Шпаргалка: Асинхронные методы MySql.Data

Краткое описание методов

OpenAsync()

- Асинхронно открывает соединение с базой данных
- Возвращает Task
- Может принимать CancellationToken для возможности отмены операции

ExecuteNonQueryAsync()

- Асинхронно выполняет SQL-команды, которые не возвращают данные (INSERT, UPDATE, DELETE)
- Возвращает Task<int> - количество затронутых строк
- Используется для операций изменения данных

ExecuteReaderAsync()

- Асинхронно выполняет SQL-запросы, возвращающие набор строк (SELECT)
- Возвращает Task<MySqlDataReader>
- Позволяет построчно читать результаты запроса

ExecuteScalarAsync()

- Асинхронно выполняет запрос и возвращает одно значение
- Возвращает Task<object>
- Полезен для агрегатных функций (COUNT, SUM, etc.)

GetFieldValueAsync<T>()

- Асинхронно получает значение определённого поля из текущей строки
- Возвращает Task<T>
- Позволяет явно указать тип возвращаемого значения

Основные операции

Подключение

```
// Открытие соединения
await connection.OpenAsync();
await connection.OpenAsync(cancellationToken);

// Закрытие соединения
await connection.CloseAsync();
await connection.CloseAsync(cancellationToken);
```

Выполнение команд

```
MySqlCommand command = new MySqlCommand(query, connection);

// INSERT, UPDATE, DELETE
await command.ExecuteNonQueryAsync();

// Получение одного значения (например, COUNT)
object result = await command.ExecuteScalarAsync();

// Получение набора данных
MySqlDataReader reader = await command.ExecuteReaderAsync();
```

Работа с Reader

```
// Чтение данных
while (await reader.ReadAsync())
{
    string value = reader["column_name"].ToString();
    // или с явным указанием типа
    int number = await reader.GetFieldValueAsync<int>(0);
    string text = await reader.GetFieldValueAsync<string>("column_name");
}
```

Примеры использования

SELECT запрос

```
async Task<List<User>> GetUsersAsync(string city)
{
    var users = new List<User>();
    using var connection = new MySqlConnection(connectionString);
    await connection.OpenAsync();

    var command = new MySqlCommand(
        "SELECT * FROM users WHERE city = @city",
        connection
    );
    command.Parameters.AddWithValue("@city", city);

    using var reader = await command.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        users.Add(new User
        {
            Id = Convert.ToInt32(reader["id"]),
            Name = reader["name"].ToString(),
            City = reader["city"].ToString()
        });
    }

    return users;
}
```

INSERT запрос

```
async Task AddUserAsync(string name, string city)
{
    using var connection = new MySqlConnection(connectionString);
    await connection.OpenAsync();

    var command = new MySqlCommand(
        "INSERT INTO users (name, city) VALUES (@name, @city)",
        connection
    );

    command.Parameters.AddWithValue("@name", name);
    command.Parameters.AddWithValue("@city", city);

    await command.ExecuteNonQueryAsync();
}
```

UPDATE запрос

```
async Task UpdateUserCityAsync(int userId, string newCity)
{
    using var connection = new MySqlConnection(connectionString);
    await connection.OpenAsync();

    var command = new MySqlCommand(
        "UPDATE users SET city = @city WHERE id = @id",
        connection
    );

    command.Parameters.AddWithValue("@city", newCity);
    command.Parameters.AddWithValue("@id", userId);

    await command.ExecuteNonQueryAsync();
}
```

DELETE запрос

```
// Простой DELETE по ID
async Task DeleteUserAsync(int userId)
{
    using var connection = new MySqlConnection(connectionString);
    await connection.OpenAsync();

    var command = new MySqlCommand(
        "DELETE FROM users WHERE id = @id",
        connection
    );

    command.Parameters.AddWithValue("@id", userId);

    await command.ExecuteNonQueryAsync();
}

// Пример удаления с дополнительным условием
async Task DeleteUsersByConditionAsync(string city, int ageLimit)
{
    using var connection = new MySqlConnection(connectionString);
    await connection.OpenAsync();

    var command = new MySqlCommand(
        "DELETE FROM users WHERE city = @city AND age < @age",
        connection
    );

    command.Parameters.AddWithValue("@city", city);
    command.Parameters.AddWithValue("@age", ageLimit);

    int rowsAffected = await command.ExecuteNonQueryAsync();
    // rowsAffected содержит количество удаленных записей
}
```

Важные замечания

1. Всегда используйте `using` для освобождения ресурсов
2. Используйте параметризованные запросы для защиты от SQL-инъекций
3. Обрабатывайте исключения при работе с базой данных
4. Закрывайте reader перед выполнением новых команд
5. Используйте CancellationToken для возможности отмены длительных операций

Типичные исключения

- `MySqlException` - общие ошибки MySQL

- `InvalidOperationException` - ошибки состояния соединения
- `OperationCanceledException` - операция была отменена
- `ObjectDisposedException` - попытка использовать закрытое соединение