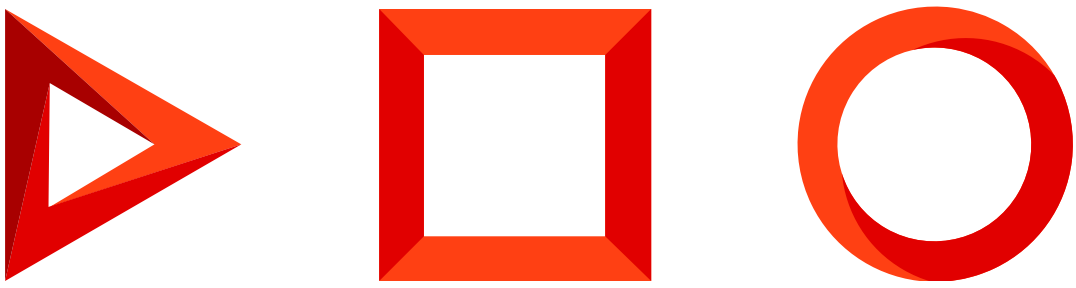
Creatio Academy

Операции с данными (back-end)

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Фоновое выполнение операций

Средний

Фоновое выполнение операций позволяет в фоновом режиме выполнять операции, которые требуют длительного времени выполнения, без задержек в работе пользовательского интерфейса.

Для запуска фоновых операций в классе `Terrasoft.Core.Tasks.Task` реализованы методы `StartNew()` и `StartNewWithUserConnection()`. В качестве параметров методов можно использовать базовые типы данных в .NET (например, `string`, `int`, `Guid` и т. д.) или пользовательские типы. **Отличие** метода `StartNewWithUserConnection()` — запуск фоновой операции, которая требует использования пользовательского соединения `UserConnection`.

Параметры, которые принимаются фоновой операцией, преобразуются в массив байт с помощью модуля `MessagePack-CSharp`. Реализация модуля `MessagePack-CSharp` содержится на сайте GitHub. Если не удается сериализовать или десериализовать значение параметра, могут сгенерироваться исключения.

Важно. В фоновой операции не рекомендуется использовать бесконечные циклы, поскольку это приводит к невозможности запуска в приложении других задач.

Действие асинхронной операции описывается в отдельном классе, который должен реализовать интерфейс `IBackgroundTask<in TParameters>`.

Интерфейс `IBackgroundTask<in TParameters>`

```
namespace Terrasoft.Core.Tasks
{
    public interface IBackgroundTask<in TParameters>
    {
        void Run(TParameters parameters);
    }
}
```

Если для выполнения действия требуется пользовательское соединение, то класс должен дополнительно реализовать интерфейс `IUserConnectionRequired`.

Интерфейс `IUserConnectionRequired`

```
namespace Terrasoft.Core
{
    public interface IUserConnectionRequired
    {
        void SetUserConnection(UserConnection userConnection);
```

```
    }
}
```

В классе, который реализует интерфейсы `IBackgroundTask<in TParameters>` и `IUserConnectionRequired` необходимо реализовать методы интерфейсов `Run` и `SetUserConnection`.

При **реализации методов** следует учесть:

- В метод `Run` не следует передавать `UserConnection`.
- В методе `Run` не следует вызывать метод `SetUserConnection` — ядро системы вызывает этот метод и инициализирует `UserConnection` при старте фоновой операции.
- В метод `Run` допустимо передавать структуры, состоящие только из простых типов данных. Если передавать сложные экземпляры классов, то с большой вероятностью произойдет ошибка сериализации парамеров.

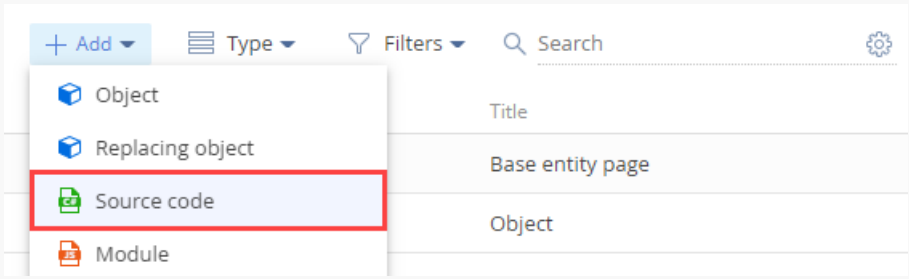
Зарегистрировать фоновую операцию

Средний

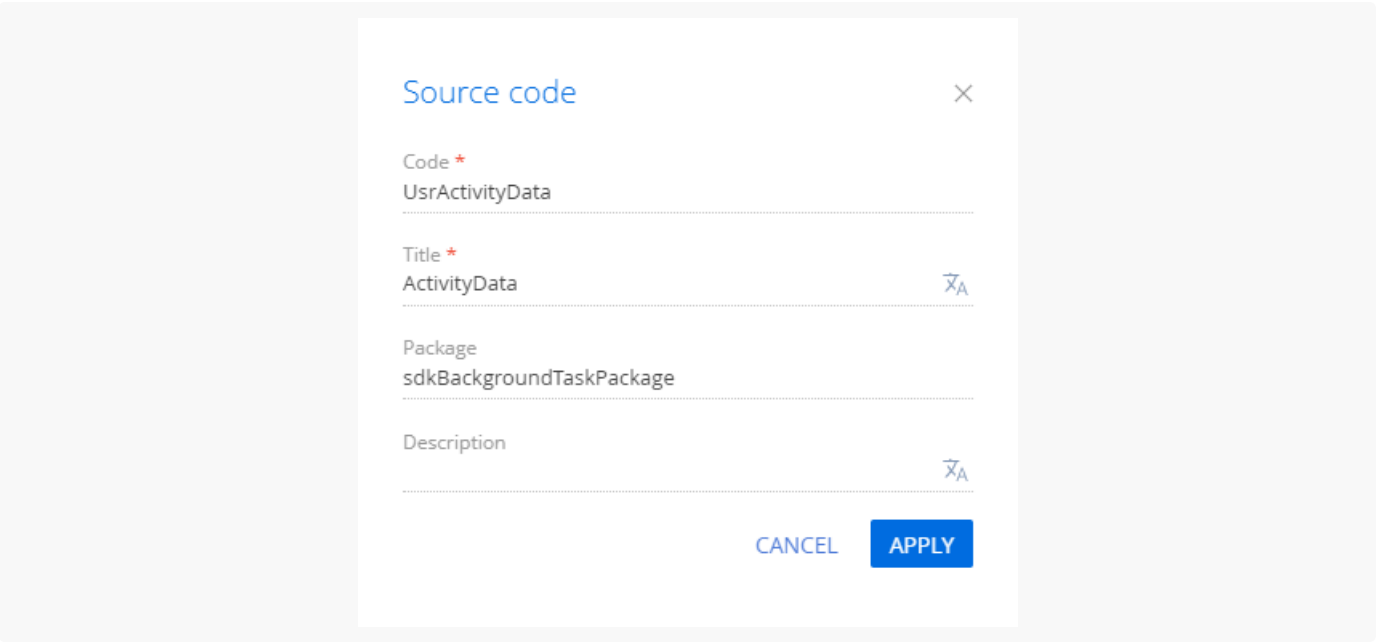
Пример. Создать бизнес-процесс, который регистрирует фоновую операцию. Фоновая операция выполняется около 30 секунд. По истечении этого времени в списочном представлении реестра раздела [*Активности*] ([*Activities*]) добавляется запись [*Activity created by background task*].

1. Создать класс для объекта активности

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Исходный код*] ([*Add*] —> [*Source code*]).



3. В дизайнере схем заполните свойства схемы:
   - [*Код*] ([*Code*]) — "`UsrActivityData`".
   - [*Заголовок*] ([*Title*]) — "`ActivityData`".



Для применения заданных свойств нажмите [*Применить*] ([*Apply*]).

4. В дизайнере схем добавьте исходный код.

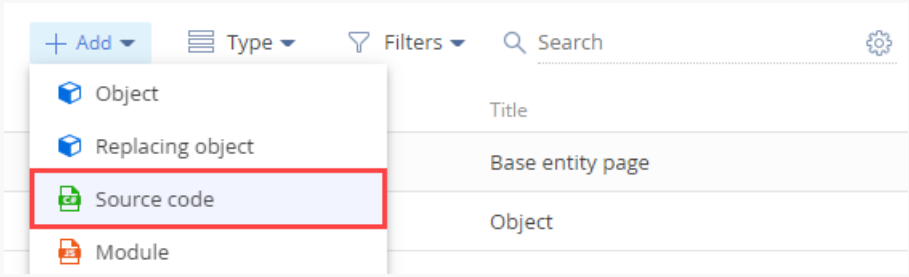
UsrActivityData

```
namespace Terrasoft.Configuration
{
    using System;
    using Terrasoft.Common;
    using Terrasoft.Core;
    using Terrasoft.Core.DB;
    public class UsrActivityData
    {
        public string Title { get; set; }
        public Guid TypeId { get; set; }
    }
}
```

5. На панели инструментов дизайнера нажмите [*Сохранить*] ([*Save*]), а затем [*Опубликовать*] ([*Publish*]).

