**Практическая работа № 7. Реализация приложения базы данных на базе классов SQLConnection, SQLCommand, технологии WPF.**

**Программное обеспечение: Windows 8 и выше, Microsoft Office Word 2016 и выше, Microsoft SQL Server 2017 и выше, Microsoft SQL Server Management Studio 2017 и выше, Microsoft Visual Studio 2019 Community и выше.**

**Цель работы:**

**На основании разработанной и заполненной структуры базы данных, реализовать простейшее настольное приложение базы данных, к одной из подсистем индивидуальной предметной области.**

**Время на выполнение практической работы:** 4 часа.

**Получение навыков:**

* **В работе с технологией доступа к данным ADO.Net;**
* **В работе с пространствами имён Data и SqlClient;**
* **В работе с классами SqlConnection, SqlCommand, SqlDependency;**
* **В реализации методов вывода данных из таблиц базы данных в визуальные компоненты;**
* **В реализации методов по добавлению, изменению, удалению данных.**

**Теоретические сведения к выполнению практической работы:**

**Приложение базы данных** - программа или комплекс программ, направленный на работу с базой данных многих пользователей.

**Приложение базы данных** – это графический интерфейс работы с базой данных.

**Интерфейс** – графическая оболочка для общения конечного пользователя с аппаратной частью персонального компьютера и прочих устройств.

* 1 – о приложение может выполнять полный набор задач, а может быть разделено на несколько отдельных приложений.
* Каждое из приложений является рабочим местом для каждого из пользователей.

**Технология доступа к данным** – набор инструментов (библиотек, пространств имён, классов), которые позволяют инструментальной среде разработки приложения (в частности языку программирования), реализовывать механизмы по манипулированию и ведению информации в источниках данных.

ADO.Net (**A**ctiveX **D**ata **O**bject для . **NET**) с официального сайта Microsoft - предоставляет согласованный доступ к таким источникам данных, как SQL Server и XML, а также к источникам данных, предоставляемым при помощи OLE DB и ODBC. Пользовательские приложения, использующие общие данные, могут использовать ADO.NET для соединения с этими источниками данных и для получения, обработки и обновления имеющихся в них данных.

**System.Data.SqlClient** – пространство имён предоставляет классы, методы для организации, функций по ведению и манипулированию структурой и информации в источнике данных;

**System.Data.Sql** – пространство имён предоставляет, классы для поддержки функционирования SQL Server.

**SQLClient**:

1. **SqlConnection** – класс для организации подключения к источнику данных;
2. **SqlCommand** – класс для организации манипулирования над структурой и данными, к конкретному подключению;
3. **SqlDependency** – класс для организации прослушивания SQL Server.

**SqlConnection.**

*Перегрузки:*

***SqlConnection <New\_Name> = new SqlConnection();***

***SqlConnection <New\_Name> = new SqlConnection(“<Connection String>”);***

*Строка подключения:*

***“Data Source = <Source>; Initial Catalog = <Database Name>; <Authorization Method>;”***

**Source** – источник данных, может быть указан как, динамический IP адрес: MyPCName\NameSQLServer, или как статический IP: 192.168.0.1;

**Database Name** – название объекта базы данных на сервере;

**Authorization Method** – определяет принцип аутентификации в SQL Server:

* Для приложений БД рекомендуется, авторизация со стороны Windows (***Integrated Security = True;***);
* Если необходимо выполнить проверку подлинности со стороны SQL Server (***Persist Security Info = True; User = User\_Name; Password = ‘Pa$$w0rd’***).

**SqlConnection.**

*Примеры создания класса в двух режимах проверки подлинности:*

Проверка подлинности Windows - ***SqlConnection con = new SqlConnection(“Data Source = MyPC\MySQLServer; Initial catalog = MyDataBase; Integrated Security = true;”);***

Проверка подлинности SQL Server - ***SqlConnection(“Data Source = MyPC\MySQLServer; Initial catalog = MyDataBase; Persist Security Info = True; User = sa; Password = ‘Pa$$w0rd’;”);***

**SqlConnection.**

*Свойства:*

***ConnectionString*** – *get & set* свойство (тип: string), для редактирования строки подключения к источнику данных;

***State*** – *enum get* свойство, которое возвращает одно из перечислений, указывающее на состояние подключения, после последней операции, из коллекции ***ConnectionState***: ***Open, Closed, Broken, Connecting, Executing***.

*Методы:*

***Open();*** - метод открытия подключения (работы) с базой данных.

**Важно!** Для выполнения всех запросов отправляемых в БД, через SqlCommand и SqlDataAdapter, экземпляр класса дожен быть открыт;

***Close();*** - метод закрытия подключения (завершения работы) с базой данных.

**Важно!** Для последующего открытия подключения, все предыдущие подключения должны быть закрыты.

*Статические методы:*

***ClearAllPools();*** - закрывает все открытие подключения доступному экземпляру;

***ChangePassword(“Connection String”,”New Password”);*** – меняет пароль у экземпляра подключения.

**SqlCommand.**

*Перегрузки:*

***SqlCommand <New\_Name> = new SqlCommand();***

***SqlCommand <New\_Name> = new SqlCommand(“SQL Query String”);***

***SqlCommand <New\_Name> = new SqlCommand(“SQL Query String”, <SqlConnection>);***

**SQL Query String** – запрос SQL любой группы команд;

**SqlConnection** – экземпляр класса.

**SqlCommand.**

*Пример объявления класса:*

***SqlConnection con = new SqlConnection (“Data Source = MyPC\MySQLServer; Initial catalog = MyDataBase; Integrated Security = true;”);***

***SqlCommand cmd = new SqlCommand(“select \* from [dbo].[Table]”, con);***

**SqlCommand.**

*Свойства:*

***Connection*** – get & set свойство (тип: SqlConnection), настройки у экземпляра класса, класса подключения к источнику данных;

***CommandText*** – get & set свойство (тип: string), указания SQL запроса любой группы команды;

***CommandType*** – enum get & set свойство, определяет тип SqlCommand, получает значение из коллекции ***CommandType***, где ***Text*** – работа с SQL запросами, ***StoredProcedure*** – работа с хранимыми процедурами;

***Parameters*** – get свойство (тип: SqlParameterCollection), возвращает коллекцию параметров для ввода данных в запросы, имеет собственные методы:

* ***Add(“Name”);*** - добавление нового параметра в конец списка;
* ***AddWithValue(“Name”, <Value>);*** - добавление нового параметра в конец списка, с присвоением значения;
* ***Clear();*** - отчистка коллекции параметров.

***Notification*** – get & set свойство, получает значение отклика от сервера сопутствующей команды, при значении Null уведомления не запрашиваются.

*Методы:*

***ExecuteNonQuery();*** – выполняет указанный SQL запрос. Ничего не возвращает;

***ExecuterReader();*** – выполняет указанный SQL запрос с ожидаем возврата табличного вида в виртуальном представлении;

***ExcuteScalar();*** – выполняет указанный SQL запрос с ожидаем возврата одной строки и одного столбца.

**SqlDependency.**

*Перегрузки:*

***SqlDependecy <New\_Name> = new SqlDependency();***

*Пример объявления:*

***SqlDependecy depend = new SqlDependency();***

*События:*

***OnChange*** – возникает в тот период времени, когда на сервере в таблице, происходит любое манипулирование над данными;

*Методы:*

***AddCommandDependency(SqlCommand);*** - добавление экземпляра класса SqlCommand;

*Статические методы:*

***Start(SqlConnection.ConnectionString);*** - метод запуска прослушивания событий на сервере;

***Stop(SqlConnection.ConnectionString);*** - метод остановки прослушивания событий на сервере.

**Описание визуальных компонентов и классов:**

**Window** – визуальный компонент, выполняет роль контейнера, на котором располагаются прочие элементы управления.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Title = “” | Свойство определяет заголовок окна |
| Name = “” | Свойство опредедят имя объекта |
| Height = “” | Свойство определяет высоту объекта |
| Width = “” | Свойство определяет ширину объекта |
| WindowState = “” | Свойство определяет развёртуемость окна |
| WindowStartupLocation = “” | Свойство определяет стартовое положение окна |
| Visibility | Свойство определяет видимость объекта |
| Distpatcher | Свойство хранит коллекцию элементов управления контейнеа |
| IsActive | Свойство определяет доступность элемента в момент работы ПП |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| Show(); | Метод вызова объекта |
| Invoke(); | Метод подстановки делегата в качестве формального параметра, в момент обращения фонового потока, к элементу управления в основной поток |
| Close(); | Метод закрытия объекта |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| Closing | Событие в момент перед закрытием объекта |

