Практическая работа 35
Разработка модуля для взаимодействия с базой данных

Цель занятия: Получить практический опыт подключения к базе данных, расположенной в Microsoft SQL Server.

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
MS SQL Server
MS Visual Studio

Краткие теоретические сведения

Для создания подключения к источнику данных в ADO.NET существует специальный объект Connection. В зависимости от выбранного источника данных этот объект может называться по-разному. Для создания подключения к базам данных MS SQL Server следует использовать объект SqlConnection, который находится в пространстве имен System.Data.SqlClient

Пространство имен System.Data.SqlClient является поставщиком данных .NET FrameWork для источников данных MS SQL Server

Для работы с объектом SqlConnection ему нужно предоставить строку соединения, которая указывает каким образом нужно подключиться к источнику данных. Строка соединения – строка, состоящая из пар имя — значение, содержащая сведения об инициализации, передаваемые в виде параметра от приложения к источнику данных. Синтаксис строки соединения зависит от выбранного источника данных.

Класс SqlConnection из пространства имен Microsoft.Data.SqlClient имеет три конструктора:

- SqlConnection()
- SqlConnection(String)
- SqlConnection(String, SqlCredential)

Второй и третий конструкторы в качестве первого параметра принимают строку подключения. Третий конструктор также принимает объект SqlCredential, который фактически представляет логин и пароль.

Теперь проверим подключение на примере сервера LocalDB:

```
using Microsoft.Data.SqlClient;
using System;
using System.Data;
using System.Threading.Tasks;
```

```
namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string connectionString =
"Server=(localdb)\\mssqllocaldb;Database=master;Trusted_Connection=True;";

            // Создание подключения
            SqlConnection connection = new SqlConnection(connectionString);
            try
            {
                // Открываем подключение
                await connection.OpenAsync();
                Console.WriteLine("Подключение открыто");
            }
            catch (SqlException ex)
            {
                Console.WriteLine(ex.Message);
            }
            finally
            {
                // если подключение открыто
                if (connection.State == ConnectionState.Open)
                {
                    // закрываем подключение
                    await connection.CloseAsync();
                    Console.WriteLine("Подключение закрыто...");
                }
            }
            Console.WriteLine("Программа завершила работу.");
            Console.Read();
        }
    }
}
```

В данном случае подключение осуществляется к серверу LocalDB и его базе данных master (по умолчанию база данных master уже должна быть на любом MS SQL Servere).

Для начала взаимодействия с базой данных нам надо открыть подключение с помощью методов Open() (синхронный) или OpenAsync() (асинхронный).

По окончании работы с SqlConnection необходимо закрыть подключение к серверу, вызвав метод Close()/CloseAsync() или Dispose()/DisposeAsync(). В данном случае вначале проверяем, что подключение открыто и, если оно открыто, вызываем асинхронный метод OpenAsync().

В итоге, если указана валидная строка подключения, то мы должны увидеть на консоли следующие строки:

```
Подключение открыто
Подключение закрыто...
Программа завершила работу.
```

Вместо явного закрытия подключения также можно использовать конструкцию using, которая автоматически закрывает подключение:

```
using Microsoft.Data.SqlClient;
using System;
using System.Threading.Tasks;

namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string connectionString = "Server=(localdb)\\mssqllocaldb;Database=master;Trusted_Connection=True;";
            using(SqlConnection connection = new SqlConnection(connectionString))
            {
                await connection.OpenAsync();
                Console.WriteLine("Подключение открыто");
            }
            Console.WriteLine("Подключение закрыто...");
            Console.WriteLine("Программа завершила работу.");
            Console.Read();
        }
    }
}
```