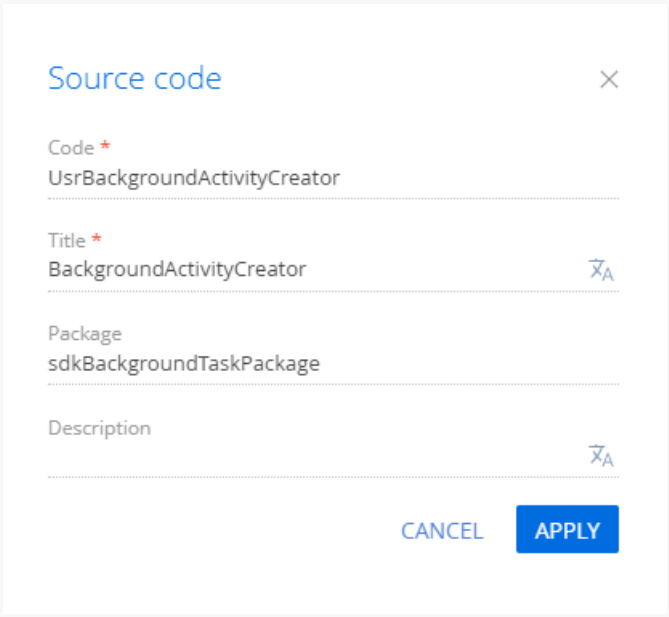
2. Создать класс для добавления активности

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Исходный код*] ([*Add*] —> [*Source code*]).



3. В дизайнере схем заполните свойства схемы:

   - [*Код*] ([*Code*]) — "`UsrBackgroundActivityCreator`".
   - [*Заголовок*] ([*Title*]) — "`BackgroundActivityCreator`".



   Для применения заданных свойств нажмите [*Применить*] ([*Apply*]).

4. В дизайнере схем добавьте исходный код.

`UsrBackgroundActivityCreator`

```
namespace Terrasoft.Configuration
{
    using System;
    using Terrasoft.Common;
    using Terrasoft.Core;
    using Terrasoft.Core.DB;
    using Terrasoft.Core.Tasks;
    using System.Threading.Tasks;

    public class UsrBackgroundActivityCreator : IBackgroundTask<UsrActivityData>, IUserConnec
    {
```

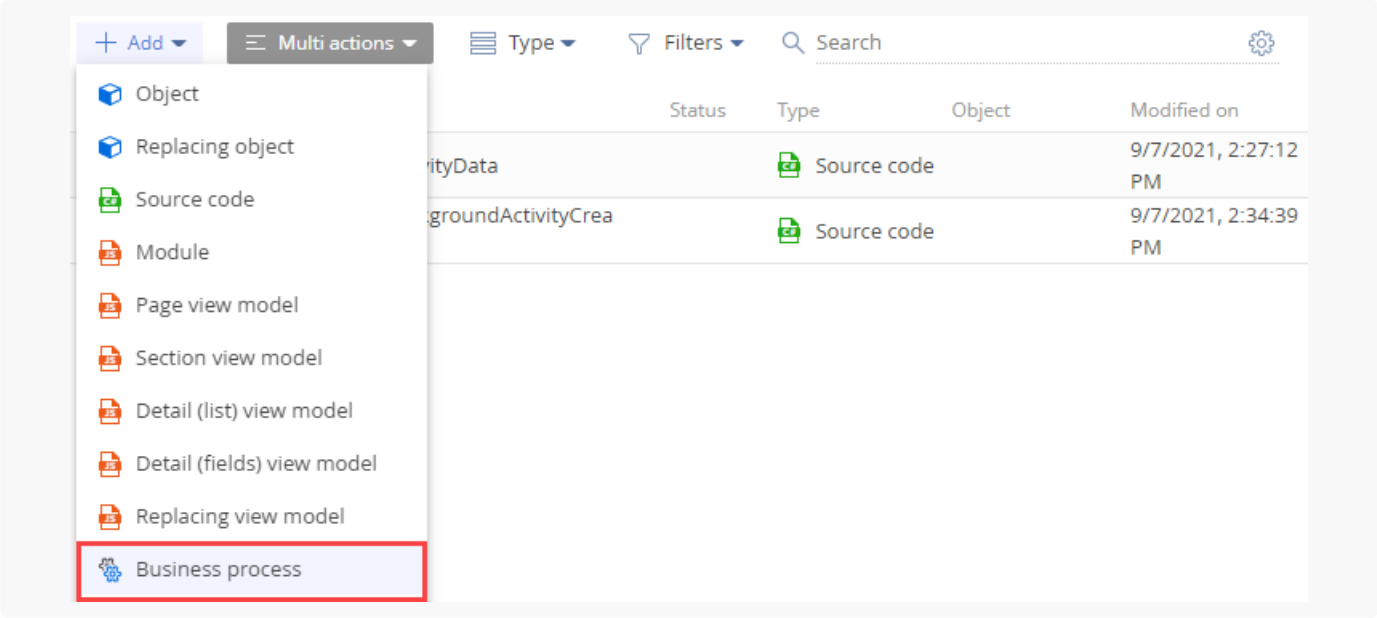
```
        private UserConnection _userConnection;

        /* Implement the Run method of the IBackgroundTask interface. */
        public void Run(UsrActivityData data) {
            /* Forced 30-second delay. */
            System.Threading.Tasks.Task.Delay(TimeSpan.FromSeconds(30));
            /* Creating activity. */
            var activity = new Activity(_userConnection){
                UseAdminRights = false,
                Id = Guid.NewGuid(),
                TypeId = data.TypeId,
                Title = data.Title,

                /* Activity category is "To do". */
                ActivityCategoryId = new Guid("F51C4643-58E6-DF11-971B-001D60E938C6")
            };
            activity.SetDefColumnValues();
            activity.Save(false);
        }

        /* Implement the SetUserConnection method of the IUserConnectionRequired interface. *
        public void SetUserConnection(UserConnection userConnection) {
            _userConnection = userConnection;
        }
    }
}
```

Класс `UsrBackgroundActivityCreator` реализует интерфейсы `IBackgroundTask<UsrActivityData>` и `IUserConnectionRequired`. В методе `Run()` после принудительной задержки в 30 секунд на основе предоставленного экземпляра `UsrActivityData` создается экземпляр объекта раздела [*Активности*] ([*Activities*]).

5. На панели инструментов дизайнера нажмите [*Сохранить*] ([*Save*]), а затем [*Опубликовать*] ([*Publish*]).

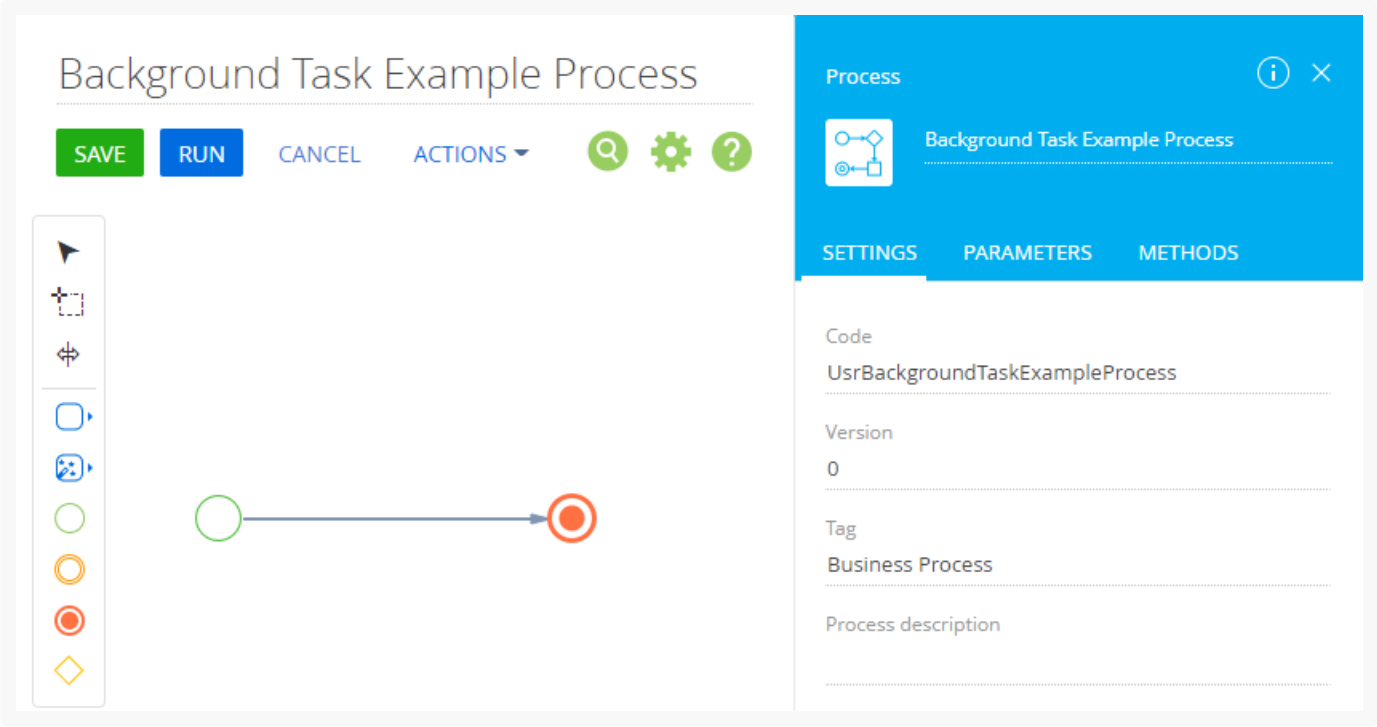
3. Создать бизнес-процесс для запуска фоновой операции

1. Перейдите в раздел [*Конфигурация*] ([*Configuration*]) и выберите пользовательский пакет, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Бизнес процесс*] ([*Add*] —> [*Business process*]).