**Grid** – визуальный компонент, который формирует сетку из строк и столбцов в окне, в ячейках которого можно располагать прочие элементы управления.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Grid.ColumnDefinitions | Свойство определяет коллекцию столбцов в сетке |
| ColumnDefinition | Свойство определяет один столбец в коллекции столбцов |
| ColumnDefinition Width = “” | Свойство определяет ширину объекта |
| Grid.RowDefinition | Свойство определяет коллекцию строк в сетке |
| RowDefinition | Свойство определяет одну строку в коллекции столбцов |
| RowDefinition Height = “” | Свойство опрелеят высоту объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.RowSpan = “” | Свойство определяет расположение объекта на определённое количество строк |
| Visibility = “” | Свойство определяет видимость объекта |

**Label** – визуальный компонент, представялет собой надпись.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Content = “” | Свойство определяет визуальное отображение текста на объекте |
| HorizontalAligment = “” | Свойство определяет горизонталное положение текста в объекте |
| VerticalAligment = “” | Свойство определяет вертикальное положение текста в объекте |
| FontSize = “” | Свойство определяет размер текста в объекта |
| Grid.ColumnSpan = “” | Свойство определяет расположение объекта на определённое количество столбцов |

**PasswordBox** – визуальный компонент, представляет собой поле ввода, которое скрывает вводимые символы.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| VerticalAligment = “” | Свойство определяет вертикальное положение текста в объекте |
| FontSize = “” | Свойство определяет размер текста в объекта |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| Focus(); | Метод переводит курсор и фокус на объект |
| Clear(); | Метод отчищает содержимое объекта от строковых значений |

**Button** – визуальный компонент, представляет собой кнопку с надписью.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| FontSize = “” | Свойство определяет размер текста в объекта |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| Click | Событие возникает в момент, когда пользователь нажимает на объект |

**DataGrid** – ваизуальный компонент, представляет собой табличное представление информации.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Grid.ColumnSpan = “” | Свойство определяет расположение объекта на определённое количество столбцов |
| CanUserAddRows = “” | Свойство определяет, может ли пользователь добавлять строки через объект |
| CanUserDeleteRows = “” | Свойство определяет, может ли пользователь удалять строки через объект |
| ItemsSource | Свойство определяет источник данных из Кэш таблицы |
| Columns[].Visibility | Свойство определяет видимость столбца |
| Columns[].Header | Свойство определяет надпись в заголовке столбца |
| Items.Count | Свойство возвращает количество элементов объекте |
| SelectedItems.Count | Свойство возвращает количество выбранных элементов в объекте |
| SelectedItems[] | Свойство определяет по индексу выбранный элемент в объекте |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| SelectionChanged | Событие возникает в тот период времени, когда пользователь изменяет выбор ячейки в таблице |

**TextBox** – визуальный компонет, представляет собой стандартное поле ввода.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Grid.ColumnSpan = “” | Свойство определяет расположение объекта на определённое количество столбцов |
| Text | Свойство определяет визуальное отображение текста в объекте |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| Clear(); | Метод отчищает содержимое объекта от строковых значений |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| TextChanged | Метод воникает в тот момент времени, когда меняется содержимое объекта |

**MaskedTextBox** – визуальный элемент управления (NuGet компонент, из пространства имён Extended.WPF.Toolkit), предсталяет собой поле ввода с маской ввода данных.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Grid.ColumnSpan = “” | Свойство определяет расположение объекта на определённое количество столбцов |
| Mask = “” | Свойство определяет маску ввода в объекте |
| Text | Свойство определяет визуальное отображение текста в объекте |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| ToString(); | Метод переводит результат предыдущей команды в строку |
| Substring(); | Метод выводит подстроку из строки, где 1 формальный параметр, стартовая позиция подстроки (индекс), второй формальный парметр, длинна подстроки |
| Clear(); | Метод отчищает содержимое объекта от строковых значений |

**ComboBox** – визуальный компонент, представляет собой выпадающий список данных.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Grid.ColumnSpan = “” | Свойство определяет расположение объекта на определённое количество столбцов |
| ItemsSource | Свойство определяет источник данных из Кэш таблицы |
| SelectedValuePath | Свойство определяет значение первичного или суррогатного ключа с указанием названием соотвествующего столбца из таблицы БД |
| DisplayMemberPath | Свойство определяет значение неключевого столбца, который пользователь будет видеть на экране с указанием названия соответствующего столбца из таблицы БД |
| Items.Count | Свойство возвращает количество элементов объекте |
| SelectedValue | Свойство определяет значение первичного или суррогатного ключа |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| SelectionChanged | Событие возникает в тот период времени, когда пользователь изменяет выбор элемента из списка |

**ListBox** – визуальный компонент, представляет собой список данных.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Name = “” | Свойство опредедяет имя объекта |
| Grid.Row = “” | Свойство определяет номер строки, позиционирования объекта, в родительской сетке |
| Grid.Column = “” | Свойство определяет номер столбца, позиционирования объекта, в родительской сетке |
| Grid.ColumnSpan = “” | Свойство определяет расположение объекта на определённое количество столбцов |
| ItemsSource | Свойство определяет источник данных из Кэш таблицы |
| SelectedValuePath | Свойство определяет значение первичного или суррогатного ключа с указанием названием соотвествующего столбца из таблицы БД |
| DisplayMemberPath | Свойство определяет значение неключевого столбца, который пользователь будет видеть на экране с указанием названия соответствующего столбца из таблицы БД |
| Items.Count | Свойство возвращает количество элементов объекте |
| SelectedItems.Count | Свойство возвращает количество выбранных элементов в объекте |
| SelectedItems[] | Свойство определяет по индексу выбранный элемент в объекте |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| SelectionChanged | Событие возникает в тот период времени, когда пользователь изменяет выбор элемента из списка |

**Описание не визуальных компонентов:**

**SqlConnection** – класс определяет и конфигурирует подключение к источнику данных.

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| Open(); | Метод открытия соединения с сервером и базой данных |
| Close(); | Метод закрытия соединения с сервером и базой данных |

**SqlCommand** – класс определяет и конфигурирует SQL запросы к источнику данных.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Connection | Свойство определяет экземпляр класса SqlConnection |
| CommandText | Свойство определяет текст SQL запроса, который необходимо передать на сервер в базу данных |
| Notification | Свойство определяет сообщение о событиях относительно сервера |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| ExecuteReader(); | Метод возвращает табличное представление результата запроса на выборку данных |
| ExecuteNonQuery(); | Метод отпраялет на выполнение любой запрос базы данных, на сервер |

**DataTable** – класс определяет и конфигурирует, виртуальную КЭШ-таблицу в оперативной памяти ПК.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| DefaultView | Свойство определяет стандартное представление данных, применяется для технологии WPF |
| Columns[].ColumnName | Свойство определяет по индексу столбца, его название |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| Load(); | Метод возвращает в объект загруженное табличное представление информации |

**SqlDependecy** – класс определяет и конфигурирует зависимость между приложением БД и таблицами сервера БД.

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| AddCommandDependencty(); | Метод добавляет к объекту команду запроса для будущей зависимости |
| Star(); | Статический метод запускает процесс зависимости класса от строки подключения к источнику данных |

События:

|  |  |
| --- | --- |
| Команда | Описание |
| OnChange | Событие возникает в тот период времени, когда на сервере в таблице базы данных, произошли манипуляции над данными |