Получение информации о подключении

Объект SqlConnection обладает рядом свойств, которые позволяют получить информацию о подключении:

```
using Microsoft.Data.SqlClient;
using System;
using System.Threading.Tasks;
```

```
namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string connectionString =
"Server=(localdb)\\mssqllocaldb;Database=master;Trusted_Connection=True;";

            using(SqlConnection connection = new SqlConnection(connectionString))
            {
                await connection.OpenAsync();
                Console.WriteLine("Подключение открыто");
                // Вывод информации о подключении
                Console.WriteLine("Свойства подключения:");
                Console.WriteLine($"\tСтрока подключения:
{connection.ConnectionString}");
                Console.WriteLine($"\tБаза данных: {connection.Database}");
                Console.WriteLine($"\tСервер: {connection.DataSource}");
                Console.WriteLine($"\tВерсия сервера: {connection.ServerVersion}");
                Console.WriteLine($"\tСостояние: {connection.State}");
                Console.WriteLine($"\tWorkstationId: {connection.WorkstationId}");
            }
            Console.WriteLine("Подключение закрыто...");
            Console.WriteLine("Программа завершила работу.");
            Console.Read();
        }
    }
}
```

```
Подключение открыто
Свойства подключения:
        Строка подключения:
        База данных: master
        Сервер: (localdb)\mssqllocaldb
        Версия сервера: 15.00.2000
        Состояние: Open
        WorkstationId: EUGENEPC
Подключение закрыто...
Программа завершила работу.
```

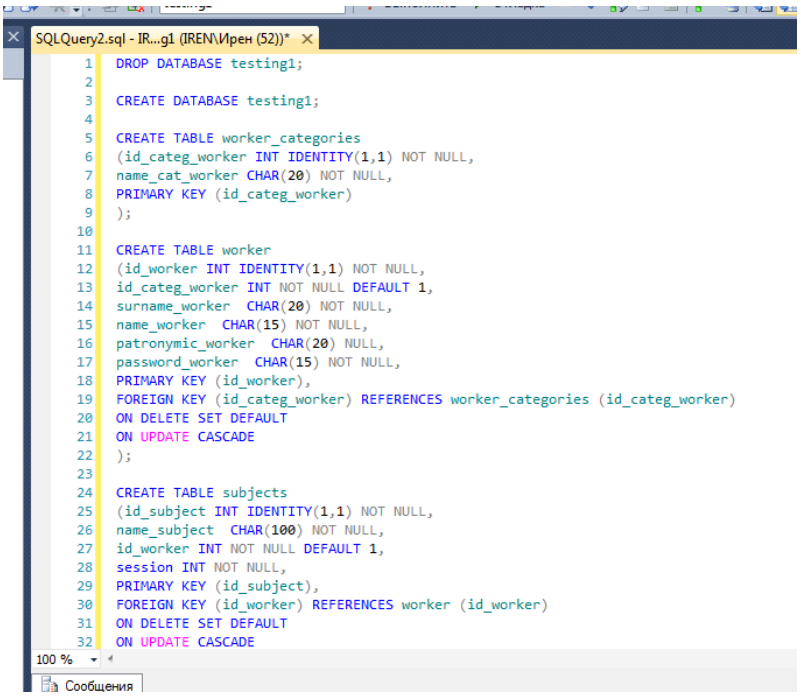
Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания разработать проект.

Порядок выполнения работы

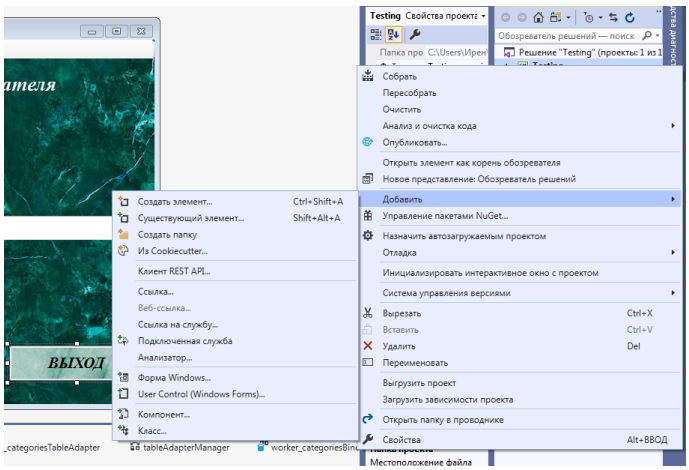
Рассмотрим на примере приложения Тестирование.
Создадим базу данных testing в SQL Server Management Studio.