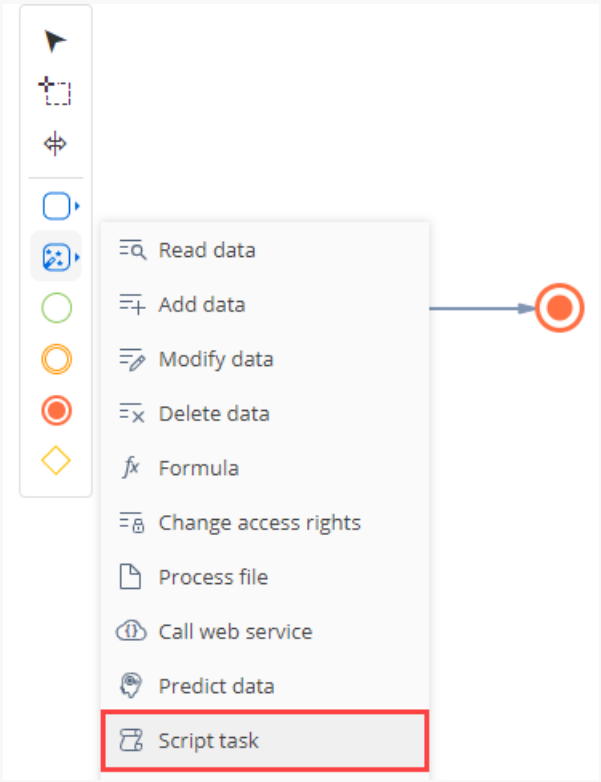
3. В дизайнере процессов заполните свойства процесса:

   - На панели настройки элементов заполните свойство [*Заголовок*] ([*Title*]) — "`Background Task Example Process`".
   - На вкладке [*Настройки*] ([*Settings*]) панели настройки элементов заполните свойство [*Имя*] ([*Code*]) — "`UsrBackgroundTaskExampleProcess`".

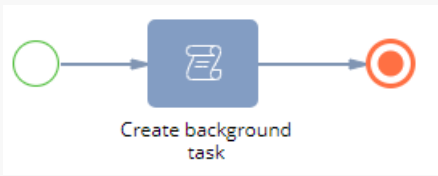


4. Реализуйте бизнес-процесс.

   a. В области элементов дизайнера нажмите [*Действия системы*] ([*System actions*] и разместите элемент [*Задание-сценарий*] ([*Script task*]) в рабочей области дизайнера процессов между начальным событием [*Простое*] ([*Simple*]) и завершающим событием [*Останов*] ([*Terminate*]).



b. Элементу [*Задание-сценарий*] ([*Script task*]) добавьте имя "`Создать фоновую операцию`" ("`Create background task`").

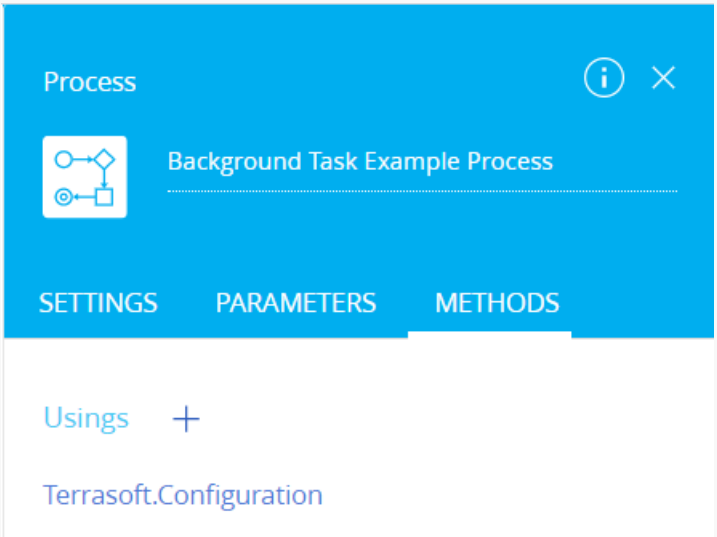


c. Добавьте код элемента [*Задание-сценарий*] ([*Script task*]).

Код элемента [*Задание-сценарий*] ([*Script task*])

```
var data = new UsrActivityData {
    Title = "Activity created by background task",
    TypeId = ActivityConsts.TaskTypeUId
};
Terrasoft.Core.Tasks.Task.StartNewWithUserConnection<UsrBackgroundActivityCreator, UsrActi

return true;
```

d. В дизайнере процессов на вкладке [*Методы*] ([*Methods*]) в блоке [*Usings*] нажмите кнопку + и добавьте пространство имен `Terrasoft.Configuration`. Это необходимо для использования в бизнес-процессе реализации класса для объекта активности и класса для добавления активности.

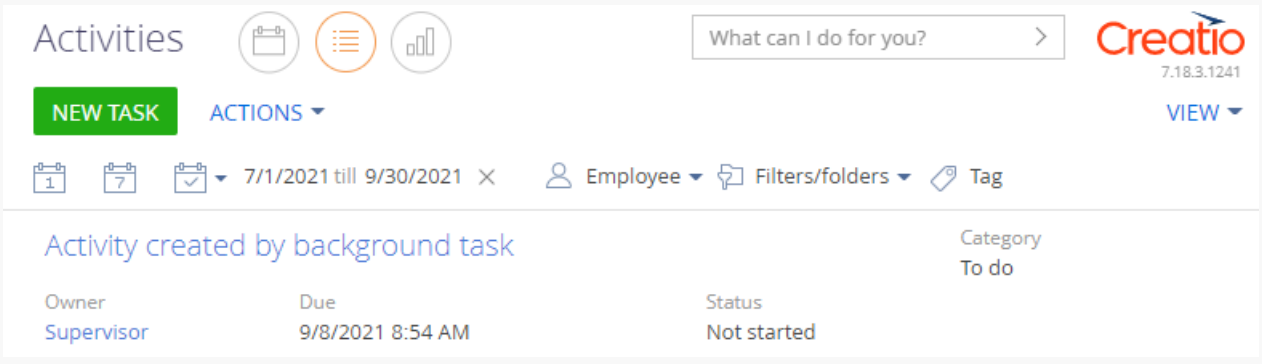


5. На панели инструментов дизайнера нажмите [*Сохранить*] ([*Save*]).
6. Во всплывающем окне нажмите [*Опубликовать*] ([*Publish*]) для компиляции кода элемента [*Задание-сценарий*] ([*Script task*]).

Результат выполнения примера

Чтобы **запустить бизнес-процесс** `Background Task Example Process`, на панели инструментов дизайнера процессов нажмите [*Запустить*] ([*Run*]).

В результате выполнения бизнес-процесса `Background Task Example Process` в списочном представлении реестра раздела [*Активности*] ([*Activities*]) добавляется запись [*Activity created by background task*].



Прямой доступ к данным

Средний

Способы доступа к базе данных, которые предоставляют back-end компоненты ядра:

- Доступ через ORM-модель.
- Прямой доступ.

Для доступа к данным рекомендуется использовать ORM-модель, хотя прямой доступ к базе данных также реализован в back-end компонентах ядра. Выполнение запросов к базе данных через ORM-модель подробно описано в статье Доступ к данным через ORM.

В этой статье будет рассмотрено выполнение запросов к базе данных через прямой доступ.

Классы, которые реализуют работу с данными через прямой доступ:

- `Terrasoft.Core.DB.Select` — построение запросов на получение записей из таблиц базы данных.
- `Terrasoft.Core.DB.Insert` — построение запросов на добавление записей в таблицы базы данных.
- `Terrasoft.Core.DB.InsertSelect` — построение запросов на добавление записей в таблицу базы данных на основании данных, полученных в запросах на получение записей из таблицы базы данных.
- `Terrasoft.Core.DB.Update` — построение запросов на изменение записей в таблице базы данных.
- `Terrasoft.Core.DB.UpdateSelect` — построение запросов на изменение записей в таблице базы данных на основании данных, полученных в запросах на получение записей из таблицы базы данных.
- `Terrasoft.Core.DB.Delete` — построение запросов на удаление записей в таблице базы данных.
- `Terrasoft.Core.DB.DBExecutor` — построение и выполнение сложных запросов (например, с несколькими вложенными фильтрациями, различными комбинациями join-ов и т. д.) к базе данных.

Получить данные из базы данных

Классы, которые реализуют получение данных из базы данных:

- `Terrasoft.Core.DB.Select` — получение данных из базы данных через прямой доступ.
- `Terrasoft.Core.Entities.EntitySchemaQuery` — получение данных из базы данных через ORM-модель. Выполнение запросов к базе данных с использованием класса `Terrasoft.Core.Entities.EntitySchemaQuery` подробно описано в статье Доступ к данным через ORM.

Назначение класса `Terrasoft.Core.DB.Select` — построение запросов на выборку записей из таблиц базы данных. После создания и конфигурирования экземпляра класса будет построен `SELECT`-запрос к базе данных приложения. В запрос можно добавить колонки, фильтры и условия ограничений.

Особенности класса `Terrasoft.Core.DB.Select`:

- В результирующем запросе не учитываются права доступа текущего пользователя. Пользователь получает доступ ко всем таблицам и записям базы данных.
- В результирующем запросе не учитываются данные из хранилища кэша.

Результат выполнения запроса — экземпляр, реализующий интерфейс `System.Data.IDataReader`, или скалярное значение соответствующего типа.

При необходимости построения запросов на выборку записей из базы данных с учетом прав доступа пользователя и данных из хранилища кэша, используйте класс `Terrasoft.Core.Entities.EntitySchemaQuery`.

Добавить данные в базу данных

Классы, которые реализуют добавление данных в базу данных:

- `Terrasoft.Core.DB.Insert`.
- `Terrasoft.Core.DB.InsertSelect`.

Массовое добавление данных

Назначение класса `Terrasoft.Core.DB.Insert` — построение запросов на добавление записей в таблицы базы данных. После создания и конфигурирования экземпляра класса будет построен `INSERT`-запрос к базе данных приложения.

Результат выполнения запроса — количество записей, которые были добавлены с помощью запроса.

Класс содержит реализацию функциональности многострочной вставки. Для этого предназначен метод `Values()`. При вызове метода `Values()` все последующие вызовы метода `Set()` попадают в новый экземпляр `ColumnsValues`. Если коллекция `ColumnsValuesCollection` содержит более одного набора данных, то будет построен запрос с несколькими блоками `Values()`.

