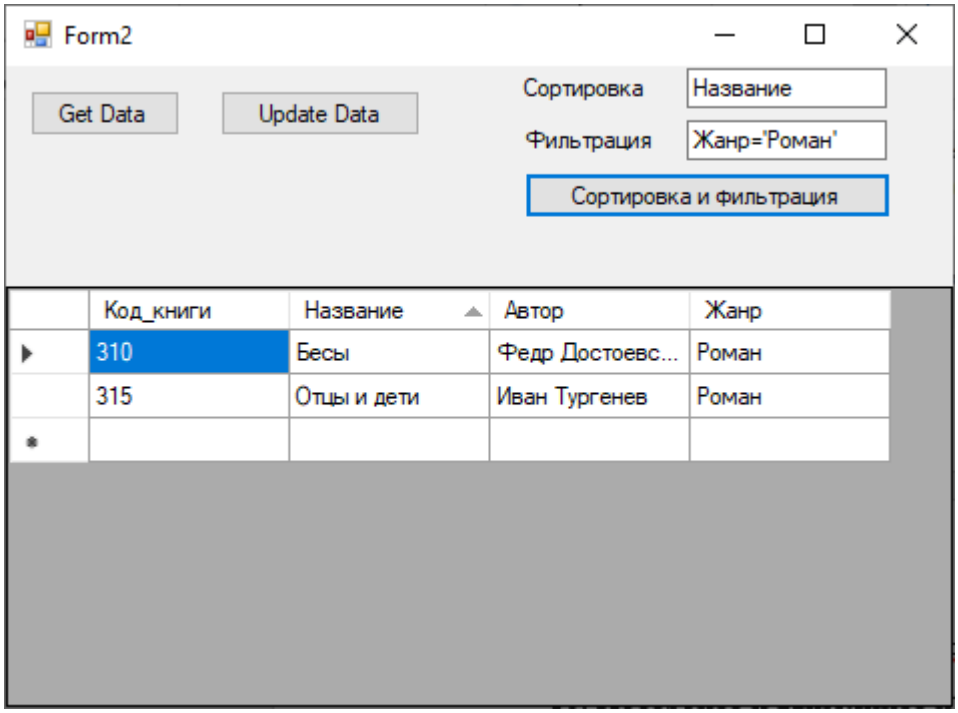


15. Расположите на форме кнопку с текстом «Сортировка и фильтрация», и добавьте следующий код к обработчику события щелчка кнопки:

```
КнигиDataView.Sort = SortTextBox.Text;
КнигиDataView.RowFilter = FilterTextBox.Text;
```

16. Постройте и выполните приложение. Загрузите данные и отфильтруйте их. Просмотрите результаты.

17. Измените жанр книги на 'Роман' и нажмите кнопку «Сортировка и фильтрация».



Лабораторная работа 5. Связывание данных с элементами управления

Цель работы: научиться связывать данные с элементами управления.

Ход работы:

Связывание элементов управления с данными является просто описанием процесса отображения данных (таких как данные из базы данных) в элементах управления Windows Forms.

Простое связывание данных описывает отображение одного элемента данных в элементе управления, например, отображение в **TextBox** значения одного столбца таблицы, типа названия компании.

Необходимо создать приложение Windows и выполнить простое связывание с данными элементов управления (связывание данных настраивается в коде).

1. Создайте приложение Windows и назовите его **WinDataBinding**.

Реализация доступа к БД

2. В окне Источники данных щелкните **Добавить новый источник данных…**

3. Оставьте выбранный по умолчанию База данных и щелкните Далее.

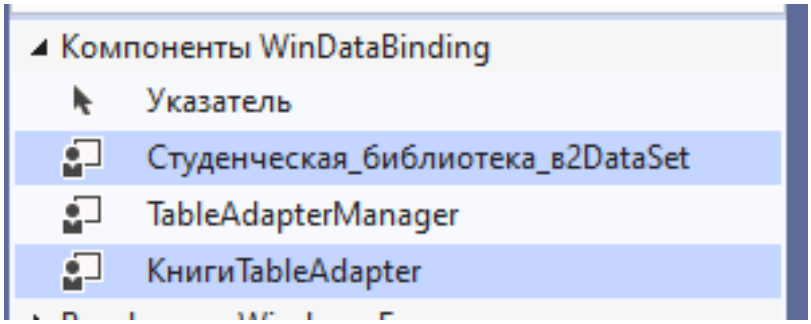
4. На странице **Выбор подключения к базе данных** создайте подключение к базе данных Студенческая библиотека.

