Шпаргалка по работе с базами данных MySQL

1. Установка необходимых библиотек

Для работы с MySQL в C# необходимо установить библиотеку MySql.Data. Это можно сделать через NuGet Package Manager:

```
Install-Package MySql.Data
```

2. Подключение к базе данных

```
using MySql.Data.MySqlClient; // Подключаем библиотеку для работы с MySQL

// Строка подключения к базе данных
string connectionString = "Server=localhost;Database=your_database;User ID=your_user;Password=your_password;";
```

3. Чтение данных

```
private void LoadData()
{
    // SQL запрос для выбора всех данных из таблицы
    string query = "SELECT * FROM your_table";

    // Создаем соединение с базой данных
    using (MySqlConnection connection = new MySqlConnection(connectionString))
    {
        connection.Open(); // Открываем соединение

        // Создаем команду для выполнения SQL запроса
        using (MySqlCommand command = new MySqlCommand(query, connection))
        {
            using (MySqlDataReader reader = command.ExecuteReader()) // Выполняем запрос и получаем данные
            {
                // Читаем данные из базы и выводим в консоль
                while (reader.Read())
                {
                    Console.WriteLine(reader["column_name"].ToString()); // Замените column_name на нужное имя столбца
                }
            }
        }
    }
}
```

4. Вставка данных

```
private void InsertData(string value1, string value2)
{
    // SQL запрос для вставки данных
    string insertQuery = "INSERT INTO your_table (column1, column2) VALUES (@value1, @value2)";

    // Создаем соединение с базой данных
    using (MySqlConnection connection = new MySqlConnection(connectionString))
    {
        connection.Open(); // Открываем соединение

        // Создаем команду для выполнения SQL запроса
        using (MySqlCommand command = new MySqlCommand(insertQuery, connection))
        {
            // Добавляем параметры к запросу
            command.Parameters.AddWithValue("@value1", value1);
            command.Parameters.AddWithValue("@value2", value2);
            command.ExecuteNonQuery(); // Выполняем запрос
        }
    }
}
```

5. Обновление данных

```
private void UpdateData(string newValue, string conditionValue)
{
    // SQL запрос для обновления данных
    string updateQuery = "UPDATE your_table SET column1 = @value WHERE column2 = @condition";

    // Создаем соединение с базой данных
    using (MySqlConnection connection = new MySqlConnection(connectionString))
    {
        connection.Open(); // Открываем соединение

        // Создаем команду для выполнения SQL запроса
        using (MySqlCommand command = new MySqlCommand(updateQuery, connection))
        {
            // Добавляем параметры к запросу
            command.Parameters.AddWithValue("@value", newValue);
            command.Parameters.AddWithValue("@condition", conditionValue);
            command.ExecuteNonQuery(); // Выполняем запрос
        }
    }
}
```

6. Удаление данных

```
private void DeleteData(string conditionValue)
{
    // SQL запрос для удаления данных
    string deleteQuery = "DELETE FROM your_table WHERE column_name = @condition";

    // Создаем соединение с базой данных
    using (MySqlConnection connection = new MySqlConnection(connectionString))
    {
        connection.Open(); // Открываем соединение

        // Создаем команду для выполнения SQL запроса
        using (MySqlCommand command = new MySqlCommand(deleteQuery, connection))
        {
            // Добавляем параметр к запросу
            command.Parameters.AddWithValue("@condition", conditionValue);
            command.ExecuteNonQuery(); // Выполняем запрос
        }
    }
}
```

7. Обработка ошибок

```
try
{
    // Ваш код для работы с БД
}
catch (MySqlException ex)
{
    // Обработка ошибок подключения к БД
    MessageBox.Show($"Ошибка: {ex.Message}");
}
```

8. Закрытие соединения

Всегда закрывайте соединение с базой данных после выполнения операций.

```
finally
{
    if (connection != null && connection.State == System.Data.ConnectionState.Open)
    {
        connection.Close();
    }
}
```

Основные классы библиотеки MySql.Data:

1. MySqlConnection

Класс для управления подключением к базе данных.
Основные методы:
Open()
Открывает соединение с базой данных.
Close()
Закрывает соединение с базой данных.
Dispose()
Освобождает ресурсы, используемые объектом MySqlConnection.
BeginTransaction()
Начинает транзакцию в базе данных.

2. MySqlCommand

Класс для выполнения SQL-запросов.
Основные методы:
ExecuteNonQuery()
Выполняет SQL-запросы, которые не возвращают данные (например, INSERT, UPDATE, DELETE). Возвращает количество затронутых строк.
ExecuteReader()
Выполняет SQL-запросы, которые возвращают данные (например, SELECT). Возвращает объект MySqlDataReader для чтения данных.
ExecuteScalar()
Выполняет SQL-запрос и возвращает первое значение первой строки результата (например, для запросов с COUNT, SUM и т.д.).
Prepare()
Подготавливает команду для выполнения, что может повысить производительность при многократном выполнении.

3. MySqlDataReader

Класс для чтения данных, возвращаемых SQL-запросом.
Основные методы:
Read()
Перемещает курсор к следующей строке данных. Возвращает true, если данные доступны, и false, если строк больше нет.
GetString(), GetInt32(), GetBoolean() и т.д.
Получает значение указанного типа из текущей строки.
GetOrdinal(string columnName)
Возвращает индекс столбца по его имени.
Close()
Закрывает объект MySqlDataReader.

4. MySqlDataAdapter

Класс для заполнения DataSet или DataTable данными из базы данных.
Основные методы:
Fill(DataSet dataSet)
Заполняет DataSet данными из базы данных.
Fill(DataTable dataTable)
Заполняет DataTable данными из базы данных.
Update(DataSet dataSet)
Обновляет базу данных на основе изменений в DataSet.

5. MySqlParameter

Класс для работы с параметрами в SQL-запросах.

Основные свойства:
ParameterName
Имя параметра (например, @name).
Value
Значение параметра.
DbType
Тип данных параметра.

6. MySqlHelper
Статический класс для упрощённой работы с базой данных.
Основные методы:
ExecuteNonQuery(string connectionString, string query)
Выполняет SQL-запрос без возврата данных.
ExecuteReader(string connectionString, string query)
Выполняет SQL-запрос и возвращает MySqlDataReader.
ExecuteScalar(string connectionString, string query)
Выполняет SQL-запрос и возвращает первое значение первой строки.