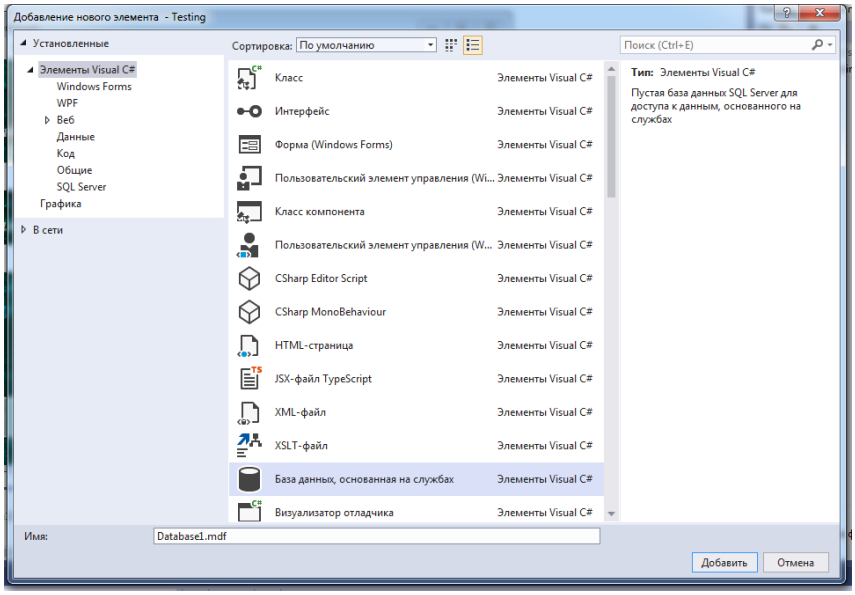
В результате анализа предметной области приложение имеет следующие внешние сущности: руководитель, сотрудник, преподаватель, студент, имеющие различные уровни доступа и привилегии. Добавим администратора базы данных, имеющего самый высокий уровень доступа, и гостя, имеющего доступ только к тестам без регистрации, для желающих проверить свои знания.

```
INSERT INTO dbo.worker_categories (name_cat_worker)
VALUES ('Администратор');
INSERT INTO dbo.worker_categories (name_cat_worker)
VALUES ('Сотрудник');
INSERT INTO dbo.worker_categories (name_cat_worker)
VALUES ('Преподаватель');
INSERT INTO dbo.worker_categories (name_cat_worker)
VALUES ('Студент');
INSERT INTO dbo.worker_categories (name_cat_worker)
VALUES ('Гость');
INSERT INTO dbo.worker_categories (name_cat_worker)
VALUES ('Руководитель');
```

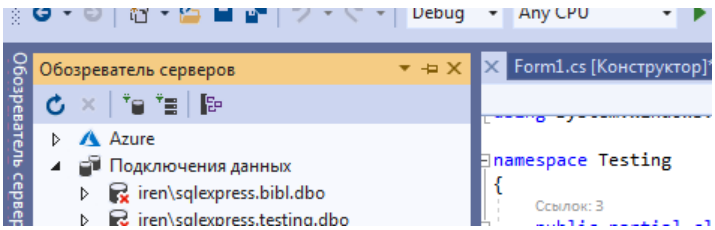
Для работы программы необходимо подключить базу данных к приложению. Нажмем ПКМ но проекте и выберем Добавить, а затем Компонент.