5. Щелкайте Далее сохраняя значения по умолчанию, пока вы не дойдете до страницы **Выбор объектов базы данных**, где выберите таблицу Книги данных в узле *Таблицы*.

6. Щелкните **Готово** для добавления набора данных к вашему проекту.

7. Соберите проект.

8. Перетащите объекты **Студенческая_библиотека_в2DataSet** и **КнигиTableAdapter** из Панели элементов на форму.



Создание интерфейса для просмотра данных

Теперь, когда имеется набор данных (и **TableAdapter** для его заполнения), создайте несколько элементов управления для отображения данных набора данных) и перемещения по этим данным:

9. Добавьте три элемента управления **TextBox** к форме и присвойте их свойствам **Name** значение **FamtextBox, NametextBox** и **SectiontextBox** соответственно.

10. Слева от каждого элемента **TextBox** добавьте элемент **label** и укажите свойству **Text** значения: Название, Автор, Жанр соответственно.

11. Добавьте две кнопки для перемещения по записям.

12. Для первой кнопки установите следующие свойства: свойству **Name** значение *Previousbutton*, свойству **Text** значение *Previous.*

13. Для второй кнопки также установите свойства: **Name -** *Nextbutton*, **Text -** *Next.*

Настройка связывания данных

14. Дважды щелкните пустую область на форме, и создайте обработчик события Form1_Load.

15. Перед обработчиком Form1_Load объявите **BindingSource** для таблицы Книги:

```
private BindingSource sotrBindingSourse;
```

16. Добавьте код к обработчику события Form1_Load для настройки связывания данных:

```
// Загрузка таблицы данными:
книгиTableAdapter1.Fill(студенческая_библиотека_в2DataSet1.Книги);
// Создание BindingSource для таблицы Книги:
sotrBindingSourse = new
BindingSource(студенческая_библиотека_в2DataSet1, "Книги");
// Настройка связывания для элементов TextBox:
FamtextBox.DataBindings.Add("Text", sotrBindingSourse, "Название");
NametextBox.DataBindings.Add("Text", sotrBindingSourse, "Автор");
SectiontextBox.DataBindings.Add("Text", sotrBindingSourse, "Жанр");
```

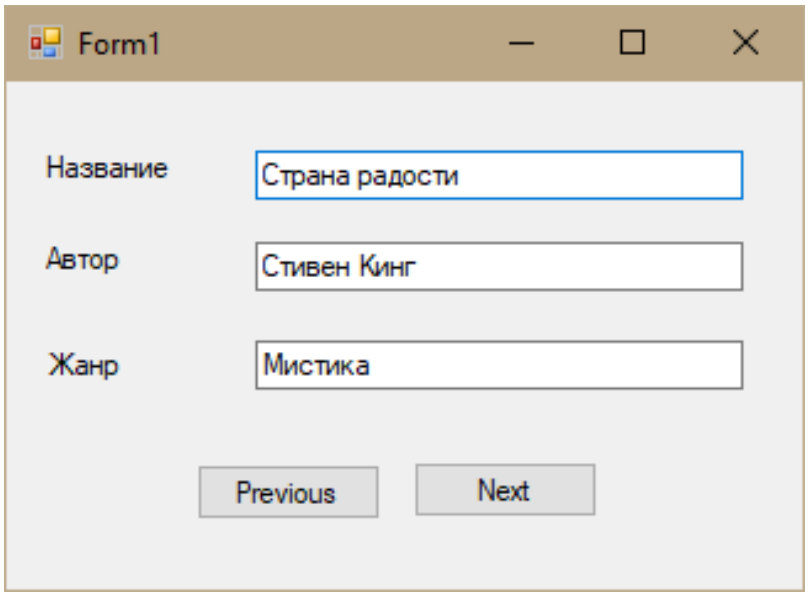
17. Дважды щелкните кнопку *Previous* и добавьте код, который перемещает к предыдущей записи в источнике данных *BindingSource:*