**Action** – делегат, который представляет собой, некоторое действие, которое ничего не возвращает.

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| Action action = () => { }; | Метод объявления делегата, где в круглых скобках, можно указать формальные парметры, для передачи значений в тело делгата, между фигурными скобками указывается тело делегата |

**SqlNotificationEventArgs** – класс определяет и возвращает события полученные из таблице на сервере БД.

Свойства:

|  |  |
| --- | --- |
| Команда | Описание |
| Info | Свойство возвращает событие с сервера базы данных, о состоянии зависимой таблицы |

**DataRowView** – класс определяет и конфигурирует обращение к представлению виртуальной КЭШ-строки.

Свойства:

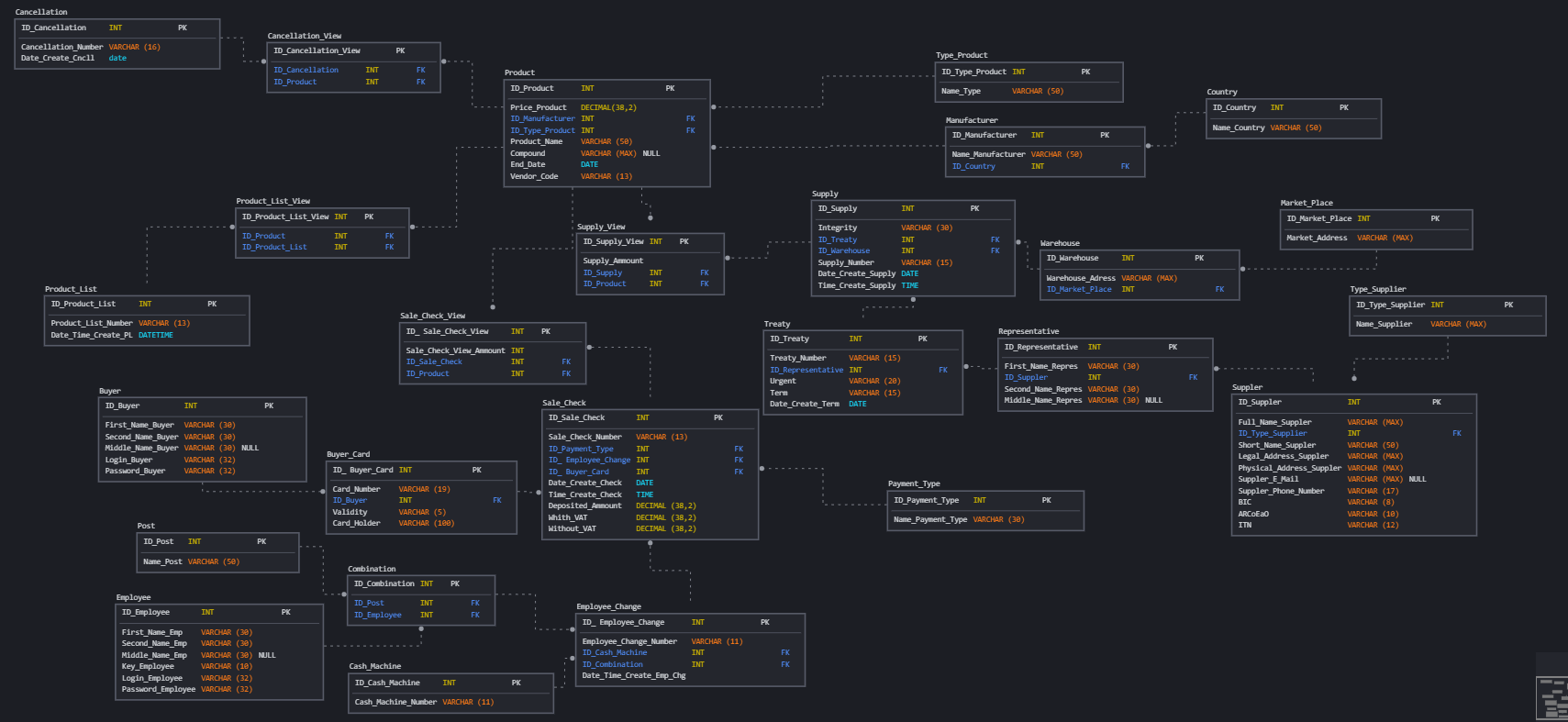
|  |  |
| --- | --- |
| Команда | Описание |
| <Экземпляр класса>[] | Свойство возвращает по индексу значение выбранной строки |

Методы:

|  |  |
| --- | --- |
| Команда | Описание |
| ToString(); | Метод переводит результат предыдущей команды в строку |

**Этапы выполнения:**

1. На даталогической модели данных отметить те таблицы, которые относятся к одной из подсистем и не относятся к операционным таблицам (Приложение пишется для администратора системы, в задачу которого входит актуализация данных в справочных таблицах);



1. В инструментальной среде реализовать универсальный класс для работы с запросами к базе данных;

using System.Data;

using System.Data.SqlClient;

namespace AdministratorApp

{

class DataBaseClass

{

public static string Users\_ID = "null", Password = "null", App\_Name = "Администратор - Продажа товара";

public static string ConnectionStrig = "Data Source = IDAPEN\\IDAPEN\_SERVER; Initial Catalog = Sale\_Product\_Database; Persist Security Info = true; User ID = {0}; Password = '{1}';";

public SqlConnection connection = new SqlConnection(ConnectionStrig);

private SqlCommand command = new SqlCommand();

public DataTable resultTable = new DataTable();

public SqlDependency dependency = new SqlDependency();

public enum act { select, manipulation};

public void sqlExecute(string quety, act act)

{

command.Connection = connection;

command.CommandText = quety;

command.Notification = null;

switch (act)

{

case act.select:

dependency.AddCommandDependency(command);

SqlDependency.Start(connection.ConnectionString);

connection.Open();

resultTable.Load(command.ExecuteReader());

connection.Close();

break;

case act.manipulation:

connection.Open();

command.ExecuteNonQuery();

connection.Close();

break;

}

}

}

}

1. Реализовать интерфейс для справочных таблиц;

