Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет   
имени первого Президента России Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

**Отчёт по Лабораторной работе №3**

по дисциплине «Средства управления информационными ресурсами АС»

**Тема:** Создание вручную Windows-приложения для работы с БД

Научный руководитель Парфенов Ю. П.

Курс, группа РИЗ-300016у

Студент Кулаков М. И.

Екатеринбург, 2022

# СОДЕРЖАНИЕ

1 Постановка задачи и вариант задания 3

2 Образы экранных форм в конструкторе и исполняемой программе с пояснениями элементов управления 3

3 Подробное описание свойств и методов классов *DataTable*, *DbDataAdapter*, *DbCommand*, *DataGridview*, *DbcommandBuilder* 10

# 1 Постановка задачи

1. На третьей вкладке проекта индивидуального задания «Лабораторная работа №3» разместите компоненту ***DataGridView*** для отображения данных из главной таблицы индивидуального варианта задания.
2. Напишите программный код, создающий объекты, необходимые для отображения и управления данными из главной таблицы.
3. Приведите подробный комментарий для каждого использованного оператора программы.
4. Проверьте работу в форме по управлению данными.

# 2 Образы экранных форм в конструкторе и исполняемой программе с пояснениями элементов управления

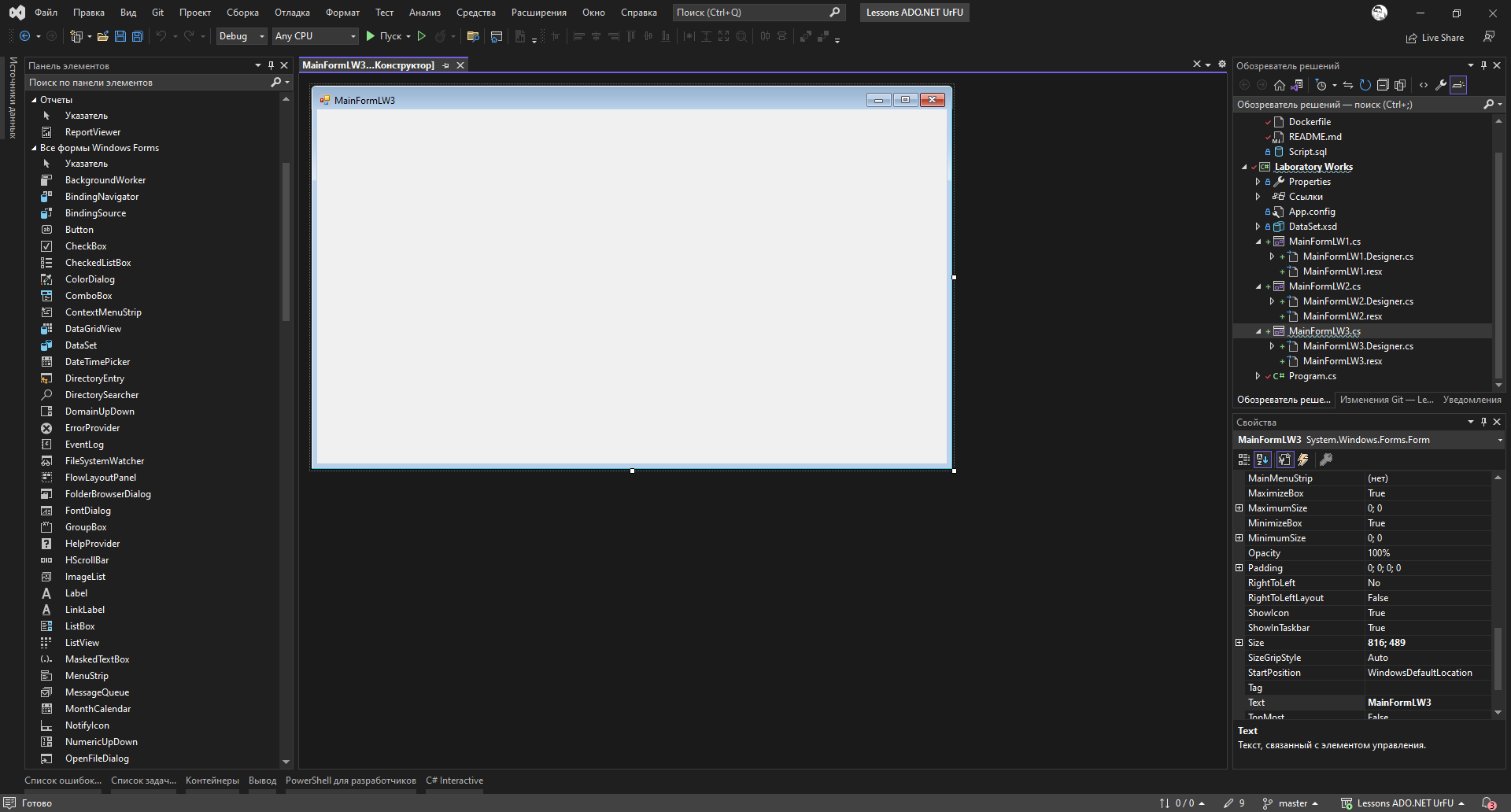


Рисунок №1 – Экранная форма в конструкторе

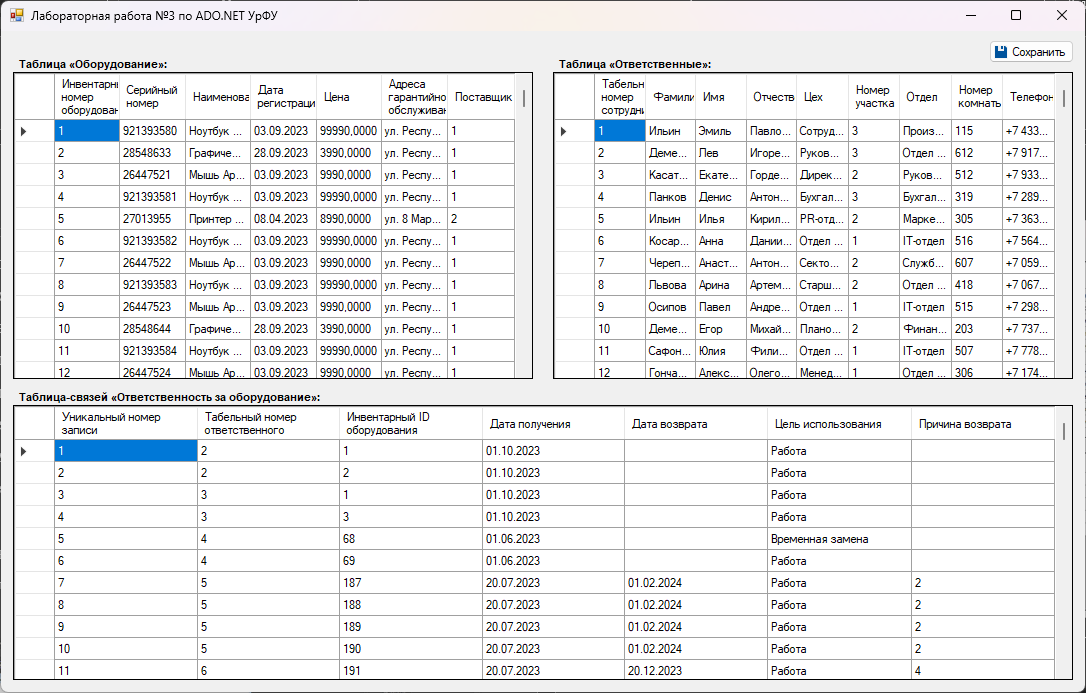


Рисунок №2 – Исполняемая программа

Согласно заданию, с помощью конструктора форм не были добавлены какие-либо объекты. Все объекты были добавлены на форму с помощью кода. На форме, по аналогии с первой лабораторной работой, было добавлено:

* три текстовые метки, класса ***Label***;
* три табличных представления данных, класса ***DataGridView***;
* одна кнопка для отправки изменений на SQL‑сервер, класса ***Button***.