Пример многострочной вставки

```
new Insert(UserConnection)
.Into("Table")
.Values()
    .Set("Column1", Column.Parameter(1))
    .Set("Column2", Column.Parameter(1))
    .Set("Column3", Column.Parameter(1))
.Values()
    .Set("Column1", Column.Parameter(2))
    .Set("Column2", Column.Parameter(2))
    .Set("Column3", Column.Parameter(2))
.Values()
    .Set("Column1", Column.Parameter(3))
    .Set("Column2", Column.Parameter(3))
    .Set("Column3", Column.Parameter(3))
.Execute();
```

В результате будет сформирован SQL-запрос.

SQL-запрос

```
-- Для MSSQL или PostgreSQL
INSERT INTO [dbo].[Table] (Column1, Column2, Column3)
VALUES (1, 1, 1),
    (2, 2, 2),
    (3, 3, 3)

-- Для Oracle
INSERT ALL
    into Table (column1, column2, column3) values (1, 1, 1)
```

```
    into Table (column1, column2, column3) values (2, 2, 2)
    into Table (column1, column2, column3) values (3, 3, 3)
SELECT * FROM dual
```

Особенности использования многострочного добавления данных:

- Ограничение количества параметров в MS SQL при использовании `Column.Parameter` в выражении `Set()` — 2100 параметров.
- При превышении допустимого количества параметров разбивку запроса на подзапросы должен выполнить разработчик, поскольку класс `Terrasoft.Core.DB.Insert` не предоставляет такую возможность.

Пример

```
IEnumerable<IEnumerable<ImportEntity>> GetImportEntitiesChunks(IEnumerable<ImportEntity> enti
    var entitiesList = entities.ToList();
    var columnsList = keyColumns.ToList();
    var maxParamsPerChunk = Math.Abs(MaxParametersCountPerQueryChunk / columnsList.Count + 1)
    var chunksCount = (int)Math.Ceiling(entitiesList.Count / (double)maxParamsPerChunk);
    return entitiesList.SplitOnParts(chunksCount);
}

var entitiesList = GetImportEntitiesChunks(entities, importColumns);
entitiesList.AsParallel().AsOrdered()
    .ForAll(entitiesBatch => {
        try {
            var insertQuery = GetBufferedImportEntityInsertQuery();
            foreach (var importEntity in entitiesBatch) {
                insertQuery.Values();
                SetBufferedImportEntityInsertColumnValues(importEntity, insertQuery,
                        importColumns);
                insertQuery.Set("ImportSessionId", Column.Parameter(importSessionId));
            }
            insertQuery.Execute();
        } catch (Exception e) {
            //...
        }
});
```

- Валидацию совпадения количества колонок и количества условий `Set()` должен выполнить разработчик, поскольку класс `Terrasoft.Core.DB.Insert` не предоставляет такую возможность. При несовпадении количества возникнет исключение на уровне работы СУБД.

Добавление данных из выборки

Назначение класса `Terrasoft.Core.DB.InsertSelect` — построение запросов на добавление записей в

таблицы базы данных на основании данных, полученных в запросах на получение записей из таблицы базы данных. То есть в качестве источника данных запроса используется экземпляр класса `Terrasoft.Core.DB.Select`. После создания и конфигурирования экземпляра класса будет построен `INSERT INTO SELECT`-запрос к базе данных приложения.

Особенность класса `Terrasoft.Core.DB.InsertSelect` — в результирующем запросе для добавляемых записей не учитываются права доступа текущего пользователя. Пользовательское соединение используется только для доступа к таблице базы данных.

Результат выполнения запроса — добавление в таблицы базы данных записей, которые были получены в `Select`-запросе.

Изменить данные в базе данных

Классы, которые реализуют изменение данных в базе данных:

- `Terrasoft.Core.DB.Update`.
- `Terrasoft.Core.DB.UpdateSelect`.

Массовое изменение данных

Назначение класса `Terrasoft.Core.DB.Update` — построение запросов на изменение записей в таблицах базы данных. После создания и конфигурирования экземпляра класса будет построен `UPDATE`-запрос к базе данных приложения.

Изменение данных на основании выборки

Назначение класса `Terrasoft.Core.DB.UpdateSelect` — построение запросов на изменение записей в таблицах базы данных на основании данных, полученных в запросах на получение записей из таблицы базы данных. То есть в качестве источника данных запроса используется экземпляр класса `Terrasoft.Core.DB.Select`. После создания и конфигурирования экземпляра класса будет построен `UPDATE FROM`-запрос к базе данных приложения.

Результат выполнения запроса — изменение в таблице базы данных записей, которые были получены в `Select`-запросе.

Удалить данные из базы данных

`Terrasoft.Core.DB.Delete` — класс, который реализует удаление данных из базы данных.

Назначение класса `Terrasoft.Core.DB.Delete` — построение запросов на удаление записей из таблиц базы данных. После создания и конфигурирования экземпляра класса будет построен `DELETE`-запрос к базе данных приложения.

Использовать многопоточность при работе с базой данных

Многопоточность — использование нескольких параллельных потоков при отправке запросов к базе

данных через `UserConnection`.

`Terrasoft.Core.DB.DBExecutor` — класс, который позволяет использовать многопоточность. Реализует построение и выполнение нескольких запросов к базе данных в одной транзакции. Одному пользователю доступен только один экземпляр `DBExecutor`. Пользователь не имеет возможности создавать новые экземпляры.

Использование многопоточности может привести к проблемам синхронизации старта и подтверждения транзакций. Проблема возникает, даже если `DBExecutor` не используется напрямую, а используется, например, через `EntitySchemaQuery`.

Особенность класса `Terrasoft.Core.DB.DBExecutor` — создание экземпляра `DBExecutor` необходимо оборачивать в оператор `using`. Это связано с тем, что для работы с базой данных используются неуправляемые (`unmanaged`) ресурсы. Также для освобождения ресурсов можно явно вызвать метод `Dispose()`. Использование оператора `using` подробно описано в официальной документации Microsoft.

Транзакция начинается вызовом метода `dbExecutor.StartTransaction` и заканчивается вызовом `dbExecutor.CommitTransaction` или `dbExecutor.RollbackTransaction`. Если выполнение вышло за область видимости блока `using` и не был вызван метод `dbExecutor.CommitTransaction`, происходит автоматический откат транзакции.

Важно. При выполнении нескольких запросов в одной транзакции необходимо передавать `dbExecutor` в методы `Execute`, `ExecuteReader`, `ExecuteScalar`.

Ниже представлены фрагменты исходного кода с использованием `DBExecutor`. Нельзя выполнять вызов методов экземпляра `DBExecutor` в параллельных потоках.

Пример правильного использования DBExecutor

```
/* Первое использование экземпляра DBExecutor в основном потоке. */
using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection()) {
    dbExecutor.StartTransaction();
    //...
    dbExecutor.CommitTransaction();
}
//...
var select = (Select)new Select(UserConnection)
    .Column("Id")
    .From("Contact")
    .Where("Name")
    .IsEqual(Column.Parameter("Supervisor"));
/* Повторное использование экземпляра DBExecutor в основном потоке. */
using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection()) {
    using (IDataReader dataReader = select.ExecuteReader(dbExecutor)) {
        while (dataReader.Read()) {
            //...
        }
    }
```

```
}
```

Пример неправильного использования DBExecutor

```
/* Создание параллельного потока. */
var task = new Task(() => {

    /* Использование экземпляра DBExecutor в параллельном потоке. */
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection()) {
        dbExecutor.StartTransaction();
        //...
        dbExecutor.CommitTransaction();
    }
});

/* Запуск асинхронной задачи в параллельном потоке. Выполнение программы в основном потоке продо
task.Start();
//...
var select = (Select)new Select(UserConnection)
    .Column("Id")
    .From("Contact")
    .Where("Name")
    .IsEqual(Column.Parameter("Supervisor"));

/* Использование экземпляра DBExecutor в основном потоке приведет к возникновению ошибки, поскол
using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection()) {
    using (IDataReader dataReader = select.ExecuteReader(dbExecutor)) {
        while (dataReader.Read()) {
            //...
        }
    }
}
```

Получить данные из базы данных

 Сложный

На заметку. Примеры, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

Ниже приведен метод `CreateJson`, который используется в примерах для обработки результата запросов.

Метод `CreateJson`

```
private string CreateJson(IDataReader dataReader)
{
    var list = new List<dynamic>();
    var cnt = dataReader.FieldCount;
    var fields = new List<string>();
    for (int i = 0; i < cnt; i++)
    {
        fields.Add(dataReader.GetName(i));
    }
    while (dataReader.Read())
    {
        dynamic exo = new System.Dynamic.ExpandoObject();
        foreach (var field in fields)
        {
            ((IDictionary<String, Object>)exo).Add(field, dataReader.GetColumnValue(field));
        }
        list.Add(exo);
    }
    return JsonConvert.SerializeObject(list);
}
```

Пример 1

Пример. Выбрать определенное количество записей из требуемой таблицы (схемы объекта).

Метод `SelectColumns`

```
public string SelectColumns(string tableName, int top)
{
    top = top > 0 ? top : 1;
    var result = "{}";
    var select = new Select(UserConnection)
        .Top(top)
        .Column(Column.Asterisk())
        .From(tableName);
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection())
    {
        using (IDataReader dataReader = select.ExecuteReader(dbExecutor))
        {
            result = CreateJson(dataReader);
        }
```

```
    }
    return result;
}
```

Пример 2

Пример. Выбрать идентификатор, имя и дату рождения контактов, дата рождения которых позже требуемого года.