|  |
| --- |
| Код разметки XAML – Администратор |
| <Window x:Class="AdministratorApp.SimpleTablesWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:ms ="http://schemas.xceed.com/wpf/xaml/toolkit"  xmlns:local="clr-namespace:AdministratorApp"  mc:Ignorable="d"  Title="Администратор - Актуализация справочников" Height="650" Name="SimpleTables" Width="1000" WindowStartupLocation="CenterScreen" WindowState="Maximized" Closing="SimpleTables\_Closing">  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="0.3\*"></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  </Grid.RowDefinitions>  <Button Name="btSimpleTable" Grid.Column="0" Grid.Row="0" Content="Справочники" Click="btSimpleTable\_Click"></Button>  <Button Name="btSecondaryTable" Grid.Column="0" Grid.Row="1" Content="Сопастовление" Click="btSecondaryTable\_Click"></Button>  <Button Name="btSupplierInfo" Grid.Column="0" Grid.Row="2" Content="Поставщики и представители" Click="btSupplierInfo\_Click"></Button>  <Button Name="btClose" Grid.Row="3" Grid.Column="0" Content="Закрыть" Click="btClose\_Click"></Button>  <Grid Name="grdOneFielTable" Grid.Column="1" Grid.Row="0" Grid.RowSpan="4" Visibility="Visible">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition></RowDefinition>  </Grid.RowDefinitions>  <Grid Grid.Row="0" Grid.Column="0">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  </Grid.RowDefinitions>  <ListBox Name="lstbxPost" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="lstbxPost\_SelectionChanged"></ListBox>  <TextBox Name="tbPostName" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Text="Название должности" VerticalContentAlignment="Center"/>  <Button Name="btInsertPost" Grid.Row="2" Grid.Column="0" Content="Новая запись" Click="btInsertPost\_Click"></Button>  <Button Name="btUpdatePost" Grid.Row="2" Grid.Column="1" Content="Изменить" Click="btUpdatePost\_Click"></Button>  <Button Name="btDeletePost" Grid.Row="2" Grid.Column="2" Content="Удалить запись" Click="btDeletePost\_Click"></Button>  </Grid>  <Grid Grid.Row="0" Grid.Column="1">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  </Grid.RowDefinitions>  <ListBox Name="lstbxCashMachine" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="lstbxCashMachine\_SelectionChanged"></ListBox>  <ms:MaskedTextBox Name="mktbCashMachineNumber" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Mask="КА-99999999" VerticalContentAlignment="Center"></ms:MaskedTextBox>  <Button Name="btInsertCashMachine" Grid.Row="2" Grid.Column="0" Content="Новая запись" Click="btInsertCashMachine\_Click"></Button>  <Button Name="btUpdateCashMachine" Grid.Row="2" Grid.Column="1" Content="Изменить" Click="btUpdateCashMachine\_Click"></Button>  <Button Name="btDeleteCashMachine" Grid.Row="2" Grid.Column="2" Content="Удалить запись" Click="btDeleteCashMachine\_Click"></Button>  </Grid>  <Grid Grid.Row="0" Grid.Column="2">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  </Grid.RowDefinitions>  <ListBox Name="lstbxTypeProduct" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="lstbxTypeProduct\_SelectionChanged"></ListBox>  <TextBox Name="tbTypProductName" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Text="Название типа продукта" VerticalContentAlignment="Center"></TextBox>  <Button Name="btInsertTypeProduct" Grid.Row="2" Grid.Column="0" Content="Новая запись" Click="btInsertTypeProduct\_Click"></Button>  <Button Name="btUpdateTypeProduct" Grid.Row="2" Grid.Column="1" Content="Изменить" Click="btUpdateTypeProduct\_Click"></Button>  <Button Name="btDeleteTypeProduct" Grid.Row="2" Grid.Column="2" Content="Удалить запись" Click="btDeleteTypeProduct\_Click"></Button>  </Grid>  <Grid Grid.Row="1" Grid.Column="0">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  </Grid.RowDefinitions>  <ListBox Name="lstbxCountry" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="lstbxCountry\_SelectionChanged"></ListBox>  <TextBox Name="tbCountryName" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Text="Название страны" VerticalContentAlignment="Center"></TextBox>  <Button Name="btInsertCountry" Grid.Row="2" Grid.Column="0" Content="Новая запись" Click="btInsertCountry\_Click"></Button>  <Button Name="btUpdateCountry" Grid.Row="2" Grid.Column="1" Content="Изменить" Click="btUpdateCountry\_Click"></Button>  <Button Name="btDeleteCountry" Grid.Row="2" Grid.Column="2" Content="Удалить запись" Click="btDeleteCountry\_Click"></Button>  </Grid>  <Grid Grid.Row="1" Grid.Column="1">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  </Grid.RowDefinitions>  <ListBox Name="lstbxMarketPlace" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="lstbxMarketPlace\_SelectionChanged"></ListBox>  <TextBox Name="tbMarketAdress" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Text="Адрес магазина" VerticalContentAlignment="Center"></TextBox>  <Button Name="btInsertMarketPlace" Grid.Row="2" Grid.Column="0" Content="Новая запись" Click="btInsertMarketPlace\_Click"></Button>  <Button Name="btUpdateMarketPlace" Grid.Row="2" Grid.Column="1" Content="Изменить" Click="btUpdateMarketPlace\_Click"></Button>  <Button Name="btDeleteMarketPlace" Grid.Row="2" Grid.Column="2" Content="Удалить запись" Click="btDeleteMarketPlace\_Click"></Button>  </Grid>  <Grid Grid.Row="1" Grid.Column="2">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  <RowDefinition Height="0.2\*"></RowDefinition>  </Grid.RowDefinitions>  <ListBox Name="lstbxTypeSupplier" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="lstbxTypeSupplier\_SelectionChanged"></ListBox>  <TextBox Name="tbTypeSupplierName" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Text="Название типа поставщика" VerticalContentAlignment="Center"></TextBox>  <Button Name="btInsertTypeSupplier" Grid.Row="2" Grid.Column="0" Content="Новая запись" Click="btInsertTypeSupplier\_Click"></Button>  <Button Name="btUpdateTypeSupplier" Grid.Row="2" Grid.Column="1" Content="Изменить" Click="btUpdateTypeSupplier\_Click"></Button>  <Button Name="btDeleteTypeSupplier" Grid.Row="2" Grid.Column="2" Content="Удалить запись" Click="btDeleteTypeSupplier\_Click"></Button>  </Grid>  </Grid>  <Grid Name="grdSecondaryTable" Grid.Column="1" Grid.Row="0" Grid.RowSpan="4" Visibility="Hidden">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid Grid.Column="0">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.06\*"></RowDefinition>  </Grid.RowDefinitions>  <DataGrid Name="dgManufacture" Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="3" CanUserAddRows="False" CanUserDeleteRows="False" SelectionChanged="dgManufacture\_SelectionChanged"></DataGrid>  <Label Name="lblManufactureName" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Content="Название производителя:"></Label>  <TextBox Name="tbManufactureName" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="3"></TextBox>  <Label Name="lblCountrySelect" Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="3" Content="Название страны производителя:"></Label>  <ComboBox Name="cbCountrySelect" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="cbCountrySelect\_SelectionChanged"></ComboBox>  <Button Name="InsertManufacture" Grid.Row="5" Grid.Column="0" Content="Новая запись" Click="InsertManufacture\_Click"></Button>  <Button Name="UpdateManufacture" Grid.Row="5" Grid.Column="1" Content="Сохранить изменения" Click="UpdateManufacture\_Click"></Button>  <Button Name="DeleteManufacture" Grid.Row="5" Grid.Column="2" Content="Удалить запись" Click="DeleteManufacture\_Click"></Button>  </Grid>  <Grid Grid.Column="1">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.06\*"></RowDefinition>  </Grid.RowDefinitions>  <DataGrid Name="dgWarehouse" Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="3" CanUserAddRows="False" CanUserDeleteRows="False" SelectionChanged="dgWarehouse\_SelectionChanged"></DataGrid>  <Label Name="lblWarehouseAdress" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Content="Адрес склада:"></Label>  <TextBox Name="tbWarehouseAdress" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="3"></TextBox>  <Label Name="lblMarketPlaceSelect" Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="3" Content="Адрес торговой точки:"></Label>  <ComboBox Name="cbMarketPlaceSelect" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="cbMarketPlaceSelect\_SelectionChanged"></ComboBox>  <Button Name="InsertWarehouse" Grid.Row="5" Grid.Column="0" Content="Новая запись" Click="InsertWarehouse\_Click"></Button>  <Button Name="UpdateWarehouse" Grid.Row="5" Grid.Column="1" Content="Сохранить изменения" Click="UpdateWarehouse\_Click"></Button>  <Button Name="DeleteWarehouse" Grid.Row="5" Grid.Column="2" Content="Удалить запись" Click="DeleteWarehouse\_Click"></Button>  </Grid>  </Grid>  <Grid Name="grdSupplierInfo" Grid.Column="1" Grid.Row="0" Grid.RowSpan="4" Visibility="Hidden">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition Width="0.5\*"></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid Grid.Column="0">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.08\*"></RowDefinition>  </Grid.RowDefinitions>  <DataGrid Name="dgSuppler" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" CanUserAddRows="False" CanUserDeleteRows="False" SelectionChanged="dgSuppler\_SelectionChanged"></DataGrid>  <Label Name="lblTypeSupplerSelect" Grid.Row="1" Grid.Column="0" Content="Вид поставщика:" VerticalAlignment="Center"></Label>  <ComboBox Name="cbTypeSupplerSelect" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2" SelectionChanged="cbTypeSupplerSelect\_SelectionChanged"></ComboBox>  <Label Name="lblFullNameSuppler" Grid.Row="2" Grid.Column="0" Content="Полное название:" VerticalAlignment="Center"></Label>  <TextBox Name="tbFullNameSuppler" Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="2" TextChanged="tbFullNameSuppler\_TextChanged"></TextBox>  <Label Name="lblSortNameSuppler" Grid.Row="3" Grid.Column="0" Content="Соркращённое название:" VerticalAlignment="Center"></Label>  <TextBox Name="tbSortNameSuppler" Grid.Row="3" Grid.Column="1" Grid.ColumnSpan="2"></TextBox>  <Label Name="lblLegalAddressSuppler" Grid.Row="4" Grid.Column="0" Content="Юридический адрес:" VerticalAlignment="Center"></Label>  <TextBox Name="tbLegalAddressSuppler" Grid.Row="4" Grid.Column="1" Grid.ColumnSpan="2" TextChanged="tbLegalAddressSuppler\_TextChanged"></TextBox>  <Label Name="lblPhysicalAddressSuppler" Grid.Row="5" Grid.Column="0" Content="Физический адрес:" VerticalAlignment="Center"></Label>  <TextBox Name="tbPhysicalAddressSuppler" Grid.Row="5" Grid.Column="1" Grid.ColumnSpan="2"></TextBox>  <Label Name="lblSupplerEMail" Grid.Row="6" Grid.Column="0" Content="Адрес эл. почты:" VerticalAlignment="Center"></Label>  <TextBox Name="tbSupplerEMail" Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2"></TextBox>  <Label Name="lblSupplerPhoneNumber" Grid.Row="7" Grid.Column="0" Content="Контактный телефон:" VerticalAlignment="Center"></Label>  <ms:MaskedTextBox Name="msktbSupplerPhoneNumbe" Grid.Row="7" Grid.Column="1" Grid.ColumnSpan="2" Mask="+7(999)999-99-99"></ms:MaskedTextBox>  <Label Name="lblBIC" Grid.Row="8" Grid.Column="0" Content="БИК:" VerticalAlignment="Center"></Label>  <ms:MaskedTextBox Name="msktbBIC" Grid.Row="8" Grid.Column="1" Grid.ColumnSpan="2" Mask="99999999"></ms:MaskedTextBox>  <Label Name="lblARCoEaO" Grid.Row="9" Grid.Column="0" Content="ОКПО:" VerticalAlignment="Center"></Label>  <ms:MaskedTextBox Name="msktbARCoEaO" Grid.Row="9" Grid.Column="1" Grid.ColumnSpan="2" Mask="9999999999"></ms:MaskedTextBox>  <Label Name="lblITN" Grid.Row="10" Grid.Column="0" Content="ИНН:" VerticalAlignment="Center"></Label>  <ms:MaskedTextBox Name="msktbITN" Grid.Row="10" Grid.Column="1" Grid.ColumnSpan="2" Mask="999999999999"></ms:MaskedTextBox>  <Button Name="btInsertSuppler" Grid.Row="11" Grid.Column="0" Content="Новая запись" Click="btInsertSuppler\_Click"></Button>  <Button Name="btUpdateSuppler" Grid.Row="11" Grid.Column="1" Content="Сохранить изменения" Click="btUpdateSuppler\_Click"></Button>  <Button Name="btDeleteSuppler" Grid.Row="11" Grid.Column="2" Content="Удалить запись" Click="btDeleteSuppler\_Click"></Button>  </Grid>  <Grid Grid.Column="1">  <Grid.ColumnDefinitions>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition Height="0.07\*"></RowDefinition>  </Grid.RowDefinitions>  <DataGrid Name="dgRepresentative" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" CanUserAddRows="False" CanUserDeleteRows="False" SelectionChanged="dgRepresentative\_SelectionChanged"></DataGrid>  <Label Name="lblSupplerSelect" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" Content="Фирма представителя"></Label>  <ComboBox Name="cbSupplerSelect" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="3" SelectionChanged="cbSupplerSelect\_SelectionChanged"></ComboBox>  <Label Name="lblFirstNameRepres" Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="3" Content="Фамилия представителя:"></Label>  <TextBox Name="tbFirstNameRepres" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="3"></TextBox>  <Label Name="lblSecondNameRepres" Grid.Row="5" Grid.Column="0" Grid.ColumnSpan="3" Content="Имя представителя:"></Label>  <TextBox Name="tbSecondNameRepres" Grid.Row="6" Grid.Column="0" Grid.ColumnSpan="3"></TextBox>  <Label Name="lblMiddleNameRepres" Grid.Row="7" Grid.Column="0" Grid.ColumnSpan="3" Content="Отчество представителя:"></Label>  <TextBox Name="tbMiddleNameRepres" Grid.Row="8" Grid.Column="0" Grid.ColumnSpan="3"></TextBox>  <Button Name="btInsertRepresentative" Grid.Row="9" Grid.Column="0" Content="Новая запись" Click="btInsertRepresentative\_Click"/>  <Button Name="btUpdateRepresentative" Grid.Row="9" Grid.Column="1" Content="Сохранить изменения" Click="btUpdateRepresentative\_Click"></Button>  <Button Name="btDeleteRepresentative" Grid.Row="9" Grid.Column="2" Content="Удалить запись" Click="btDeleteRepresentative\_Click"></Button>  </Grid>  </Grid>  </Grid>  </Window> |
| Результат |
| Вкладка «Справочники»    Вкладка «Сопоставление»    Вкладка «Поставщики и представление» |