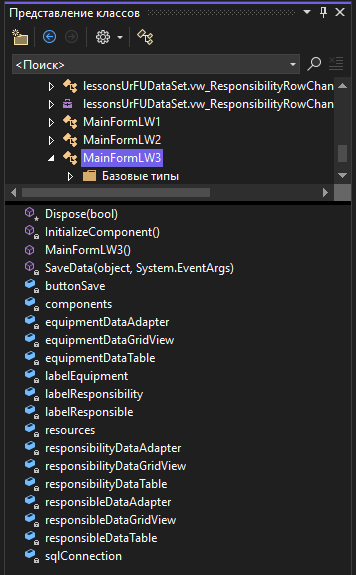


Рисунок №3 – Структура класса исполняемой программы

Объекты на форме также являются закрытыми полями класса **MainFormLW3**. Свойства объектов формы задаются при их инициализации. В конструкторе класса выполняется добавление объектов на форму, настройка свойств самой формы, а также подключение и получение данных с базы данных. Метод **SaveData** формирует SQL команды **UPDATE**, **INSERT**, DELETE в виде объектов класса ***SqlCommandBuilder***. После, с помощью объектов классов ***SqlConnection*** и ***SqlDataAdapter*** выполняет их отправку на сервер. Код класса **MainFormLW3**:

using System.Data.SqlClient;  
using System.Windows.Forms;  
using System.Data;  
using System;  
  
namespace LaboratoryWorks  
{  
 public sealed partial class LabWorkForm3 : Form  
 {  
 private static readonly System.ComponentModel.ComponentResourceManager Resources = new(typeof(LabWorkForm3));  
  
 */// <summary>  
 /// Соединение с базой данных  
 /// </summary>* private readonly SqlConnection \_sqlConnection = new ("Data Source=localhost;" +  
 "Initial Catalog=lessonsUrFU;" +  
 "Persist Security Info=True;" +  
 "User ID=sa;" +  
 "Password=KqGN?a9Yvi");  
  
  
 *// ------------------------------------  
 // Текстовые метки в качестве ---------  
 // заголовков перед таблицами ---------  
 // ------------------------------------  
  
 /// <summary>  
 /// Текстовая метка, в качестве заголовка таблицы «Оборудование»  
 /// </summary>* private readonly Label \_labelEquipment = new()  
 {  
 AutoSize = true,  
 Location = new Point(15, 26),  
 Text = "Таблица «Оборудование»:",  
 Font = new Font("Arial", 8.25F, FontStyle.**Bold**),  
 TabIndex = 0  
 };  
  
 */// <summary>  
 /// Текстовая метка, в качестве заголовка таблицы «Ответственные»  
 /// </summary>* private readonly Label \_labelResponsible = new()  
 {  
 AutoSize = true,  
 Location = new Point(555, 26),  
 Text = "Таблица «Ответственные»:",  
 Font = new Font("Arial", 8.25F, FontStyle.**Bold**),  
 TabIndex = 0  
 };  
  
 */// <summary>  
 /// Текстовая метка, в качестве заголовка таблицы связей «Ответственность за оборудование»  
 /// </summary>* private readonly Label \_labelResponsibility = new()  
 {  
 AutoSize = true,  
 Location = new Point(15, 359),  
 Text = "Таблица-связей «Ответственность за оборудование»:",  
 Font = new Font("Arial", 8.25F, FontStyle.**Bold**),  
 TabIndex = 0  
 };  
  
  
 *// ------------------------------------  
 // DatGridViews -----------------------  
 // ------------------------------------  
  
 /// <summary>  
 /// DataGridView для отображения столбцов и строк таблицы «Оборудование»  
 /// </summary>* private readonly DataGridView \_dataGridViewEquipment = new()  
 {  
 Size = new Size(520, 307),  
 Location = new Point(12, 41),  
 AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.**Fill**,  
 ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.**AutoSize**,  
 TabIndex = 1  
 };  
  