Метод `SelectContactsYoungerThan`

```
public string SelectContactsYoungerThan(string birthYear)
{
    var result = "{}";
    var year = DateTime.ParseExact(birthYear, "yyyy", CultureInfo.InvariantCulture);
    var select = new Select(UserConnection)
            .Column("Id")
            .Column("Name")
            .Column("BirthDate")
        .From("Contact")
        .Where("BirthDate").IsGreater(Column.Parameter(year))
            .Or("BirthDate").IsNull()
        .OrderByDesc("BirthDate")
            as Select;
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection())
    {
        using (IDataReader dataReader = select.ExecuteReader(dbExecutor))
        {
            result = CreateJson(dataReader);
        }
    }
    return result;
}
```

Пример 3

Пример. Выбрать идентификатор, имя и дату рождения контактов, дата рождения которых позже заданного года и у которых указан контрагент.

Метод `SelectContactsYoungerThanAndHasAccountId`

```
public string SelectContactsYoungerThanAndHasAccountId(string birthYear)
{
    var result = "{}";
    var year = DateTime.ParseExact(birthYear, "yyyy", CultureInfo.InvariantCulture);
    var select = new Select(UserConnection)
            .Column("Id")
            .Column("Name")
            .Column("BirthDate")
        .From("Contact")
        .Where()
        .OpenBlock("BirthDate").IsGreater(Column.Parameter(year))
            .Or("BirthDate").IsNull()
        .CloseBlock()
        .And("AccountId").Not().IsNull()
        .OrderByDesc("BirthDate")
            as Select;
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection())
    {
        using (IDataReader dataReader = select.ExecuteReader(dbExecutor))
        {
            result = CreateJson(dataReader);
        }
    }
    return result;
}
```

Пример 4

Пример. Выбрать идентификатор и имя всех контактов, присоединив к ним идентификаторы и названия соответствующих контрагентов.

Метод `SelectContactsJoinAccount`

```
public string SelectContactsJoinAccount()
{
    var result = "{}";
    var select = new Select(UserConnection)
            .Column("Contact", "Id").As("ContactId")
            .Column("Contact", "Name").As("ContactName")
            .Column("Account", "Id").As("AccountId")
            .Column("Account", "Name").As("AccountName")
        .From("Contact")
        .Join(JoinType.Inner, "Account")
```

```
            .On("Contact", "Id").IsEqual("Account", "PrimaryContactId")
            as Select;
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection())
    {
        using (IDataReader dataReader = select.ExecuteReader(dbExecutor))
        {
            result = CreateJson(dataReader);
        }
    }
    return result;
}
```

Пример 5

Пример. Выбрать идентификатор и имя контактов, являющихся основными для контрагентов.

Метод `SelectAccountPrimaryContacts`

```
public string SelectAccountPrimaryContacts()
{
    var result = "{}";
    var select = new Select(UserConnection)
            .Column("Id")
            .Column("Name")
        .From("Contact").As("C")
        .Where()
            .Exists(new Select(UserConnection)
                            .Column("A", "PrimaryContactId")
                        .From("Account").As("A")
                        .Where("A", "PrimaryContactId").IsEqual("C", "Id"))
            as Select;
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection())
    {
        using (IDataReader dataReader = select.ExecuteReader(dbExecutor))
        {
            result = CreateJson(dataReader);
        }
    }
    return result;
}
```

Пример 6

Пример. Выбрать страны и количество городов в стране, если количество городов больше указанного.

Метод `SelectCountriesWithCitiesCount`

```
public string SelectCountriesWithCitiesCount(int count)
{
    var result = "{}";
    var select = new Select(UserConnection)
            .Column(Func.Count("City", "Id")).As("CitiesCount")
            .Column("Country", "Name").As("CountryName")
        .From("City")
        .Join(JoinType.Inner, "Country")
            .On("City", "CountryId").IsEqual("Country", "Id")
        .GroupBy("Country", "Name")
        .Having(Func.Count("City", "Id")).IsGreater(Column.Parameter(count))
        .OrderByDesc("CitiesCount")
            as Select;
    using (DBExecutor dbExecutor = UserConnection.EnsureDBConnection())
    {
        using (IDataReader dataReader = select.ExecuteReader(dbExecutor))
        {
            result = CreateJson(dataReader);
        }
    }
    return result;
}
```

Пример 7

Пример. Получить идентификатор контакта по его имени.

Метод `SelectCountryIdByCityName`

```
public string SelectCountryIdByCityName(string CityName)
{
    var result = "";
    var select = new Select(UserConnection)
            .Column("CountryId")
        .From("City")
        .Where("Name").IsEqual(Column.Parameter(CityName)) as Select;
    result = select.ExecuteScalar<Guid>().ToString();
```

```
    return result;
}
```

Добавить данные в базу данных

Сложный

На заметку. Примеры, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

Пример 1

Пример. Добавить контакт с указанным именем.

Метод `InsertContact`

```
public string InsertContact(string contactName)
{
    contactName = contactName ?? "Unknown contact";
    var ins = new Insert(UserConnection)
        .Into("Contact")
        .Set("Name", Column.Parameter(contactName));
    var affectedRows = ins.Execute();
    var result = $"Inserted new contact with name '{contactName}'. {affectedRows} rows affected"
    return result;
}
```

Пример 2

Пример. Добавить город с указанным названием, привязав его к указанной стране.

Метод `InsertCity`

```
public string InsertCity(string city, string country)
{
    city = city ?? "unknown city";
    country = country ?? "unknown country";
```

```
    var ins = new Insert(UserConnection)
        .Into("City")
        .Set("Name", Column.Parameter(city))
        .Set("CountryId",
            new Select(UserConnection)
                    .Top(1)
                    .Column("Id")
                .From("Country")
                .Where("Name")
                    .IsEqual(Column.Parameter(country)));
    var affectedRows = ins.Execute();
    var result = $"Inserted new city with name '{city}' located in '{country}'. {affectedRows} r
    return result;
}
```

Добавить данные в базу данных с помощью подзапросов

Сложный

На заметку. Примеры, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

Пример 1

Пример. Добавить контакт с указанными именем и названием контрагента.

Метод `InsertContactWithAccount`

```
public string InsertContactWithAccount(string contactName, string accountName)
{
    contactName = contactName ?? "Unknown contact";
    accountName = accountName ?? "Unknown account";

    var id = Guid.NewGuid();
    var selectQuery = new Select(UserConnection)
            .Column(Column.Parameter(contactName))
            .Column("Id")
        .From("Account")
        .Where("Name").IsEqual(Column.Parameter(accountName)) as Select;
    var insertSelectQuery = new InsertSelect(UserConnection)
```

```
        .Into("Contact")
        .Set("Name", "AccountId")
        .FromSelect(selectQuery);

    var affectedRows = insertSelectQuery.Execute();
    var result = $"Inserted new contact with name '{contactName}'" +
                 $" and account '{accountName}'." +
                 $" Affected {affectedRows} rows.";
    return result;
}
```

Пример 2

Пример. Добавить контакт с указанным именем, связав его со всеми контрагентами.

Метод `InsertAllAccountsContact`

```
public string InsertAllAccountsContact(string contactName)
{
    contactName = contactName ?? "Unknown contact";

    var id = Guid.NewGuid();
    var insertSelectQuery = new InsertSelect(UserConnection)
        .Into("Contact")
        .Set("Name", "AccountId")
        .FromSelect(
            new Select(UserConnection)
                .Column(Column.Parameter(contactName))
                .Column("Id")
                .From("Account") as Select);

    var affectedRows = insertSelectQuery.Execute();
    var result = $"Inserted {affectedRows} new contacts with name '{contactName}'";
    return result;
}
```

Изменить данные в базе данных

Сложный

На заметку. Примеры, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

В большинстве случаев запрос на изменение данных должен содержать условие `Where`, которое уточняет какие именно записи необходимо изменить. Если не указать условие `Where`, то будут изменены все записи.

Пример 1

Пример. Изменить имя контакта.

Метод `ChangeContactName`

```
public string ChangeContactName(string oldName, string newName)
{
    var update = new Update(UserConnection, "Contact")
        .Set("Name", Column.Parameter(newName))
        .Where ("Name").IsEqual(Column.Parameter(oldName));
    var cnt = update.Execute();
    return $"Contacts {oldName} changed to {newName}. {cnt} rows affected.";
}
```

Пример 2

Пример. Для всех существующих контактов поменять пользователя, изменившего запись, на указанного.

Метод `ChangeAllContactModifiedBy`

```
public string ChangeAllContactModifiedBy(string Name)
{
    var update = new Update(UserConnection, "Contact")
        .Set("ModifiedById",
            new Select(UserConnection).Top(1)
                .Column("Id")
                .From("Contact")
                .Where("Name").IsEqual(Column.Parameter(Name)));
    var cnt = update.Execute();
    return $"All contacts are changed by {Name} now. {cnt} rows affected.";
}
```

Условие `Where` относится к запросу `Select`. Запрос `Update` не содержит условия `Where`, поскольку необходимо изменить все записи.

Удалить данные из базы данных

Сложный

На заметку. Примеры, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

В большинстве случаев запрос на удаление данных должен содержать условие `Where`, которое уточняет какие именно записи необходимо удалить. Если не указать условие `Where`, то будут удалены все записи.

Пример

Пример. Удалить контакт с указанным именем.

Метод `DeleteContacts`

```
public string DeleteContacts(string name)
{
    var delete = new Delete(UserConnection)
        .From("Contact")
        .Where("Name").IsEqual(Column.Parameter(name));
    var cnt = delete.Execute();
    return $"Contacts with name {name} were deleted. {cnt} rows affected";
}
```

Класс Select C#

Сложный

Пространство имен `Terrasoft.Core.DB`.

Класс `Terrasoft.Core.DB.Select` предназначен для построения запросов выборки записей из таблиц базы данных. В результате создания и конфигурирования экземпляра этого класса будет построен запрос в базу данных приложения в виде SQL-выражения `SELECT`. В запрос можно добавить требуемые колонки, фильтры и условия ограничений. Результаты выполнения запроса возвращаются в виде экземпляра, реализующего интерфейс `System.Data.IDataReader`, либо скалярного значения требуемого типа.

