**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по дисциплине **«Языки программирования высокого уровня»**

на тему: **«Использование технологии ADO.NET для доступа к данных»**

Выполнил: студент гр. ИП-22

Коваленко А. И.

Принял: ст. преподаватель Романькова Т. Л.

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2022

**Использование технологии ADO.NET для доступа к данных**

**Цель**: *познакомиться с различными режимами работы в ADO.Net, научиться работать с объектами подключений, объектами команд и объектами чтения данных с использованием различных провайдеров, получить навыки использования классов DataSet, DataTable и объектов адаптеров. Познакомиться с возможностями Entity Framework*

**Задание.**

Разработать приложение, позволяющее выполнять основные операции с данными из различных источников:

* Просмотр в виде таблицы, поиск, фильтрация по различным критериям
* Добавление нового объекта с подтверждением или отменой действия
* Удаление текущего объекта с диалогом контроля

Вариант предметной области взять в соответствии с заданием в файле ***Предметные области.docx***

Реализовать решение в виде следующих проектов:

1. *Приложение Windows Forms* для организации интерфейса пользователя, которое позволяет:

* Загрузить данные из выбранного источника в коллекцию. В качестве источников данных использовать:
* XML-.файлы;
* JSON- файлы;
* базы данных MS SQL,
* базы данных SQLite

* Просмотр и корректировку данных:
* Просмотр в виде таблицы, поиск, фильтрация
* Добавление нового объекта с диалогом контроля
* Удаление текущего объекта с диалогом контроля
* Сохранить данные в файле или в базе данных по выбору пользователя. Обеспечить сохранение данных с помощью двух диалогов: «Сохранить» и «Сохранить как…». Диалог «Сохранить как …» должен включать выбор типа хранилища данных

1. *Библиотека классов* для обеспечения доступа к файлам – источникам данных. Должна включать средства для чтения и записи объектов в файлы следующих типов:
   * XML-.файлы;
   * JSON- файлы.
2. *Библиотека классов*, обеспечивающих **доступ к данным на основе ADO.NET.** Создатьклассы доступа к данным

* для БД MS SQL,
* для БД SQLite

Классы должны включать открытие, чтение и закрытие источника данных.

1. *Библиотека классов*, обеспечивающих **доступ к данным на основе Entity Framework**

Создатьклассы доступа к данным

* для БД MS SQL,
* для БД SQLite

Классы должны включать открытие, чтение и закрытие источника данных.

| 7) Разработка ИС «Географический справочник» | 1. Страна-название  2. Площадь,в млн.  кв.км  3. Население, в  млн.чел.  4. Континет  5. Столица | Население | Столица, вхождение подстроки в строку |
| --- | --- | --- | --- |

Листинг программы:

Notebook.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6.AdoDataAccess

{

public class Notebook

{

public string Day { get; set; }

public DateTime Time { get; set; }

public string Event { get; set; }

public string Name { get; set; }

public string Number { get; set; }

}

}

NotebookContext.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6.AdoDataAccess

{

public class NotebookContext

{

public const string AdoMSSQLConnectionString = "Server=DESKTOP-AIDC8ES\\SQLEXPRESS;" +

"Database=Notebook;" +

"Trusted\_Connection=True;" +

"MultipleActiveResultSets=true;";

public const string AdoSQLiteConnectionString = @"Data Source=D:\универ\Programms 2\Programms 2\ЯПВУ\6-th Lab\6\Notebook.db";

}

}

AdoMSSQLRepository.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6.AdoDataAccess

{

public class AdoMSSQLRepository

{

public void Create(Notebook item)

{

string sqlExpression = $"INSERT INTO Notebook(Day, Time, Event, Name, Number)" +

" VALUES (@day, @time, @event, @name, @number);";

using (SqlConnection connection = new SqlConnection(NotebookContext.AdoMSSQLConnectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(sqlExpression, connection))

{

command.Parameters.AddRange(new SqlParameter[]

{

new SqlParameter("@day", item.Day),

new SqlParameter("@time", item.Time),

new SqlParameter("@event",item.Event),

new SqlParameter("@name",item.Name),

new SqlParameter("@number",item.Number),

}); ;

command.ExecuteNonQuery();

}

}

}

public void Delete(string evvent)

{

string sqlExpression = "DELETE FROM Notebook WHERE Event = @event";

using (SqlConnection connection = new SqlConnection(NotebookContext.AdoMSSQLConnectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(sqlExpression, connection))

{

SqlParameter typeParam = new SqlParameter("@event", evvent);

command.Parameters.Add(typeParam);

command.ExecuteNonQuery();

}

}

}

public IEnumerable<Notebook> GetAll()

{

string sqlExpression = "SELECT Day, Time, Event, Name, Number FROM Notebook";

List<Notebook> notebooks = new List<Notebook>();

using (SqlConnection connection = new SqlConnection(NotebookContext.AdoMSSQLConnectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(sqlExpression, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

notebooks.Add(new Notebook()

{

Day = (string)reader["Day"],

Time = (DateTime)reader["Time"],

Event = (string)reader["Event"],

Name = (string)reader["Name"],

Number = (string)reader["Number"],

});

}

}

}

}

return notebooks;

}

public Notebook GetById(string evvent)