 */// <summary>  
 /// DataGridView для отображения столбцов и строк таблицы «Ответственные»  
 /// </summary>* private readonly DataGridView \_dataGridViewResponsible = new()  
 {  
 Size = new Size(520, 307),  
 Location = new Point(552, 41),  
 AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.**Fill**,  
 ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.**AutoSize**,  
 TabIndex = 2  
 };  
  
 */// <summary>  
 /// DataGridView для отображения столбцов и строк таблицы-связей «Ответственность за оборудование»  
 /// </summary>* private readonly DataGridView \_dataGridViewResponsibility = new()  
 {  
 Size = new Size(1060, 275),  
 Location = new Point(12, 374),  
 AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.**Fill**,  
 ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.**AutoSize**,  
 TabIndex = 3  
 };  
  
  
 *// ------------------------------------  
 // DataTable --------------------------  
 // ------------------------------------* private readonly DataTable \_dataTableEquipment = new();  
  
 private readonly DataTable \_dataTableResponsible = new();  
  
 private readonly DataTable \_dataTableResponsibility = new();  
  
  
 *// ------------------------------------  
 // SqlDataAdapter ---------------------  
 // ------------------------------------* private readonly SqlDataAdapter \_dataAdapterEquipment;  
  
 private readonly SqlDataAdapter \_dataAdapterResponsible;  
  
 private readonly SqlDataAdapter \_dataAdapterResponsibility;  
  
  
 *// ------------------------------------  
 // Кнопки -----------------------------  
 // ------------------------------------* private readonly Button \_buttonSave = new()  
 {  
 Location = new Point(988, 9),  
 Size = new Size(85, 23),  
 Text = "Сохранить",  
 TextAlign = ContentAlignment.**MiddleRight**,  
 Image = (Image)Resources.GetObject("save.Image"),  
 ImageAlign = ContentAlignment.**MiddleLeft**,  
 TabIndex = 4  
 };  
  
  
 *// ------------------------------------  
 // Конструктор ------------------------  
 // ------------------------------------* public LabWorkForm3()  
 {  
 InitializeComponent();  
  
 *// Настроки формы* this.Text = "Лабораторная работа №3 по ADO.NET УрФУ";  
 this.MinimumSize = new Size(1100, 700);  
 this.MaximumSize = new Size(1100, 700);  
 this.FormBorderStyle = FormBorderStyle.**FixedSingle**;  
 this.StartPosition = FormStartPosition.**CenterScreen**;  
  
 *// Добавление текстовых меток на форму* this.Controls.Add(\_labelEquipment);  
 this.Controls.Add(\_labelResponsible);  
 this.Controls.Add(\_labelResponsibility);  
  
 *// Добавление DataGridView-ов на форму* this.Controls.Add(\_dataGridViewEquipment);  
 this.Controls.Add(\_dataGridViewResponsible);  
 this.Controls.Add(\_dataGridViewResponsibility);  
  
 *// Добавление кнопки на форму* this.Controls.Add(\_buttonSave);  
  
  
 *// --------------------------------  
 // Работа с базой данных ----------  
 // --------------------------------  
  
 // Создание запросов на получение данных* var commands = new[]  
 {  
 new SqlCommand(@"SELECT InventoryID AS N'Инвентарный номер оборудования', SerialNumber AS N'Серийный номер', EquipmentName AS N'Наименование', RegistrationDate AS N'Дата регистрации', Price AS N'Цена', WarrantyServiceAddress AS N'Адреса гарантийного обслуживания', Supplier AS N'Поставщик' FROM Equipment;", \_sqlConnection),  
  
 new SqlCommand(@"SELECT PersonnelNumber AS N'Табельный номер сотрудника', Surname AS N'Фамилия', Name AS N'Имя', MiddleName AS 'Отчество', Workshop AS N'Цех', LotNumber AS N'Номер участка', Department AS N'Отдел', RoomNumber AS N'Номер комнаты', Telephone AS N'Телефон' FROM Responsible;", \_sqlConnection),  
  
 new SqlCommand(@"SELECT Id AS N'Уникальный номер записи', Responsible AS N'Табельный номер ответственного', Equipment AS N'Инвентарный ID оборудования', AS N'Дата получения', AS N'Дата возврата', PurposeOfUse AS N'Цель использования', ReasonForReturn AS N'Причина возврата' FROM Responsibility;", \_sqlConnection)  
 };  
  