1. Реализовать функции добавления, изменения и удаления данных;

|  |  |
| --- | --- |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void postFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Post], [Name\_Post] from [dbo].[Post]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_post;  lstbxPost.ItemsSource = dataBaseClass.resultTable.DefaultView;  lstbxPost.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  lstbxPost.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_post(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  postFill();  } |
| Добавление данных |
| private void btInsertPost\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Post] ([Name\_Post]) values ('{0}')", tbPostName.Text), DataBaseClass.act.manipulation);  tbPostName.Clear();  } |
| Изменение данных |
| private void btUpdatePost\_Click(object sender, RoutedEventArgs e)  {  if (lstbxPost.Items.Count != 0 & lstbxPost.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxPost.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("update [dbo].[Post] set [Name\_Post] = '{0}' where [ID\_Post] = {1}", tbPostName.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeletePost\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (lstbxPost.Items.Count != 0 & lstbxPost.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxPost.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Post] where [ID\_Post] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void lstbxPost\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if(lstbxPost.Items.Count != 0 & lstbxPost.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)lstbxPost.SelectedItems[0];  tbPostName.Text = dataRow[1].ToString();  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void cashMachineFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Cash\_Machine], [Cash\_Machine\_Number] from [dbo].[Cash\_Machine]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_cashMachine;  lstbxCashMachine.ItemsSource = dataBaseClass.resultTable.DefaultView;  lstbxCashMachine.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  lstbxCashMachine.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_cashMachine(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  cashMachineFill();  } |
| Добавление данных |
| private void btInsertCashMachine\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Cash\_Machine] ([Cash\_Machine\_Number]) values ('{0}')", mktbCashMachineNumber.Text), DataBaseClass.act.manipulation);  mktbCashMachineNumber.Clear();  } |
| Изменение данных |
| private void btUpdateCashMachine\_Click(object sender, RoutedEventArgs e)  {  if (lstbxCashMachine.Items.Count != 0 & lstbxCashMachine.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxCashMachine.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("update [dbo].[Cash\_Machine] set [Cash\_Machine\_Number] = '{0}' where [ID\_Cash\_Machine] = {1}", mktbCashMachineNumber.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteCashMachine\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (lstbxCashMachine.Items.Count != 0 & lstbxCashMachine.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxCashMachine.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Cash\_Machine] where [ID\_Cash\_Machine] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void lstbxCashMachine\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (lstbxCashMachine.Items.Count != 0 & lstbxCashMachine.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)lstbxCashMachine.SelectedItems[0];  mktbCashMachineNumber.Text = dataRow[1].ToString().Substring(3,8);  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void typeProductFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Type\_Product], [Name\_Type] from [dbo].[Type\_Product]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_typeProduct;  lstbxTypeProduct.ItemsSource = dataBaseClass.resultTable.DefaultView;  lstbxTypeProduct.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  lstbxTypeProduct.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_typeProduct(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  typeProductFill();  } |
| Добавление данных |
| private void btInsertTypeProduct\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Type\_Product] ([Name\_Type]) values ('{0}')", tbTypProductName.Text), DataBaseClass.act.manipulation);  tbTypProductName.Clear();  } |
| Изменение данных |
| private void btUpdateTypeProduct\_Click(object sender, RoutedEventArgs e)  {  if (lstbxTypeProduct.Items.Count != 0 & lstbxTypeProduct.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxTypeProduct.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("update [dbo].[Type\_Product] set [Name\_Type] = '{0}' where [ID\_Type\_Product] = {1}", tbTypProductName.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteTypeProduct\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (lstbxTypeProduct.Items.Count != 0 & lstbxTypeProduct.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxTypeProduct.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Type\_Product] where [ID\_Type\_Product] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void lstbxTypeProduct\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (lstbxTypeProduct.Items.Count != 0 & lstbxTypeProduct.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)lstbxTypeProduct.SelectedItems[0];  tbTypProductName.Text = dataRow[1].ToString();  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void countryFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Country], [Name\_Country] from [dbo].[Country]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_country;  lstbxCountry.ItemsSource = dataBaseClass.resultTable.DefaultView;  lstbxCountry.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  lstbxCountry.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_country(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  countryFill();  } |
| Добавление данных |
| private void btInsertCountry\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Country] ([Name\_Country]) values ('{0}')", tbCountryName.Text), DataBaseClass.act.manipulation);  tbCountryName.Clear();  } |
| Изменение данных |
| private void btUpdateCountry\_Click(object sender, RoutedEventArgs e)  {  if (lstbxCountry.Items.Count != 0 & lstbxCountry.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxCountry.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("update [dbo].[Country] set [Name\_Country] = '{0}' where [ID\_Country] = {1}", tbCountryName.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteCountry\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (lstbxCountry.Items.Count != 0 & lstbxCountry.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxCountry.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Country] where [ID\_Country] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void lstbxCountry\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (lstbxCountry.Items.Count != 0 & lstbxCountry.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)lstbxCountry.SelectedItems[0];  tbCountryName.Text = dataRow[1].ToString();  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void marketPlaceFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Market\_Place], [Market\_Address] from [dbo].[Market\_Place]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_marketPlace;  lstbxMarketPlace.ItemsSource = dataBaseClass.resultTable.DefaultView;  lstbxMarketPlace.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  lstbxMarketPlace.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_marketPlace(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  marketPlaceFill();  } |
| Добавление данных |
| private void btInsertMarketPlace\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Market\_Place] ([Market\_Address]) values ('{0}')", tbMarketAdress.Text), DataBaseClass.act.manipulation);  tbMarketAdress.Clear();  } |
| Изменение данных |
| private void btUpdateMarketPlace\_Click(object sender, RoutedEventArgs e)  {  if (lstbxMarketPlace.Items.Count != 0 & lstbxMarketPlace.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxMarketPlace.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("update [dbo].[Market\_Place] set [Market\_Address] = '{0}' where [ID\_Market\_Place] = {1}", tbMarketAdress.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteMarketPlace\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (lstbxMarketPlace.Items.Count != 0 & lstbxMarketPlace.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxMarketPlace.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Market\_Place] where [ID\_Market\_Place] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void lstbxMarketPlace\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (lstbxMarketPlace.Items.Count != 0 & lstbxMarketPlace.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)lstbxMarketPlace.SelectedItems[0];  tbMarketAdress.Text = dataRow[1].ToString();  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void typeSupplierFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Type\_Supplier], [Name\_Supplier] from [dbo].[Type\_Supplier]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_typeSupplier;  lstbxTypeSupplier.ItemsSource = dataBaseClass.resultTable.DefaultView;  lstbxTypeSupplier.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  lstbxTypeSupplier.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_typeSupplier(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  typeSupplierFill();  } |