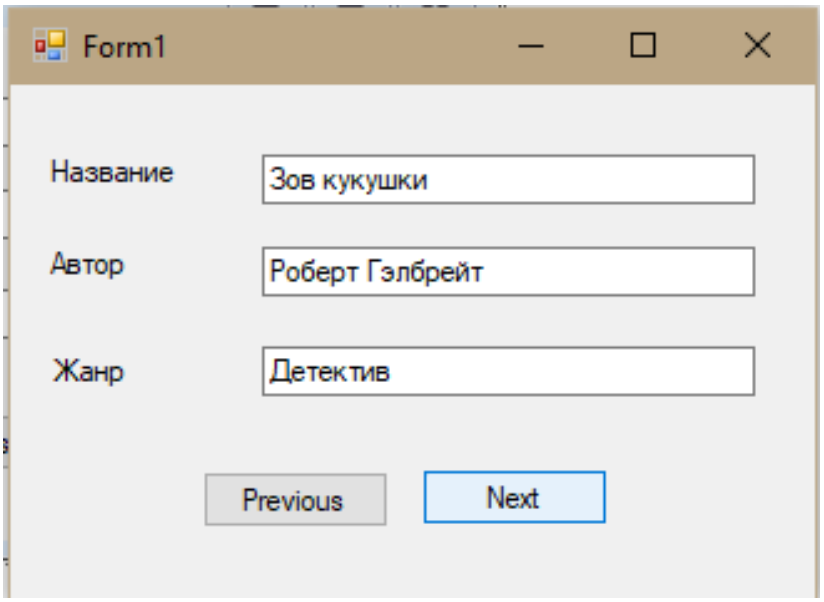
```
sotrBindingSourse.MovePrevious();
```

18. Дважды щелкните кнопку *Next* и добавьте код, который перемещает к следующей записи в источнике данных *BindingSource:*

```
sotrBindingSourse.MoveNext();
```

19. Постройте и запустите приложение. Протестируйте его работу.





Лабораторная работа 6. Создание связанной с данными формы в мастере источников данных

Цель работы: создание источников данных и связывание элементов управления с данными.

Ход работы:

Мастер источников данных (Data Source Configuration Wizard) создает в приложении типизированный **DataSet** и заполняет окно Источники данных объектами, выбранными во время работы мастера. После выполнения мастера остается еще переместить элементы в вашу форму для создания экземпляров объектов, которым необходим доступ к данным.

Необходимо создать приложение Windows, создать источник данных и связать элементы управления с данными, перетаскивая элементы из окна Источники данных.

1. Создайте приложение Windows и назовите его **WinDataSourcesWizard**.
2. Запустите Мастер настройки источника данных, выбрав **Добавить новый источник данных** в меню **Источники данных.**
3. На странице **Выбор типа источника данных** оставьте выбранный по умолчанию База данных и щелкните Далее.
4. На странице **Выбор подключения к базе данных** выберите подключение к базе данных Студенческая библиотека или создайте, если нужно, новое подключение.
5. Щелкайте Далее, сохраняя значения по умолчанию, пока не дойдете до страницы Choose Your Database Objects (Выбор объектов базы данных), и в узле **Tables** выберите таблицы *Бронирование* и *Студенты***.**
6. Щелкните Готово для добавления набора данных к проекту.
7. Перетащите узел *Студенты* из окна Источники данных на форму **Form1.**

8. К форме добавятся **DataGridView** и **BindingNavigator,** и в области компонентов появится несколько относящихся к данным объектов.

9. Постройте приложение.

10. В этот момент вы имеете рабочее приложение с **DataGridView,** связанным с данными таблицы *Студенты***.** Если в интегрированной среде разработки вы переключитесь в режим кода, то увидите, что был добавлен код к событию загрузки формы для заполнения таблицы С*туденты* данными, а в **BindingNavigator** — к сохраняющей данные кнопке для отправки обновлений обратно в базу данных.