Важно. При работе с классом `Select` не учитываются права доступа пользователя, использующего

текущее соединение. Доступны абсолютно все записи из базы данных приложения. Также не учитываются данные, помещенные в хранилище кэша. Если необходимы дополнительные возможности по управлению правами доступа и работе с хранилищем кэша Creatio, следует использовать класс `EntitySchemaQuery`.

На заметку. Полный перечень методов и свойств класса `Select`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`Select(UserConnection userConnection)`

Создает экземпляр класса с указанным `UserConnection`.

`Select(UserConnection userConnection, CancellationToken cancellationToken)`

Создает экземпляр класса с указанным `UserConnection` и токеном отмены выполнения управляемого потока.

`Select(Select source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`UserConnection` `Terrasoft.Core.UserConnection`

Пользовательское подключение, используемое при выполнении запроса.

`RowCount` `int`

Количество записей, которые вернет запрос после выполнения.

`Parameters` `Terrasoft.Core.DB.QueryParameterCollection`

Коллекция параметров запроса.

`HasParameters` `bool`

Определяет наличие параметров у запроса.

`BuildParametersAsValue` `bool`

Определяет, добавлять ли параметры запроса в текст запроса как значения.

`Joins` `Terrasoft.Core.DB.JoinCollection`

Коллекция выражений `Join` в запросе.

`HasJoins` `bool`

Определяет наличие выражений `Join` в запросе.

`Condition` `Terrasoft.Core.DB.QueryCondition`

Условие выражения `Where` запроса.

`HasCondition` `bool`

Определяет наличие выражения `Where` в запросе.

`HavingCondition` `Terrasoft.Core.DB.QueryCondition`

Условие выражения `Having` запроса.

`HasHavingCondition` `bool`

Определяет наличие выражения `Having` в запросе.

`OrderByItems` `Terrasoft.Core.DB.OrderByItemCollection`

Коллекция выражений, по которым выполняется сортировка результатов запроса.

`HasOrderByItems` `bool`

Определяет наличие условий сортировки результатов запроса.

`GroupByItems` `Terrasoft.Core.DB.QueryColumnExpressionCollection`

Коллекция выражений, по которым выполняется группировка результатов запроса.

`HasGroupByItems` `bool`

Определяет наличие условий группировки результатов запроса.

`IsDistinct` `bool`

Определяет, должен ли запрос возвращать только уникальные записи.

`Columns` `Terrasoft.Core.DB.QueryColumnExpressionCollection`

Коллекция выражений колонок запроса.

`OffsetFetchPaging` `bool`

Определяет возможность постраничного возврата результата запроса.

`RowsOffset` `int`

Количество строк, которые необходимо пропустить при возврате результата запроса.

`QueryKind` `Terrasoft.Common.QueryKind`

Тип запроса (см. статью Настройка отдельного пула запросов).

Методы

`void ResetCachedSqlText()`

Очищает закэшированный текст запроса.

`QueryParameterCollection GetUsingParameters()`

Возвращает коллекцию параметров, используемых запросом.

`void ResetParameters()`

Очищает коллекцию параметров запроса.

`QueryParameterCollection InitializeParameters()`

Инициализирует коллекцию параметров запроса.

`IDataReader ExecuteReader(DBExecutor dbExecutor)`

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает объект, реализующий интерфейс `IDataReader`.

Параметры

dbExecutor	Экземпляр `DBExecutor`, который используется для выполнения запроса.

`IDataReader ExecuteReader(DBExecutor dbExecutor, CommandBehavior behavior)`

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает объект, реализующий интерфейс `IDataReader`.

Параметры

behavior	Предоставляет описание результатов запроса и их эффект на базу данных.
dbExecutor	Экземпляр `DBExecutor`, который используется для выполнения запроса.

`void ExecuteReader(ExecuteReaderReadMethod readMethod)`

Выполняет запрос, вызывая переданный метод делегата `ExecuteReaderReadMethod` для каждой записи результирующего набора.

Параметры

readMethod	Метод делегата `ExecuteReaderReadMethod`.

`TResult ExecuteScalar<TResult>()`

Выполняет запрос. Возвращает типизированный первый столбец первой записи результирующего набора.

`TResult ExecuteScalar<TResult>(DBExecutor dbExecutor)`

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает типизированный первый столбец первой записи результирующего набора.

Параметры

dbExecutor	Экземпляр `DBExecutor`, который используется для выполнения запроса.

`Select Distinct()`

Добавляет к SQL-запросу ключевое слово `DISTINCT`. Исключает дублирование записей в результирующем наборе. Возвращает экземпляр запроса.

`Select Top(int rowCount)`

Устанавливает количество записей, возвращаемых в результирующем наборе. При этом меняется значение свойства `RowCount`. Возвращает экземпляр запроса.

Параметры

`rowCount`	Количество первых записей результирующего набора.

`Select As(string alias)`

Добавляет указанный в аргументе псевдоним для последнего выражения запроса. Возвращает экземпляр запроса.

Параметры

`alias`	Псевдоним выражения запроса.

```
Select Column(string sourceColumnAlias)
Select Column(string sourceAlias, string sourceColumnAlias)
Select Column(Select subSelect)
Select Column(Query subSelectQuery)
Select Column(QueryCase queryCase)
Select Column(QueryParameter queryParameter)
Select Column(QueryColumnExpression columnExpression)
```

Добавляет выражение, подзапрос или параметр в коллекцию выражений колонок запроса. Возвращает экземпляр запроса.

Параметры

`sourceColumnAlias`	Псевдоним колонки, для которой добавляется выражение.
`sourceAlias`	Псевдоним источника, из которого добавляется выражение колонки.
`subSelect`	Добавляемый подзапрос выборки данных.
`subSelectQuery`	Добавляемый подзапрос.
`queryCase`	Добавляемое выражение для оператора `Case`.
`queryParameter`	Добавляемый параметр запроса.
`columnExpression`	Выражение, для результатов которого добавляется условие.

```
Select From(string schemaName)
Select From(Select subSelect)
Select From(Query subSelectQuery)
Select From(QuerySourceExpression sourceExpression)
```

Добавляет в запрос источник данных. Возвращает экземпляр запроса.

Параметры

`schemaName`	Название схемы.
`subSelect`	Подзапрос выборки, результаты которого становятся источником данных для текущего запроса.
`subSelectQuery`	Подзапрос, результаты которого становятся источником данных для текущего запроса.
`sourceExpression`	Выражение источника данных запроса.

```
Join Join(JoinType joinType, string schemaName)
Join Join(JoinType joinType, Select subSelect)
Join Join(JoinType joinType, Query subSelectQuery)
Join Join(JoinType joinType, QuerySourceExpression sourceExpression)
```

Присоединяет к текущему запросу схему, подзапрос или выражение.

Параметры

`joinType`	Тип присоединения.
`schemaName`	Название присоединяемой схемы.
`subSelect`	Присоединяемый подзапрос выборки данных.
`subSelectQuery`	Присоединяемый подзапрос.
`sourceExpression`	Присоединяемое выражение.

Возможные значения (`Terrasoft.Core.DB.JoinType`)

`Inner`	Внутреннее соединение.
`LeftOuter`	Левое внешнее соединение.
`RightOuter`	Правое внешнее соединение.
`FullOuter`	Полное соединение.
`Cross`	Перекрестное соединение.

```
QueryCondition Where()
QueryCondition Where(string sourceColumnAlias)
QueryCondition Where(string sourceAlias, string sourceColumnAlias)
QueryCondition Where(Select subSelect)
QueryCondition Where(Query subSelectQuery)
QueryCondition Where(QueryColumnExpression columnExpression)
Query Where(QueryCondition condition)
```

Добавляет к текущему запросу начальное условие.

Параметры

sourceColumnAlias	Псевдоним колонки, для которой добавляется условие.
sourceAlias	Псевдоним источника.
subSelect	Подзапрос выборки данных, для результатов которого добавляется условие.
subSelectQuery	Подзапрос, для результатов которого добавляется условие.
columnExpression	Выражение, для результатов которого добавляется условие.
condition	Условие запроса.

```
QueryCondition And()
QueryCondition And(string sourceColumnAlias)
QueryCondition And(string sourceAlias, string sourceColumnAlias)
QueryCondition And(Select subSelect)
QueryCondition And(Query subSelectQuery)
QueryCondition And(QueryParameter parameter)
QueryCondition And(QueryColumnExpression columnExpression)
Query And(QueryCondition condition)
```

К текущему условию запроса добавляет условие (предикат), используя логическую операцию И.

Параметры

sourceColumnAlias	Псевдоним колонки, для которой добавляется предикат.
sourceAlias	Псевдоним источника.
subSelect	Подзапрос выборки данных, используемый в качестве предиката.
subSelectQuery	Подзапрос, используемый в качестве предиката.
parameter	Параметр, для которого добавляется предикат.
columnExpression	Выражение, используемое в качестве предиката.
condition	Условие запроса.

```
QueryCondition Or()
QueryCondition Or(string sourceColumnAlias)
QueryCondition Or(string sourceAlias, string sourceColumnAlias)
```

```
QueryCondition Or(Select subSelect)
QueryCondition Or(Query subSelectQuery)
QueryCondition Or(QueryParameter parameter)
QueryCondition Or(QueryColumnExpression columnExpression)
Query Or(QueryCondition condition)
```

К текущему условию запроса добавляет условие (предикат), используя логическую операцию ИЛИ.