 \_sqlConnection.Open();  
  
 try  
 {  
 *// Создания SQL адаптеров на основе ранее созданных объектов-команд* \_dataAdapterEquipment = new SqlDataAdapter(commands[0]);  
 \_dataAdapterResponsible = new SqlDataAdapter(commands[1]);  
 \_dataAdapterResponsibility = new SqlDataAdapter(commands[2]);  
  
 var dataAdapters = new[]  
 {  
 \_dataAdapterEquipment,   
 \_dataAdapterResponsible,   
 \_dataAdapterResponsibility  
 };  
 var dataTables = new[]  
 {  
 \_dataTableEquipment,   
 \_dataTableResponsible,  
 \_dataTableResponsibility  
 };  
 var dataGridViews = new[]  
 {  
 \_dataGridViewEquipment,   
 \_dataGridViewResponsible,   
 \_dataGridViewResponsibility  
 };  
  
  
 for (int i = 0; i < 3; i++)  
 {  
 dataTables[i].Reset();  
  
 *// Отпрака запроса* dataAdapters[i].Fill(dataTables[i]);  
  
 *// Связка DataGridView с DataTable* dataGridViews[i].DataSource = dataTables[i];  
 }  
 }  
 finally  
 {  
 \_sqlConnection.Close();  
 }  
  
 \_buttonSave.Click += SaveData;  
 }  
  
 */// <summary>  
 /// Обработчик события нажатия на кнопку «Сохранить»  
 /// </summary>* private void SaveData(object sender, EventArgs e)  
 {  
 *// Создание необходимых объектов, включая объекы SqlCommandBuilder, выполняющий автоматическое создание однотабличных команд* var dataAdapters = new[]  
 {  
 \_dataAdapterEquipment,   
 \_dataAdapterResponsible,  
 \_dataAdapterResponsibility  
 };  
 var dataTables = new[]  
 {  
 \_dataTableEquipment,  
 \_dataTableResponsible,  
 \_dataTableResponsibility  
 };  
 var commandsBuilders = new[]  
 {  
 new SqlCommandBuilder(\_dataAdapterEquipment),  
 new SqlCommandBuilder(\_dataAdapterResponsible),  
 new SqlCommandBuilder(\_dataAdapterResponsibility)  
 };  
  
 for (int i = 0; i < 3; i++)  
 {  
 \_sqlConnection.Open();  
  
 try  
 {  
 dataAdapters[i].UpdateCommand = commandsBuilders[i].GetUpdateCommand();  
 dataAdapters[i].InsertCommand = commandsBuilders[i].GetInsertCommand();  
 dataAdapters[i].DeleteCommand = commandsBuilders[i].GetDeleteCommand();  
  
 *// Отправка изменений* dataAdapters[i].Update(dataTables[i]);  
 }  
 finally  
 {  
 \_sqlConnection.Close();  
 }  
 }  
 }  
 }  
}

# 3 Подробное описание свойств и методов классов *DataTable*, *DbDataAdapter*, *DbCommand*, *DataGridView*, *DbCommandBuilder*

Описание свойств класса ***DataTable***:

* **Columns:** возвращает коллекцию столбцов, принадлежащих данной таблице. Возвращает объект класса ***DataColumnCollection***.
* **Rows:** возвращает коллекцию строк, принадлежащих данной таблице. Возвращает объект класса ***DataRowCollection***.
* **TableName:** возвращает или задаёт имя таблицы для объекта ***DataTable***. Возвращает и принимает строку типа ***string***.
* **DataSet:** возвращает класс ***DataSet***, к которому принадлежит данная таблица.
* **IsInitialized:** получает значение, указывающее, инициализирована ли таблица ***DataTable***.
* **Locale:** возвращает или задаёт сведения о языке, используемые для сравнения строк таблицы.
* **PrimaryKey:** возвращает или задаёт массив столбцов, которые являются столбцами первичного ключа для таблицы данных.