| Добавление данных |
| private void btInsertTypeSupplier\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Type\_Supplier] ([Name\_Supplier]) values ('{0}')", tbTypeSupplierName.Text), DataBaseClass.act.manipulation);  tbTypeSupplierName.Clear();  } |
| Изменение данных |
| private void btUpdateTypeSupplier\_Click(object sender, RoutedEventArgs e)  {  if (lstbxTypeSupplier.Items.Count != 0 & lstbxTypeSupplier.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxTypeSupplier.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("update [dbo].[Type\_Supplier] set [Name\_Supplier] = '{0}' where [ID\_Type\_Supplier] = {1}", tbTypeSupplierName.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteTypeSupplier\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (lstbxTypeSupplier.Items.Count != 0 & lstbxTypeSupplier.SelectedItems.Count != 0)  {  DataBaseClass dataBaseClass = new DataBaseClass();  DataRowView dataRowView = (DataRowView)lstbxTypeSupplier.SelectedItems[0];  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Type\_Supplier] where [ID\_Type\_Supplier] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void lstbxTypeSupplier\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (lstbxTypeSupplier.Items.Count != 0 & lstbxTypeSupplier.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)lstbxTypeSupplier.SelectedItems[0];  tbTypeSupplierName.Text = dataRow[1].ToString();  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void manufactureFill()  {  if (cbCountrySelect.Items.Count != 0)  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("select [ID\_Manufacturer], [Name\_Manufacturer], [Country\_ID], [Name\_Country] from [dbo].[Manufacturer] inner join [dbo].[Country] on [ID\_Country] = [Country\_ID] where [Country\_ID] = {0}", (cbCountrySelect.SelectedValue == null) ? "0" : cbCountrySelect.SelectedValue), DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_manufacture;  dgManufacture.ItemsSource = dataBaseClass.resultTable.DefaultView;  dgManufacture.Columns[0].Visibility = Visibility.Hidden;  dgManufacture.Columns[1].Header = "Название производителя";  dgManufacture.Columns[2].Visibility = Visibility.Hidden;  dgManufacture.Columns[3].Header = "Страна производителя";  };  Dispatcher.Invoke(action);  }  }  private void Dependency\_OnChange\_manufacture(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  manufactureFill();  }  private void countrySelectFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Country], [Name\_Country] from [dbo].[Country]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_countrySelect;  cbCountrySelect.ItemsSource = dataBaseClass.resultTable.DefaultView;  cbCountrySelect.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  cbCountrySelect.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_countrySelect(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  countrySelectFill();  } |
| Добавление данных |
| private void InsertManufacture\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Manufacturer] ([Name\_Manufacturer], [Country\_ID]) " +  "values ('{0}', {1})", tbManufactureName.Text, cbCountrySelect.SelectedValue), DataBaseClass.act.manipulation);  tbManufactureName.Clear();  } |
| Изменение данных |
| private void UpdateManufacture\_Click(object sender, RoutedEventArgs e)  {  if (dgManufacture.Items.Count != 0 & dgManufacture.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgManufacture.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("update [dbo].[Manufacturer] set " +  "[Name\_Manufacturer] = '{0}'," +  "[Country\_ID] = {1} " +  "where [ID\_Manufacturer] = {2}", tbManufactureName.Text, cbCountrySelect.SelectedValue, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void DeleteManufacture\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (dgManufacture.Items.Count != 0 & dgManufacture.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgManufacture.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Manufacturer] where [ID\_Manufacturer] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void cbCountrySelect\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (cbCountrySelect.Items.Count != 0)  manufactureFill();  }  private void cbSupplerSelect\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (cbSupplerSelect.Items.Count != 0)  representativeFill();  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void warehouseFill()  {  if (cbMarketPlaceSelect.Items.Count != 0)  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("select [ID\_Warehouse], [Warehouse\_Adress], [Market\_Place\_ID], [Market\_Address] from [dbo].[Warehouse] inner join [dbo].[Market\_Place] on [ID\_Market\_Place] = [Market\_Place\_ID] where [Market\_Place\_ID] = {0}", (cbMarketPlaceSelect.SelectedValue == null) ? "0" : cbMarketPlaceSelect.SelectedValue), DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_warehouse;  dgWarehouse.ItemsSource = dataBaseClass.resultTable.DefaultView;  dgWarehouse.Columns[0].Visibility = Visibility.Hidden;  dgWarehouse.Columns[1].Header = "Адрес склада";  dgWarehouse.Columns[2].Visibility = Visibility.Hidden;  dgWarehouse.Columns[3].Header = "Адрес торговой точки";  };  Dispatcher.Invoke(action);  }  }  private void Dependency\_OnChange\_warehouse(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  warehouseFill();  }  private void marketPlaceSelectFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Market\_Place], [Market\_Address] from [dbo].[Market\_Place]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_marketPlaceSelect;  cbMarketPlaceSelect.ItemsSource = dataBaseClass.resultTable.DefaultView;  cbMarketPlaceSelect.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  cbMarketPlaceSelect.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_marketPlaceSelect(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  marketPlaceSelectFill();  } |
| Добавление данных |
| private void InsertWarehouse\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Warehouse] ([Warehouse\_Adress], [Market\_Place\_ID]) " +  "values ('{0}', {1})", tbWarehouseAdress.Text, cbMarketPlaceSelect.SelectedValue), DataBaseClass.act.manipulation);  tbWarehouseAdress.Clear();  } |
| Изменение данных |
| private void UpdateWarehouse\_Click(object sender, RoutedEventArgs e)  {  if (dgWarehouse.Items.Count != 0 & dgWarehouse.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgWarehouse.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("update [dbo].[Warehouse] set " +  "[Warehouse\_Adress] = '{0}'," +  "[Market\_Place\_ID] = {1} " +  "where [ID\_Warehouse] = {2}", tbWarehouseAdress.Text, cbMarketPlaceSelect.SelectedValue, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void DeleteWarehouse\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (dgWarehouse.Items.Count != 0 & dgWarehouse.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgWarehouse.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Warehouse] where [ID\_Warehouse] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void cbMarketPlaceSelect\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (cbMarketPlaceSelect.Items.Count != 0)  warehouseFill();  }  private void dgWarehouse\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (dgWarehouse.Items.Count != 0 & dgWarehouse.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)dgWarehouse.SelectedItems[0];  tbWarehouseAdress.Text = dataRow[1].ToString();  }  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void supplerFill()  {  if (dgSuppler != null & cbTypeSupplerSelect != null)  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("select [ID\_Suppler], [Type\_Suppler\_ID], [Name\_Supplier], [Full\_Name\_Suppler], [Short\_Name\_Suppler], [Legal\_Address\_Suppler], [Physical\_Address\_Suppler], [Suppler\_E\_Mail], [Suppler\_Phone\_Number], [BIC], [ITN], [ARCoEaO] from [dbo].[Suppler] inner join [dbo].[Type\_Supplier] on [ID\_Type\_Supplier] = [Type\_Suppler\_ID] where [Type\_Suppler\_ID] = {0}", (cbTypeSupplerSelect.Items.Count == 0 || cbTypeSupplerSelect.SelectedValue == null) ? "0" : cbTypeSupplerSelect.SelectedValue), DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_suppler;  dgSuppler.ItemsSource = dataBaseClass.resultTable.DefaultView;  dgSuppler.Columns[0].Visibility = Visibility.Hidden;  dgSuppler.Columns[1].Visibility = Visibility.Hidden;  dgSuppler.Columns[2].Header = "Тип фирмы";  dgSuppler.Columns[3].Header = "Полное название";  dgSuppler.Columns[4].Header = "Сокращённое название";  dgSuppler.Columns[5].Header = "Юр. адрем";  dgSuppler.Columns[6].Header = "Физ. Адрес";  dgSuppler.Columns[7].Header = "Адрес E-Mail";  dgSuppler.Columns[8].Header = "Номер телефона";  dgSuppler.Columns[9].Header = "БТК";  dgSuppler.Columns[10].Header = "ИНН";  dgSuppler.Columns[11].Header = "ОКПО";  };  Dispatcher.Invoke(action);  }  }  private void Dependency\_OnChange\_suppler(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  supplerFill();  }  private void typeSupplerSelectFill()  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("select [ID\_Type\_Supplier], [Name\_Supplier] from [dbo].[Type\_Supplier]"), DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_typeSupplerSelect;  Action action = () =>  {  cbTypeSupplerSelect.ItemsSource = dataBaseClass.resultTable.DefaultView;  cbTypeSupplerSelect.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  cbTypeSupplerSelect.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_typeSupplerSelect(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  typeSupplerSelectFill();  } |