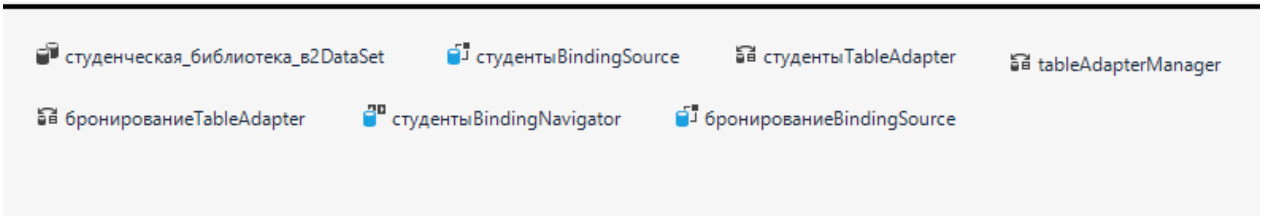
11. Запустите приложение. Выполняющееся приложение должно отобразить данные таблицы *Студенты.*

12. Остановите приложение и откройте форму в режиме Конструктор.

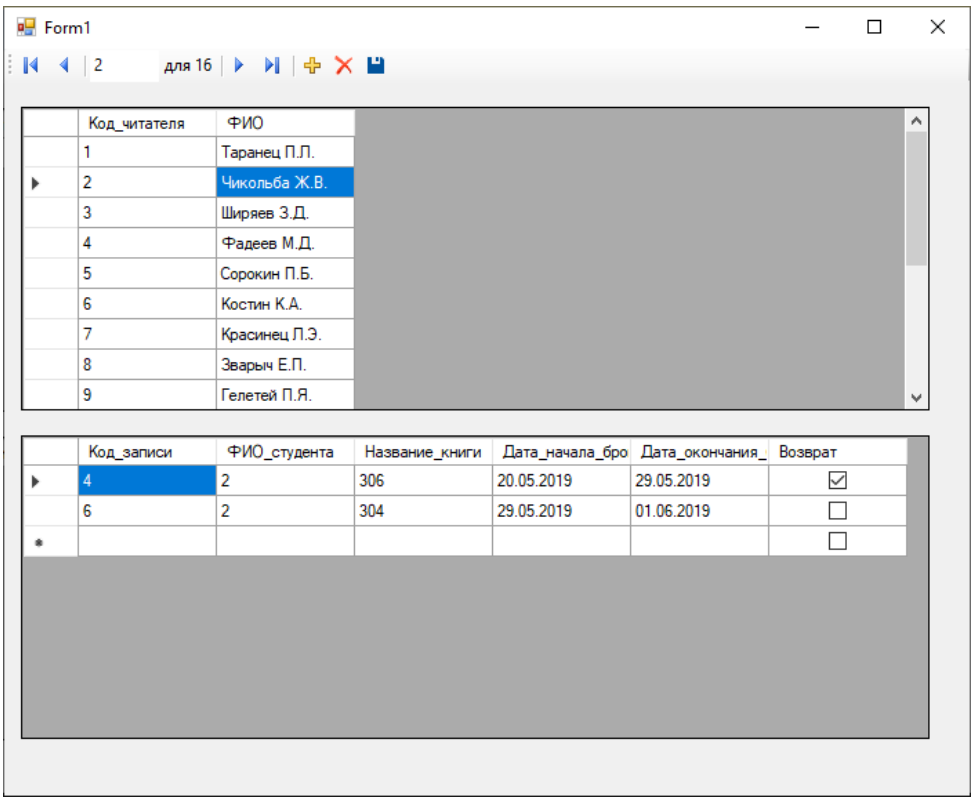
13. Разверните узел *Студенты* в окне Источники данных.

14. Перетащите на свободное место формы узел *Бронирование***,** вложенный в узел *Студенты.*

15. Обратите внимание на **BindingSource** и **TableAdapter,** добавленные в область компонентов.



17. Постройте и выполните приложение. Щелкните строку в таблице *Студенты.* Обратите внимание, что **Бронирование**DataGridView отображает все забронированные книги данного студента.