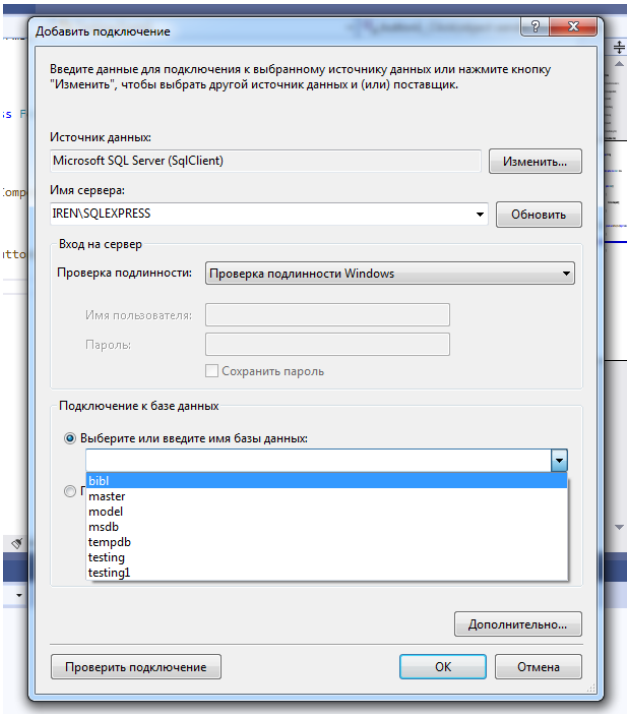
В открывшемся окне выбираем База данных, основанная на службах, а в разделе Имя пишем имя своей базы данных.



Во вкладке Обозреватель серверов нажмем значок Подключиться к базе данных

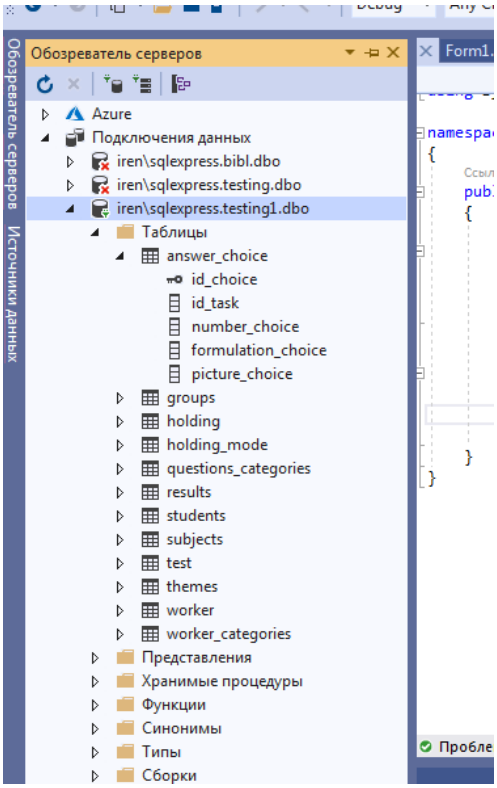


В окне **«Выбора источника»** выбираем **«Microsoft SQL Server»** (Можно подключить Access, тогда выбрать «Другое» и поставщик данных «Поставщик данных .NET Framework для OLE DB»):



При этом откроется диалог создания подключения, в нем выберем источник данных "Microsoft SQL Server…", В окне «Имя сервера» нажмем на стрелку, система сама найдет SQL сервер, аналогично с именем базы данных, которое можно выбрать из списка.

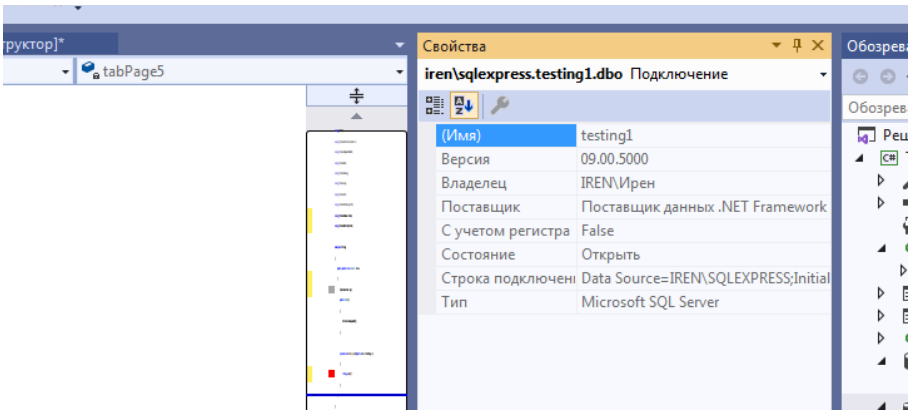
По окончанию установки связи с базой данных SQL сервер, во вкладке Обозреватель серверов можно просмотреть таблицы подключенной базы данных.



