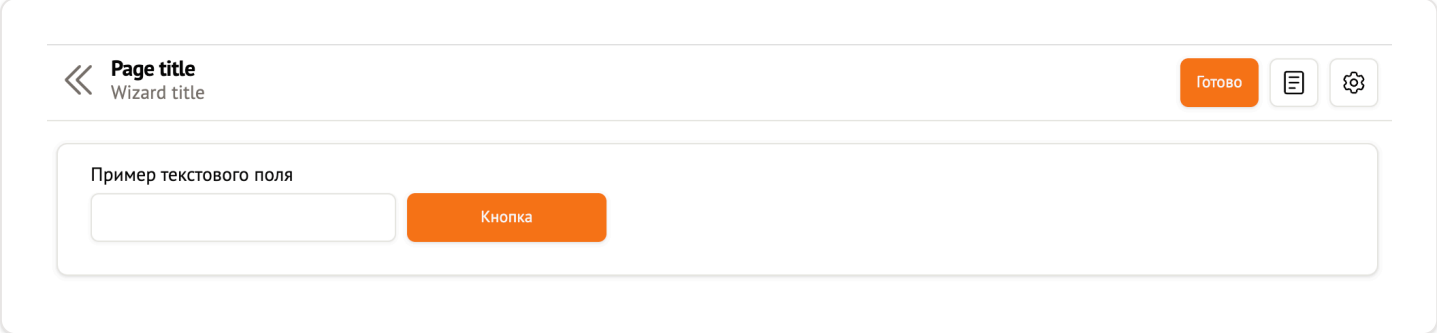
Взаимодействие с базой данных из .NET

Для работы с базой данных используется класс DataAccessor. В редакторе начните вводить `select` или `new` — сработает сниппет с готовым шаблоном.



Получение данных и создание новых записей

- В блоке `From` укажите таблицу, из которой нужно получить данные.
- В блоке `Select` перечислите поля, которые требуется вернуть. Можно оставить список пустым или указать `*` — тогда вернутся все поля таблицы. Для таблиц расширения 1–1 используйте формат:
  - конкретное поле: `ExtendedTable.FieldName`
  - все поля расширения: `ExtendedTable.*`
- По умолчанию для всех ссылочных полей и системных словарей возвращаются два значения: само числовое значение (например, поле User с Id связанного пользователя) и поле вида `User__Title`, которое содержит отображаемое название записи (обычно FullName) или текст словаря.
- Для защиты от SQL-инъекций используйте параметры в условиях.
- Результат содержит свойство Rows типа `List<Dictionary<string, object>>`, где каждый словарь — одна строка результата.

```
// Получение всех строк таблицы со статусом Status = 1
var data = DataAccessor.New()
    .From("DemoTable")
    .Select(["*"])
    .Where("[Status] = 1")
    .Run();

// Получение строк, у которых стоимость больше 10 000, с передачей условия через параметр
var data = DataAccessor.New()
    .From("DemoTable")
    .Select(["FullName", "Cost", "Comment"])
    .Where("[Cost] > @p1")
    .WithParameters(new Dictionary<string, object>()
    {
        { "@p1", 10000 },
    })
    .Run();
```

- Чтобы подготовить новые записи (например, инциденты или заявки), вызовите `New()` с указанием количества строк. В этом случае список будет заполнен полями таблицы со значениями по умолчанию, а Id каждой записи будет равен 0.

```
// Создать заготовки для 3 новых записей в таблице DemoTable
var data = DataAccessor.New(3)
    .From("DemoTable")
    .Run();
```

Работа с `List<Dictionary<string, object>>`

Доступны все возможности .NET для работы со списками и словарями.

- Так как значения хранятся в словаре `Dictionary<string, object>`, указывайте тип при чтении:

```
// Первая запись из списка
var row = data.Rows[0];

// Доступ к значениям по SQL-имени поля с приведением типа
var name = (string)row["FullName"];
var userId = (int)row["User"];
```

- Примеры обхода и изменения строк:

```
// Цикл for
for (var i = 0; i < data.Rows.Count; i++)
{
    data.Rows[i]["FullName"] = $"Новая запись номер {i + 1}";
    data.Rows[i]["Status"] = 2; // В работе
    data.Rows[i]["DetectedDate"] = DateTime.UtcNow;
}

// Цикл foreach
foreach (var row in data.Rows)
{
    row["FullName"] = "Новая запись";
    row["Status"] = 2; // В работе
    row["DetectedDate"] = DateTime.UtcNow;
}
```

Сохранение изменений в базе данных

- Пока вы изменяете данные в коде .NET, изменения не применяются к базе. Они будут записаны только после вызова методов `Insert()`, `Update()` или `Delete()`.
- После `Insert()` у каждой записи в `List<Dictionary<string, object>>` появится фактический Id (вместо `Id = 0`).

```
// Удалить записи
var data = DataAccessor.New()
    .From("DemoTable")
```

```
    .Select(["*"])
    .Where("[Status] = 1")
    .Run();

data.Delete();

// Создать записи
var data = DataAccessor.New(3)
    .From("DemoTable")
    .Run();

// .. заполнение атрибутов ..
data.Insert();
```