Form1
2
для 16
Код_читателя
ФИО
1
Таранец П.Л.
2
Чикольба Ж.В.
3
Ширяев З.Д.
4
Фадеев М.Д.
5
Сорокин П.Б.
6
Костин К.А.
7
Красинец Л.Э.
8
Зварыч Е.П.
9
Гелетей П.Я.
Код_записи
ФИО_студента
Название_книги
Дата_начала_бро
Дата_окончания_
Возврат
4
2
306
20.05.2019
29.05.2019
6
2
304
29.05.2019
01.06.2019

Глава 3. Задания для самостоятельной работы и проверочные материалы

Задание 1. Рассмотреть вопросы по теме:

1. Основы ADO .NET.
2. Принцип единообразной работы с базами данных.
3. ADO .NET. Объектная модель.
4. Объект DataTable.
5. Свойство Relations.
6. Класс Constraint.
7. Объект DataSet.
8. MVC (Model-View-Controller).

Задание 2. Письменно ответить на вопросы:

1. Каково назначение объектов классов DataSet, DataAdapter, DataTable, DataRelation?

2. Нарисуйте схему связей классов DataSet, DataTable, DataRelation; DataTable и DataRelation; DataColumn и DataTable; DataRow и DataTable.

3. Для чего нужен компонент BindingNavigator и как его использовать?

4. В чем различие визуальных и не визуальных компонент? Приведите пример их использования.

5. Как связываются объекты DataGridView и BindingNavigator с базой данных?

6. Как связываются текстовые поля TextBox с полями базы данных?

Задание 3. Содержание отчета и его форма.

Отчет по лабораторной работе должен содержать следующее.

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.

3. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

Задание 4. Создание проекта.

1. Разработка программы ведения учета материальных средств.
2. Разработка программы автоматизации работы диспетчера касс автовокзала.
3. Разработка приложения «Магазин».
4. Разработка автоматизированной информационной системы магазина электроники.
5. Разработка автоматизированной информационной системы «Заказы клиентов».
6. Разработка информационной системы для автоматизации продаж билетов железнодорожных касс.
7. Разработка электронного журнала куратора учебной группы.
8. Разработка приложения для учёта успеваемости студентов колледж.
9. Разработка приложения для автоматизации учета нагрузки преподавателей колледжа.
10. Разработка приложения для автоматизации управления финансами образовательной организации.
11. Разработка приложения для автоматизации управления кадрами образовательной организации.
12. Разработка информационной системы учета успеваемости студентов колледжа.

Библиографический список

1. Баженова, И. Ю. Введение в программирование: учебное пособие / И. Ю. Баженова, В. А. Сухомлин. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 326 с. – URL: http://www.iprbookshop.ru/97539.html (дата обращения: 10.10.2023).

2. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008): учебник / В. А. Биллиг. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 409 с. – URL: https://www.iprbookshop.ru/102029.html (дата обращения: 26.11.2023).

3. Кариев, Ч. А. Разработка Windows-приложений на основе Visual C#: учебное пособие / Ч. А. Кариев. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 978 с. – URL: https://www.iprbookshop.ru/102057.html (дата обращения: 16.11.2023).

4. Разработка приложений на C# в среде Visual Studio: учебное пособие / А. М. Нужный, Н. И. Гребенникова, В. Ф. Барабанов, О. Б. Кремер. – Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2019. – 89 с. – URL: https://www.iprbookshop.ru/93286.html (дата обращения: 26.10.2023).

5. Шварцкоп О.Н. Языки и системы программирования: рабочая программа дисциплины / О.Н. Шварцкоп. – URL: https://www.cspu.ru/sveden/files/RPD_-_YAzyki_i_sistemy_programmirovaniya.pdf (дата обращения: 15.11.2023).

Учебное издание

Шварцкоп Ольга Николаевна

ЯЗЫКИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ: СОЗДАНИЕ ПРИЛОЖЕНИЙ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ

Учебно-методическое пособие

Компьютерная верстка О.Н. Шварцкоп

Издательство ЗАО «Библиотека А.Миллера»
454091, г. Челябинск. Ул. Свободы, 159

Подписано в печать 24.11.2023 Формат 60х90/16
Объем 2,1 усл.-печ. л. Тираж 50 экз.
Заказ 321
Отпечатано с готового оригинал-макета
в типографии ЮУРГГПУ
454080, г. Челябинск, пр. Ленина, 69