| Добавление данных |
| private void btInsertSuppler\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Suppler] ([Type\_Suppler\_ID], [Full\_Name\_Suppler], [Short\_Name\_Suppler], [Legal\_Address\_Suppler], [Physical\_Address\_Suppler], [Suppler\_E\_Mail], [Suppler\_Phone\_Number], [BIC], [ITN], [ARCoEaO]) " +  "values ({0}, '{1}', '{2}', '{3}', '{4}', '{5}', '{6}', '{7}', '{8}', '{9}')", cbTypeSupplerSelect.SelectedValue, tbFullNameSuppler.Text, tbSortNameSuppler.Text, tbLegalAddressSuppler.Text, tbPhysicalAddressSuppler.Text, tbSupplerEMail.Text, msktbSupplerPhoneNumbe.Text, msktbBIC.Text, msktbITN.Text, msktbARCoEaO.Text), DataBaseClass.act.manipulation);  tbFullNameSuppler.Clear();  tbSortNameSuppler.Clear();  tbLegalAddressSuppler.Clear();  tbPhysicalAddressSuppler.Clear();  tbSupplerEMail.Clear();  msktbSupplerPhoneNumbe.Clear();  msktbBIC.Clear();  msktbITN.Clear();  msktbARCoEaO.Clear();  } |
| Изменение данных |
| private void btUpdateSuppler\_Click(object sender, RoutedEventArgs e)  {  if(dgSuppler.Items.Count != 0 & dgSuppler.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgSuppler.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("update [dbo].[Suppler] set " +  "[Type\_Suppler\_ID] = {0}," +  "[Full\_Name\_Suppler] = '{1}'," +  "[Short\_Name\_Suppler] = '{2}'," +  "[Legal\_Address\_Suppler] = '{3}'," +  "[Physical\_Address\_Suppler] = '{4}'," +  "[Suppler\_E\_Mail] = '{5}'," +  "[Suppler\_Phone\_Number] = '{6}'," +  "[BIC] = '{7}'," +  "[ITN] = '{8}'," +  "[ARCoEaO] = '{9}'" +  "where [ID\_Suppler] = {10}", cbTypeSupplerSelect.SelectedValue, tbFullNameSuppler.Text, tbSortNameSuppler.Text, tbLegalAddressSuppler.Text,  tbPhysicalAddressSuppler.Text, tbSupplerEMail.Text, msktbSupplerPhoneNumbe.Text, msktbBIC.Text, msktbITN.Text, msktbARCoEaO.Text,  dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteSuppler\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?",DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (dgSuppler.Items.Count != 0 & dgSuppler.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgSuppler.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Suppler] where [ID\_Suppler] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void cbTypeSupplerSelect\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (cbTypeSupplerSelect.Items.Count != 0)  supplerFill();  }  private void dgSuppler\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (dgSuppler.Items.Count != 0 & dgSuppler.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)dgSuppler.SelectedItems[0];  tbFullNameSuppler.Text = dataRow[3].ToString();  tbSortNameSuppler.Text = dataRow[4].ToString();  tbLegalAddressSuppler.Text = dataRow[5].ToString();  tbPhysicalAddressSuppler.Text = dataRow[6].ToString();  tbSupplerEMail.Text = dataRow[7].ToString();  msktbSupplerPhoneNumbe.Text = dataRow[8].ToString();  msktbBIC.Text = dataRow[9].ToString();  msktbITN.Text = dataRow[10].ToString();  msktbARCoEaO.Text = dataRow[11].ToString();  }  }  private void tbFullNameSuppler\_TextChanged(object sender, TextChangedEventArgs e)  {  string[] NewSohortName = tbFullNameSuppler.Text.Split(' ');  tbSortNameSuppler.Clear();  foreach (string name in NewSohortName)  {  if(name.Length >0)  tbSortNameSuppler.Text += name.Substring(0, 1).ToUpper();  }  }  private void tbLegalAddressSuppler\_TextChanged(object sender, TextChangedEventArgs e)  {  tbPhysicalAddressSuppler.Text = tbLegalAddressSuppler.Text;  } |
| Иллюстрация справочника | Вывод данных в справочник(и) |
|  | private void representativeFill()  {  Action action = () =>  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("select [ID\_Representative], [Suppler\_ID], [Full\_Name\_Suppler], [First\_Name\_Repres],[Second\_Name\_Repres], [Middle\_Name\_Repres] from [dbo].[Representative] inner join [dbo].[Suppler] on [Suppler\_ID] = [ID\_Suppler] where [Suppler\_ID] = {0}", (cbSupplerSelect.Items.Count == 0 || cbSupplerSelect.SelectedValue == null) ? "0" : cbSupplerSelect.SelectedValue), DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_representative;  dgRepresentative.ItemsSource = dataBaseClass.resultTable.DefaultView;  dgRepresentative.Columns[0].Visibility = Visibility.Hidden;  dgRepresentative.Columns[1].Visibility = Visibility.Hidden;  dgRepresentative.Columns[2].Header = "Полное название организации";  dgRepresentative.Columns[3].Header = "Фамилия представителя";  dgRepresentative.Columns[4].Header = "Имя представителя";  dgRepresentative.Columns[5].Header = "Отчество представителя";  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_representative(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  representativeFill();  }  private void supplerSelectFill()  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute("select [ID\_Suppler], [Name\_Supplier]+' '+[Full\_Name\_Suppler] as \"SupplerInfo\" from [dbo].[Suppler] inner join [dbo].[Type\_Supplier] on [ID\_Type\_Supplier] = [Type\_Suppler\_ID]", DataBaseClass.act.select);  dataBaseClass.dependency.OnChange += Dependency\_OnChange\_supplerSelect;  Action action = () =>  {  cbSupplerSelect.ItemsSource = dataBaseClass.resultTable.DefaultView;  cbSupplerSelect.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;  cbSupplerSelect.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;  };  Dispatcher.Invoke(action);  }  private void Dependency\_OnChange\_supplerSelect(object sender, SqlNotificationEventArgs e)  {  if (e.Info != SqlNotificationInfo.Invalid)  supplerSelectFill();  } |
| Добавление данных |
| private void btInsertRepresentative\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("insert into [dbo].[Representative] ([Suppler\_ID], [First\_Name\_Repres], [Second\_Name\_Repres], [Middle\_Name\_Repres]) " +  "values ({0},'{1}','{2}','{3}')", cbSupplerSelect.SelectedValue, tbFirstNameRepres.Text, tbSecondNameRepres.Text, tbMiddleNameRepres.Text), DataBaseClass.act.manipulation);  tbFirstNameRepres.Clear();  tbSecondNameRepres.Clear();  tbMiddleNameRepres.Clear();  } |
| Изменение данных |
| private void btUpdateRepresentative\_Click(object sender, RoutedEventArgs e)  {  if (dgRepresentative.Items.Count != 0 & dgRepresentative.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgRepresentative.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("update [dbo].[Representative] set " +  "[Suppler\_ID] = {0}," +  "[First\_Name\_Repres] = '{1}'," +  "[Second\_Name\_Repres] = '{2}'," +  "[Middle\_Name\_Repres] = '{3}'" +  "where [ID\_Representative] = {4}",  cbSupplerSelect.SelectedValue, tbFirstNameRepres.Text, tbSecondNameRepres.Text, tbMiddleNameRepres.Text, dataRowView[0]), DataBaseClass.act.manipulation);  }  } |
| Удаление данных |
| private void btDeleteRepresentative\_Click(object sender, RoutedEventArgs e)  {  switch (MessageBox.Show("Удалит выбранную запись?", DataBaseClass.App\_Name, MessageBoxButton.YesNo, MessageBoxImage.Question))  {  case MessageBoxResult.Yes:  if (dgRepresentative.Items.Count != 0 & dgRepresentative.SelectedItems.Count != 0)  {  DataRowView dataRowView = (DataRowView)dgRepresentative.SelectedItems[0];  DataBaseClass dataBaseClass = new DataBaseClass();  dataBaseClass.sqlExecute(string.Format("delete from [dbo].[Representative] where [ID\_Representative] = {0}", dataRowView[0]), DataBaseClass.act.manipulation);  }  break;  }  } |
| Дополнительные функции |
| private void cbSupplerSelect\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (cbSupplerSelect.Items.Count != 0)  representativeFill();  }  private void dgRepresentative\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (dgRepresentative.Items.Count != 0 & dgRepresentative.SelectedItems.Count != 0)  {  DataRowView dataRow = (DataRowView)dgRepresentative.SelectedItems[0];  tbFirstNameRepres.Text = dataRow[3].ToString();  tbSecondNameRepres.Text = dataRow[4].ToString();  tbMiddleNameRepres.Text = dataRow[5].ToString();  }  } |