Параметры

`sourceColumnAlias`	Псевдоним колонки, для которой добавляется предикат.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос на выборку данных, используемый в качестве предиката.
`subSelectQuery`	Подзапрос, используемый в качестве предиката.
`parameter`	Параметр, для которого добавляется предикат.
`columnExpression`	Выражение, используемое в качестве предиката.
`condition`	Условие запроса.

```
Query OrderBy(OrderDirectionStrict direction, string sourceColumnAlias)
Query OrderBy(OrderDirectionStrict direction, string sourceAlias, string sourceColumnAlias)
Query OrderBy(OrderDirectionStrict direction, QueryFunction queryFunction)
Query OrderBy(OrderDirectionStrict direction, Select subSelect)
Query OrderBy(OrderDirectionStrict direction, Query subSelectQuery)
Query OrderBy(OrderDirectionStrict direction, QueryColumnExpression columnExpression)
```

Выполняет сортировку результатов запроса. Возвращает экземпляр запроса.

Параметры

direction	Порядок сортировки.
sourceColumnAlias	Псевдоним колонки, по которой выполняется сортировка.
sourceAlias	Псевдоним источника.
queryFunction	Функция, значение которой используется в качестве ключа сортировки.
subSelect	Подзапрос на выборку данных, результаты которого используются в качестве ключа сортировки.
subSelectQuery	Подзапрос, результаты которого используются в качестве ключа сортировки.
columnExpression	Выражение, результаты которого используются в качестве ключа сортировки.

```
Query OrderByAsc(string sourceColumnAlias)
Query OrderByAsc(string sourceAlias, string sourceColumnAlias)
Query OrderByAsc(Select subSelect)
Query OrderByAsc(Query subSelectQuery)
Query OrderByAsc(QueryColumnExpression columnExpression)
```

Выполняет сортировку результатов запроса в порядке возрастания. Возвращает экземпляр запроса.

Параметры

sourceColumnAlias	Псевдоним колонки, по которой выполняется сортировка.
sourceAlias	Псевдоним источника.
subSelect	Подзапрос на выборку данных, результаты которого используются в качестве ключа сортировки.
subSelectQuery	Подзапрос, результаты которого используются в качестве ключа сортировки.
columnExpression	Выражение, результаты которого используются в качестве ключа сортировки.

```
Query OrderByDesc(string sourceColumnAlias)
Query OrderByDesc(string sourceAlias, string sourceColumnAlias)
Query OrderByDesc(Select subSelect)
```

```
Query OrderByDesc(Query subSelectQuery)
Query OrderByDesc(QueryColumnExpression columnExpression)
```

Выполняет сортировку результатов запроса в порядке убывания. Возвращает экземпляр запроса.

Параметры

`sourceColumnAlias`	Псевдоним колонки, по которой выполняется сортировка.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос выборки, результаты которого используются в качестве ключа сортировки.
`subSelectQuery`	Подзапрос, результаты которого используются в качестве ключа сортировки.
`columnExpression`	Выражение, результаты которого используются в качестве ключа сортировки.

```
Query GroupBy(string sourceColumnAlias)
Query GroupBy(string sourceAlias, string sourceColumnAlias)
Query GroupBy(QueryColumnExpression columnExpression)
```

Выполняет группировку результатов запроса. Возвращает экземпляр запроса.

Параметры

`sourceColumnAlias`	Псевдоним колонки, по которой выполняется группировка.
`sourceAlias`	Псевдоним источника.
`columnExpression`	Выражение, результаты которого используются в качестве ключа группировки.

```
QueryCondition Having()
QueryCondition Having(string sourceColumnAlias)
QueryCondition Having(string sourceAlias, string sourceColumnAlias)
QueryCondition Having(Select subSelect)
QueryCondition Having(Query subSelectQuery)
QueryCondition Having(QueryParameter parameter)
QueryCondition Having(QueryColumnExpression columnExpression)
```

Добавляет в текущий запрос групповое условие. Возвращает экземпляр `Terrasoft.Core.DB.QueryCondition`, представляющий групповое условие для параметра запроса.

Параметры

`sourceColumnAlias`	Псевдоним колонки, по которой добавляется групповое условие.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос выборки, для результатов которого добавляется групповое условие.
`subSelectQuery`	Подзапрос, для результатов которого добавляется групповое условие.
`parameter`	Параметр запроса, для которого добавляется групповое условие.
`columnExpression`	Выражение, используемое в качестве предиката.

```
Query Union(Select unionSelect)
Query Union(Query unionSelectQuery)
```

Объединяет результаты переданного запроса с результатами текущего запроса, исключая дубликаты из результирующего набора.

Параметры

`unionSelect`	Подзапрос выборки.
`unionSelectQuery`	Подзапрос.

```
Query UnionAll(Select unionSelect)
Query UnionAll(Query unionSelectQuery)
```

Объединяет результаты переданного запроса с результатами текущего запроса. Дубликаты из результирующего набора не исключаются.

Параметры

`unionSelect`	Подзапрос выборки.
`unionSelectQuery`	Подзапрос.

Класс Insert C#

Сложный

Пространство имен `Terrasoft.Core.DB`.

Класс `Terrasoft.Core.DB.Insert` предназначен для построения запросов на добавление записей в таблицы базы данных Creatio. В результате создания и конфигурирования экземпляра этого класса будет построен запрос в базу данных приложения в виде SQL-выражения `INSERT`. В результате выполнения запроса возвращается количество задействованных запросом записей.

Важно. При работе с классом `Insert` на добавленные записи не применяются права доступа по умолчанию. Пользовательское соединение используется только для доступа к таблице базы данных.

На заметку. Полный перечень методов и свойств класса `Insert`, его родительских классов, а также реализуемых им интерфейсов, можно найти в Библиотеке .NET классов.

Конструкторы

`Entity(UserConnection userConnection)`

Создает новый экземпляр класса `Entity` для заданного пользовательского подключения `UserConnection`.

`Insert(UserConnection userConnection)`

Создает экземпляр класса с указанным `UserConnection`.

`Insert(Insert source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`UserConnection` `Terrasoft.Core.UserConnection`

Пользовательское подключение, используемое при запросе.

`Source` `Terrasoft.Core.DB.ModifyQuerySource`

Источник данных

`Parameters` `Terrasoft.Core.DB.QueryParameterCollection`

Коллекция параметров запроса.

`HasParameters` `bool`

Определяет, имеет ли запрос параметры.

`BuildParametersAsValue` `bool`

Определяет, добавлять ли параметры запроса в текст запроса как значения.

`ColumnValues` `Terrasoft.Core.DB.ModifyQueryColumnValueCollection`

Коллекция значений колонок запроса.

`ColumnValuesCollection` `List<ModifyQueryColumnValueCollection>`

Коллекция значений колонок для множественного добавления записей.

Методы

`void ResetCachedSqlText()`

Очищает кэшированный текст запроса.

`QueryParameterCollection GetUsingParameters()`

Возвращает коллекцию параметров, используемых запросом.

`void ResetParameters()`

Очищает коллекцию параметров запроса.

`void SetParameterValue(string name, object value)`

Устанавливает значение для параметра запроса.

Параметры

`name`	Название параметра.
`value`	Значение.

```
void InitializeParameters()
```

Инициализирует коллекцию параметров запроса.

```
int Execute()
```

Выполняет запрос. Возвращает количество задействованных запросом записей.

```
int Execute(DBExecutor dbExecutor)
```

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает количество задействованных запросом записей.

```
Insert Into(string schemaName)
Insert Into(ModifyQuerySource source)
```

Добавляет в текущий запрос источник данных.

Параметры

`schemaName`	Название схемы.
`source`	Источник данных.

```
Insert Set(string sourceColumnAlias, Select subSelect)
Insert Set(string sourceColumnAlias, Query subSelectQuery)
Insert Set(string sourceColumnAlias, QueryColumnExpression columnExpression)
Insert Set(string sourceColumnAlias, QueryParameter parameter)
```

Добавляет в текущий запрос предложение `SET` для присвоения колонке переданного выражения или параметра. Возвращает текущий экземпляр `Insert`.

Параметры

`sourceColumnAlias`	Псевдоним колонки.
`subSelect`	Подзапрос на выборку.
`subSelectQuery`	Подзапрос.
`columnExpression`	Выражение колонки.
`parameter`	Параметр запроса.

`Insert Values()`

Инициализирует значения для множественного добавления записей.

Класс InsertSelect C#

Сложный

Пространство имен `Terrasoft.Core.DB`.

Класс `Terrasoft.Core.DB.InsertSelect` предназначен для построения запросов на добавление записей в таблицы базы данных Creatio. При этом в качестве источника добавляемых данных используется экземпляр класса `Terrasoft.Core.DB.Select`. В результате создания и конфигурирования экземпляра `Terrasoft.Core.DB.InsertSelect` будет построен запрос базу данных приложения в виде SQL-выражения `INSERT INTO SELECT`.

Важно. При работе с классом `InsertSelect` на добавленные записи не применяются права доступа по умолчанию. К таким записям не применены вообще никакие права приложения (по операциям на объект, по записям, по колонкам). Пользовательское соединение используется только для доступа к таблице базы данных.

На заметку. После выполнения запроса `InsertSelect` в базу данных будет добавлено столько записей, сколько вернется в его подзапросе `Select.`

На заметку. Полный перечень методов и свойств класса `InsertSelect`, его родительских классов, а также реализуемых им интерфейсов, можно найти в Библиотеке .NET классов.

Конструкторы

`InsertSelect(UserConnection userConnection)`

Создает экземпляр класса с указанным `UserConnection`.

`InsertSelect(InsertSelect source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`UserConnection` `Terrasoft.Core.UserConnection`

Пользовательское подключение, используемое при запросе.

`Source` `Terrasoft.Core.DB.ModifyQuerySource`

Источник данных

`Parameters` `Terrasoft.Core.DB.QueryParameterCollection`

Коллекция параметров запроса.

`HasParameters` `bool`

Определяет, имеет ли запрос параметры.

`BuildParametersAsValue` `bool`

Определяет, добавлять ли параметры запроса в текст запроса как значения.

`Columns` `Terrasoft.Core.DB.ModifyQueryColumnValueCollection`

Коллекция значений колонок запроса.