Описание методов класса ***DataTable***:

* **Reset():** возвращает таблицу в исходное состояние. Метод ничего не возвращает.
* **Cleat():** очищает таблицу от данных. Метод ничего не возвращает.
* **AcceptChanges():** фиксирует изменения, внесённые в таблицу.
* **Clone():** копирует структуру таблицы, включая все схемы и ограничения. Возвращает объект класса ***DataTable***.
* **Copy():** копирует структуру таблицы и данные. Возвращает объект типа ***DataTable***.
* **GetChanges():** возвращает объект класса ***DataTable***, который содержит все изменения, внесённые после последней фиксации изменений.
* **GetErrors():** возвращает массив объектов ***DataRow***, которые представляют собой записи с ошибками.
* **ImportRow(DataRow):** импортирует строки в таблицы, сохраняя все параметры свойств, а также текущие и исходные значения. Метод ничего не возвращает.
* **Load(IDataReader, LoadOption):** заполняет таблицу значениями из источника данных. Если объект ***DataTable*** уже содержит строки, поступающие данные из источника данных объединяются с существующими строками согласно значению параметра ***LoadOption***. Метод ничего не возвращает.
* **LoadDataRow(Object[], LoadOption):** находит и обновляет конкретную строку. Если нужная строка не найдена, то с помощью заданных значений создаётся новая строка. Возвращает объект ***DataRow***.
* **Merge(DataTable):** выполняет объединение таблиц. Метод ничего не возвращает.
* **NewRow():** создаёт новый объект класса ***DataRow***, который имеет такую же структуру, как и таблица. Возвращает объект класса ***DataRow***.
* **Select():** получает массив всех строк таблицы. Возвращает массив класса ***DataRow***.

Описание свойств класса ***DdDataAdapter***:

* **UpdateCommand:** команда используемая для обновления записей в источнике данных. Возвращает и принимает объект класса ***SqlCommand***.
* **SelectCommand:** команда используемая для выборки записей в источнике данных. Возвращает и принимает объект класса ***SqlCommand***.
* **InsertCommand:** команда используемая для вставки новых записей в источник данных. Возвращает и принимает объект класса ***SqlCommand***.
* **DeleteCommand:** команда используемая для удаления записей из источника данных. Возвращает и принимает объект класса ***SqlCommand***.

Описание методов класса ***DdDataAdapter***:

* **Update(DataTable):** обновляет значения в базе данных, выполняя соответствующие операторы **INSERT**, **UPDATE** или **DELETE** для каждой вставленной, обновлённой или удалённой строки в указанном ***DataTable***. Возвращает ***int*** равное количеству строк, успешно обновлённых.
* **TerminateBatching():** завершает пакетную обработку.
* **InitializeBatching():** инициализирует пакетную обработку.
* **Fill(DataTable):** добавляет или обновляет строки в указанном диапазоне в объект ***DataSet*** для получения соответствия строкам в источнике данных с использованием имени ***DataTable***. Возвращает ***int*** равное количеству строк, успешно обновлённых.
* **FillSchema(DataTable, SchemaType):** настраивает схему указанного объекта ***DataTable*** на основе указанного ***SchemaType***. Возвращает объект класса ***DataTable***.
* **GetFillParameters():** получает параметры, заданные пользователем при выполнении оператора SQL **SELECT**. Возвращает массив объектов класса ***IDataParameter***.

Описание свойств класса ***DataGridView***:

* **DataSource:** источник данных, для которого выполняется отображение данных. Возвращает и принимает ***object***.
* **DataBindings:** привязка данных к объекту. Возвращает коллекцию класса ***ControlBindingsCollection***.
* **DataMember:** имя таблицы или списка, для которого ***DataGridView*** выполняет отображение данных. Возвращает и принимает строку типа ***string***.
* **NewRowIndex:** индекс строки для новых записей. Возвращает значение типа ***int***.
* **ColumnCount:** количество столбцов, отображаемых в ***DataGridView***. Возвращает и принимает значение типа ***int***.
* **RowCount:** количество строк, отображаемых в ***DataGridView***. Возвращает и принимает значение типа ***int***.
* **Columns:** коллекция, содержащая все столбцы. Возвращает объект класса ***DataGridViewColumnCollection***.
* **Rows:** коллекция, содержащая все строки. Возвращает объект класса ***DataGridViewRowCollection***.
* **SelectedCells:** коллекция ячеек, выделенных пользователем. Возвращает объект класса ***DataGridViewSelectedCellCollection***.
* **CurrentCell:** активная ячейка. Возвращает и принимает объект класса ***DataGridViewCell***.
* **RowTemplate:** шаблон строки. возвращает объект класса ***DataGridViewRow***.
* **SortOrder:** значение, указывающие порядок сортировки. Возвращает значение перечисления ***SortOrder***.
* **Size:** высота и ширина элемента. Возвращает и принимает объект класса ***Size***.
* **Location:** координаты левого верхнего угла элемента. Возвращает и принимает объект типа ***Point***.
* **AutoSizeColumnsMode:** значение, указывающие как определяется ширина столбца. Возвращает и принимает перечисление ***DataGridViewAutoSizeColumnsMode***.
* **ColumnHeadersHeightSizeMode:** значение, указывающие может ли настраиваться высота заголовков пользователем или она должна автоматически настраиваться по содержимому. Возвращает и принимает перечисление ***DataGridViewColumnHeadersHeightSizeMode***.

Описание методов класса ***DataGridView***:

* **AreAllCellsSelected(Boolean):** возвращает ***bool*** значение, указывающие выбраны ли все сейчас ячейки. Передаваемый параметр указывает, следует ли учитывать скрытые ячейки.
* **AutoResizeColumns():** корректирует ширину столбцов по содержимому всех ячеек. Метод ничего не возвращает.
* **AutoResizeRows():** корректирует высоту строк по содержимому всех ячеек. Метод ничего не возвращает.
* **BeginEdit(Boolean):** переводит текущую ячейку в режим редактирования. Передаваемый параметр указывает, следует ли выделять всё содержимое ячейки. Возвращает ***bool***, указывающий находится ли ячейка уже в режиме редактирования.
* **CancelEdit():** отменяет режим редактирования для текущей ячейки и удаляет все изменения. Возвращает ***bool***, указывающий успешность отмены редактирования.
* **ClearSelection():** отменяет выделение всех выбранных ячеек. Метод ничего не возвращает.
* **EndEdit():** фиксирует и завершает редактирование ячейки. Возвращает ***bool***, указывающий успешность фиксации изменений.
* **SelectAll():** выбирает все ячейки. Метод ничего не возвращает.

Описание свойств класса ***DdCommandBuilder***:

* **DataAdapter:** ***DbDataAdapter***, для которого автоматически создаются инструкции T-SQL. Возвращает и принимает объект класса ***SqlDataAdapter***.
* **QuotePrefix:** начальный символ или символы, используемые для указания объекта базы данных, имена которых содержат пробел. Возвращает и принимает ***string***.
* **QuoteSuffix:** конечный символ или символы, используемые для указания объекта базы данных, имена которых содержат пробел. Возвращает и принимает ***string***.
* **SchemaSeparator:** символ, который используется в качестве разделится между идентификатором схемы и остальными идентификаторами. Возвращает и принимает ***string***.
* **SetAllValues:** указывает, следует ли включать все столбцы в инструкцию **UPDATE** или только те, чьи значения были изменены. Возвращает и принимает ***bool***.

Описание методов класса ***DdCommandBuilder***:

* **GetDeleteCommand():** генерирует команду для выполнения операций удаления в источнике данных. Возвращает объект класса ***SqlCommand***.
* **GetInsertCommand():** генерирует команду для выполнения операций вставки записей в источник данных. Возвращает объект класса ***SqlCommand***.
* **GetUpdateCommand():** генерирует команду для выполнения операций обновления записей в источнике данных. Возвращает объект класса ***SqlCommand***.
* **RefreshSchema():** очищает команды связанные с этим объектом. Метод ничего не возвращает.