{

string sqlExpression = "SELECT Day, Time, Event, Name, Number FROM Notebook" +

" WHERE Event = @event";

using (SqlConnection connection = new SqlConnection(NotebookContext.AdoMSSQLConnectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(sqlExpression, connection))

{

SqlParameter idParam = new SqlParameter(@"event", evvent);

command.Parameters.Add(idParam);

using (SqlDataReader reader = command.ExecuteReader())

{

return reader.Read() ? new Notebook()

{

Day = (string)reader["Day"],

Time = (DateTime)reader["Time"],

Event = (string)reader["Event"],

Name = (string)reader["Name"],

Number = (string)reader["Number"],

} : null;

}

}

}

}

public void Update(Notebook item)

{

string sqlExpression = "UPDATE Notebook SET Day=@day, Time=@time, Event=@event, Name=@name, Number=@number" +

" FROM Notebook" +

" WHERE Event = @event";

using (SqlConnection connection = new SqlConnection(NotebookContext.AdoMSSQLConnectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(sqlExpression, connection))

{

command.Parameters.AddRange(new SqlParameter[]

{

new SqlParameter("@day", item.Day),

new SqlParameter("@time", item.Time),

new SqlParameter("@event",item.Event),

new SqlParameter("@name",item.Name),

new SqlParameter("@number",item.Number),

});

command.ExecuteNonQuery();

}

}

}

public void Save(List<Notebook> notebooks)

{

var item = GetAll();

foreach (var i in item)

Delete(i.Event);

foreach (var notebook in notebooks)

Create(notebook);

}

}

}

AdoSQLiteRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Data.Sqlite;

namespace lab6.AdoDataAccess

{

public class AdoSQLiteRepository

{

public void Create(Notebook item)

{

string sqlExpression = $"INSERT INTO Notebook(Day, Time, Event, Name, Number)" +

" VALUES (@day, @time, @event, @name, @number);";

using (SqliteConnection connection = new SqliteConnection(NotebookContext.AdoSQLiteConnectionString))

{

connection.Open();

using (SqliteCommand command = new SqliteCommand(sqlExpression, connection))

{

command.Parameters.AddRange(new SqliteParameter[]

{

new SqliteParameter("@day", item.Day),

new SqliteParameter("@time", item.Time),

new SqliteParameter("@event",item.Event),

new SqliteParameter("@name",item.Name),

new SqliteParameter("@number",item.Number),

});

command.ExecuteNonQuery();

}

}

}

public void Delete(string evvent)

{

string sqlExpression = "DELETE FROM Notebook WHERE Event = @event";

using (SqliteConnection connection = new SqliteConnection(NotebookContext.AdoSQLiteConnectionString))

{

connection.Open();

using (SqliteCommand command = new SqliteCommand(sqlExpression, connection))

{

SqliteParameter idParam = new SqliteParameter("@event", evvent);

command.Parameters.Add(idParam);

command.ExecuteNonQuery();

}

}

}

public IEnumerable<Notebook> GetAll()

{

string sqlExpression = "SELECT Day, Time, Event, Name, Number FROM Notebook";

List<Notebook> notebooks = new List<Notebook>();

using (var connection = new SqliteConnection(NotebookContext.AdoSQLiteConnectionString))

{

connection.Open();

SqliteCommand command = new SqliteCommand(sqlExpression, connection);

using (SqliteDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

notebooks.Add(new Notebook()

{

Day = reader["Day"].ToString(),

Time = Convert.ToDateTime(reader["Time"]),

Event = reader["Event"].ToString(),

Name = reader["Name"].ToString(),

Number = reader["Number"].ToString(),

});

}

}

}

}

return notebooks;

}

public void Update(Notebook item)

{

string sqlExpression = "UPDATE Notebook SET Day=@day, Time=@time, Event=@event, Name=@name, Number=@number" +

" FROM Notebook" +

" WHERE Event = @event";

using (SqliteConnection connection = new SqliteConnection(NotebookContext.AdoSQLiteConnectionString))

{

connection.Open();

using (SqliteCommand command = new SqliteCommand(sqlExpression, connection))

{

command.Parameters.AddRange(new SqliteParameter[]

{

new SqliteParameter("@day", item.Day),

new SqliteParameter("@time", item.Time),

new SqliteParameter("@event",item.Event),

new SqliteParameter("@name",item.Name),

new SqliteParameter("@number",item.Number),

});

command.ExecuteNonQuery();

}

}

}

public void Save(List<Notebook> notebooks)

{

var item = GetAll();

foreach (var i in item)

Delete(i.Event);

foreach (var notebook in notebooks)

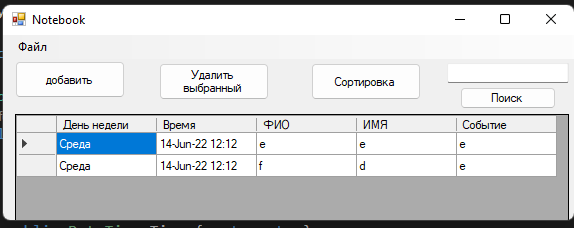
Create(notebook);

}

}

}

**Результат выполнения программы:**

****

**Вывод:** В ходе лабораторной работы были изучены особенности работs с объектами подключений, объектами команд и объектами чтения данных с использованием различных провайдеров, получить навыки использования классов DataSet, DataTable и объектов адаптеров.