1. Реализовать окно авторизации администратора, при этом администратор работает из-под логина и пароля от СУБД Microsoft SQL Server;

|  |
| --- |
| Код разметки XAML |
| <Window x:Class="AdministratorApp.AuthorizationWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:AdministratorApp"  mc:Ignorable="d"  Title="Администратор - Авторизация" Height="200" Width="600" Name="AuthorizWindow" WindowStartupLocation="CenterScreen">  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="0.1\*"></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition Width="0.1\*"></ColumnDefinition>  <ColumnDefinition></ColumnDefinition>  <ColumnDefinition Width="0.1\*"></ColumnDefinition>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition Height="0.1\*"></RowDefinition>  </Grid.RowDefinitions>  <Label Name="lblLogin" Grid.Row="1" Grid.Column="1" Content="Логин администратора:" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="16"></Label>  <Label Name="lblPassword" Grid.Row="3" Grid.Column="1" Content="Пароль администратора:" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="16"></Label>  <PasswordBox Name="pbLogin" Grid.Row="1" Grid.Column="3" VerticalAlignment="Center" FontSize="16"></PasswordBox>  <PasswordBox Name="pbPassword" Grid.Row="3" Grid.Column="3" VerticalAlignment="Center" FontSize="16"></PasswordBox>  <Button Name="btEnter" Grid.Row="5" Grid.Column="1" Content="Вход" FontSize="16" Click="btEnter\_Click"></Button>  <Button Name="btCLose" Grid.Row="5" Grid.Column="3" Content="Выход" FontSize="16" Click="btCLose\_Click"></Button>  </Grid>  </Window> |
| Результат |
|  |
| Обработка событий |
| private void btEnter\_Click(object sender, RoutedEventArgs e)  {  DataBaseClass.Users\_ID = pbLogin.Password;  DataBaseClass.Password = pbPassword.Password;  DataBaseClass.ConnectionStrig= string.Format(DataBaseClass.ConnectionStrig, DataBaseClass.Users\_ID, DataBaseClass.Password);  DataBaseClass dataBaseClass = new DataBaseClass();  try  {  dataBaseClass.connection.Open();  Visibility = Visibility.Hidden;  SimpleTablesWindow simpleTablesWindow = new SimpleTablesWindow();  simpleTablesWindow.Show();  }  catch  {  MessageBox.Show("Не верный логин или пароль!", DataBaseClass.App\_Name);  pbLogin.Focus();  }  finally  {  dataBaseClass.connection.Close();  pbLogin.Clear();  pbPassword.Clear();  }  } |