Прежде чем начать работать с этой базой данных, необходимо подключить следующие сборки:

```
using System.Data;
using System.Data.SqlClient;
```

Создадим объект класса SqlConnection для соединения с базой данных, параметром конструктора класса является путь к ней, для этого необходим физический адрес подключенной базы данных. Чтоб его получить в обозревателе серверов кликнем ПКМ по названию БД и в контекстном меню откроем Свойства



Здесь скопируем данные раздела Строка подключения. Запишем их в строковую переменную, а затем создадим подключение

```
SqlConnection sqlc= new SqlConnection(conn);
```

Чтобы это подключение действовало для всех методов и событий модуля объявим в классе Form поле типа SqlConnection:

```
SqlConnection sqlc;
```

А в коде будем использовать это поле.

```
sqlc= new SqlConnection(conn);
```

После подключения к базе данных необходимо ее открыть с помощью метода sqlc.Open(), затем с использованием класса SqlCommand составить запрос к подключенной базе данных, после чего данные, которые хранятся в таблице, нужно считать, для этого потребуется метод ExecuteReader(), возвращающий объект SqlDataReader, который используется для чтения данных.

В цикле while (reader.Read()) в порядке следования столбцов получаем данные с помощью метода GetValue(), который возвращает данные в виде объекта типа object. После завершения работы с SqlDataReader надо его закрыть методом Close(). И пока один SqlDataReader не закрыт, другой объект SqlDataReader для одного и того же подключения мы использовать не сможем.

Для того, чтобы процесс подключения к базе данных, получения информации из нее не тормозил работу пользователя, лучше использовать потоки или асинхронные методы.

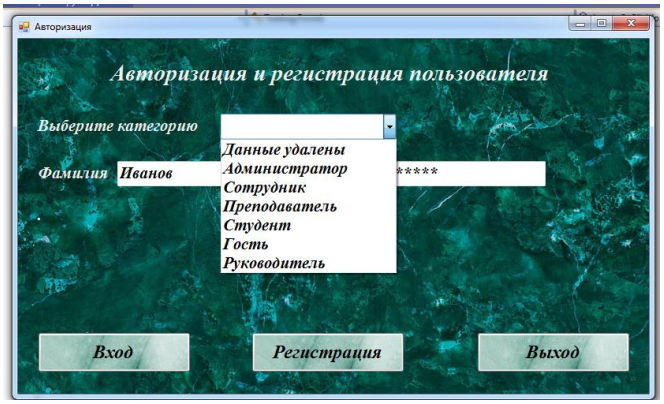
В методе private void Form1_Load(object sender, EventArgs e) добавляем слово async: private async void Form1_Load(object sender, EventArgs e). Внутри метода в командах добавляется await, например await sqlc.OpenAsync().

Для асинхронного чтения, во-первых, применяется метод ExecuteReaderAsync() класса SqlCommand, и во-вторых, метод ReadAsync() класса SqlDataReader:

После окончания работы с базой данных необходимо закрыть не только SqlDataReader, но и подключение к БД, лучше с проверкой условия наличия этого подключения и статуса:

```
if (sqlc != null && sqlc.State != ConnectionState.Closed) sqlc.Close();
```

В результате форма выглядит так



Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Вывод

Варианты заданий

Варианты заданий представлены в практической работе 28

Используемая литература

1. Гниденко, И. Г. Технология разработки программного обеспечения: учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.
2. Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.
3. Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.
4. Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.
5. http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.