`Select` `Terrasoft.Core.DB.Select`

Используемый в запросе экземпляр `Terrasoft.Core.DB.Select`.

Методы

`void ResetCachedSqlText()`

Очищает кэшированный текст запроса.

`QueryParameterCollection GetUsingParameters()`

Возвращает коллекцию параметров, используемых запросом.

`void ResetParameters()`

Очищает коллекцию параметров запроса.

`void SetParameterValue(string name, object value)`

Устанавливает значение для параметра запроса.

Параметры

`name`	Название параметра.
`value`	Значение.

```
void InitializeParameters()
```

Инициализирует коллекцию параметров запроса.

```
int Execute()
```

Выполняет запрос. Возвращает количество задействованных запросом записей.

```
int Execute(DBExecutor dbExecutor)
```

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает количество задействованных запросом записей.

```
InsertSelect Into(string schemaName)
InsertSelect Into(ModifyQuerySource source)
```

Добавляет в текущий запрос источник данных.

Параметры

`schemaName`	Название схемы.
`source`	Источник данных.

```
InsertSelect Set(IEnumerable<string> sourceColumnAliases)
InsertSelect Set(params string[] sourceColumnAliases)
InsertSelect Set(IEnumerable<ModifyQueryColumn> columns)
InsertSelect Set(params ModifyQueryColumn[] columns)
```

Добавляет в текущий запрос набор колонок, в которые будут добавлены значения с помощью подзапроса. Возвращает текущий экземпляр `InsertSelect`.

Параметры

`sourceColumnAliases`	Коллекция или массив параметров метода, содержащие псевдонимы колонок.
`columns`	Коллекция или массив параметров метода, содержащие экземпляры колонок.

```
InsertSelect FromSelect(Select subSelect)
InsertSelect FromSelect(Query subSelectQuery)
```

Добавляет в текущий запрос предложение `SELECT`.

Параметры

`subSelect`	Подзапрос на выборку.
`subSelectQuery`	Подзапрос.

Класс Update C#

Сложный

Пространство имен `Terrasoft.Core.DB`.

Класс `Terrasoft.Core.DB.Update` предназначен для построения запросов на изменение записей в таблице базы данных Creatio. В результате создания и конфигурирования экземпляра этого класса будет построен запрос базу данных приложения в виде SQL-выражения `UPDATE`.

На заметку. Полный перечень методов и свойств класса `Update`, его родительских классов, а также реализуемых им интерфейсов, можно найти в Библиотеке .NET классов.

Конструкторы

```
Update(UserConnection userConnection)
```

Создает экземпляр класса, используя `UserConnection`.

```
Update(UserConnection userConnection, string schemaName)
```

Создает экземпляр класса для схемы с указанным названием, используя `UserConnection`.

```
Update(UserConnection userConnection, ModifyQuerySource source)
```

Создает экземпляр класса для указанного источника данных, используя `UserConnection`.

`Update(Insert source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`UserConnection` `Terrasoft.Core.UserConnection`

Пользовательское подключение, используемое при выполнении запроса.

`Condition` `Terrasoft.Core.DB.QueryCondition`

Условие выражения `Where` запроса.

`HasCondition` `bool`

Определяет наличие выражения `Where` в запросе.

`Source` `Terrasoft.Core.DB.ModifyQuerySource`

Источник данных запроса.

`ColumnValues` `Terrasoft.Core.DB.ModifyQueryColumnValueCollection`

Коллекция значений колонок запроса.

Методы

`void ResetCachedSqlText()`

Очищает закэшированный текст запроса.

`QueryParameterCollection GetUsingParameters()`

Возвращает коллекцию параметров, используемых запросом.

`int Execute()`

Выполняет запрос. Возвращает количество задействованных запросом записей.

```
int Execute(DBExecutor dbExecutor)
```

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает количество задействованных запросом записей.

```
QueryCondition Where()
QueryCondition Where(string sourceColumnAlias)
QueryCondition Where(string sourceAlias, string sourceColumnAlias)
QueryCondition Where(Select subSelect)
QueryCondition Where(Query subSelectQuery)
QueryCondition Where(QueryColumnExpression columnExpression)
Query Where(QueryCondition condition)
```

Добавляет к текущему запросу начальное условие.

Параметры

`sourceColumnAlias`	Псевдоним колонки, для которой добавляется условие.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос выборки данных, для результатов которого добавляется условие.
`subSelectQuery`	Подзапрос, для результатов которого добавляется условие.
`columnExpression`	Выражение, для результатов которого добавляется условие.
`condition`	Условие запроса.

```
QueryCondition And()
QueryCondition And(string sourceColumnAlias)
QueryCondition And(string sourceAlias, string sourceColumnAlias)
QueryCondition And(Select subSelect)
QueryCondition And(Query subSelectQuery)
QueryCondition And(QueryParameter parameter)
QueryCondition And(QueryColumnExpression columnExpression)
Query And(QueryCondition condition)
```

К текущему условию запроса добавляет условие (предикат), используя логическую операцию И.

Параметры

sourceColumnAlias	Псевдоним колонки, для которой добавляется предикат.
sourceAlias	Псевдоним источника.
subSelect	Подзапрос выборки данных, используемый в качестве предиката.
subSelectQuery	Подзапрос, используемый в качестве предиката.
parameter	Параметр, для которого добавляется предикат.
columnExpression	Выражение, используемое в качестве предиката.
condition	Условие запроса.

```
QueryCondition Or()
QueryCondition Or(string sourceColumnAlias)
QueryCondition Or(string sourceAlias, string sourceColumnAlias)
QueryCondition Or(Select subSelect)
QueryCondition Or(Query subSelectQuery)
QueryCondition Or(QueryParameter parameter)
QueryCondition Or(QueryColumnExpression columnExpression)
Query Or(QueryCondition condition)
```

К текущему условию запроса добавляет условие (предикат), используя логическую операцию ИЛИ.

Параметры

sourceColumnAlias	Псевдоним колонки, для которой добавляется предикат.
sourceAlias	Псевдоним источника.
subSelect	Подзапрос на выборку данных, используемый в качестве предиката.
subSelectQuery	Подзапрос, используемый в качестве предиката.
parameter	Параметр, для которого добавляется предикат.
columnExpression	Выражение, используемое в качестве предиката.
condition	Условие запроса.

```
Update Set(string sourceColumnAlias, Select subSelect)
```

```
Update Set(string sourceColumnAlias, Query subSelectQuery)
Update Set(string sourceColumnAlias, QueryColumnExpression columnExpression)
Update Set(string sourceColumnAlias, QueryParameter parameter)
```

Добавляет в текущий запрос предложение `SET` для присвоения колонке переданного выражения или параметра. Возвращает текущий экземпляр `Update`.

Параметры

`sourceColumnAlias`	Псевдоним колонки.
`subSelect`	Подзапрос на выборку.
`subSelectQuery`	Подзапрос.
`columnExpression`	Выражение колонки.
`parameter`	Параметр запроса.

Класс UpdateSelect C#

Сложный

Пространство имен `Terrasoft.Core.DB`.

Класс `Terrasoft.Core.DB.UpdateSelect` предназначен для построения запросов на изменение записей в таблице базы данных Creatio. При этом в качестве источника добавляемых данных используется экземпляр класса `Terrasoft.Core.DB.Select`. В результате создания и конфигурирования экземпляра этого класса будет построен запрос в базу данных приложения в виде SQL-выражения `UPDATE FROM`.

На заметку. Полный перечень методов и свойств класса `UpdateSelect`, его родительских классов, а также реализуемых им интерфейсов, можно найти в Библиотеке .NET классов.

Конструкторы

```
public UpdateSelect(UserConnection userConnection, string schemaName, string alias)
```

Создает экземпляр класса для схемы с указанным названием, используя `UserConnection`.

Параметры

`userConnection`	Пользовательское подключение, используемое при запросе.
`schemaName`	Название схемы.
`alias`	Псевдоним таблицы.

Свойства

`SourceAlias` `string`

Псевдоним таблицы данных, в которую вносятся изменения.

`SourceExpression` `Terrasoft.Core.DB.QuerySourceExpression`

Выражение для `SELECT`-запроса.

Методы

`public UpdateSelect From(string schemaName, string alias)`

Добавляет к запросу `FROM`-выражение.

Параметры

`schemaName`	Название таблицы, в которую вносятся изменения.
`alias`	Псевдоним таблицы, в которую вносятся изменения.

`public UpdateSelect Set(string sourceColumnAlias, QueryColumnExpression columnExpression)`

Добавляет к запросу `SET`-выражение для колонки.

Параметры

`sourceColumnAlias`	Псевдоним колонки.
`columnExpression`	Выражение, содержащее значение колонки.

Класс Delete C#

Сложный

Пространство имен `Terrasoft.Core.DB`.

Класс `Terrasoft.Core.DB.Delete` предназначен для построения запросов на удаление записей в таблице базы данных Creatio. В результате создания и конфигурирования экземпляра этого класса будет построен запрос базу данных приложения в виде SQL-выражения `DELETE`.

На заметку. Полный перечень методов и свойств класса `Delete`, его родительских классов, а также реализуемых им интерфейсов, можно найти в Библиотеке .NET классов.

Конструкторы

`Delete(UserConnection userConnection)`

Создает экземпляр класса, используя `UserConnection`.

`Delete(Delete source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`UserConnection` `Terrasoft.Core.UserConnection`

Пользовательское подключение, используемое при выполнении запроса.

`Condition` `Terrasoft.Core.DB.QueryCondition`

Условие выражения `Where` запроса.

`HasCondition` `bool`

Определяет наличие выражения `Where` в запросе.

`Source` `Terrasoft.Core.DB.ModifyQuerySource`

Источник данных запроса.

Методы

`void ResetCachedSqlText()`

Очищает закэшированный текст запроса.

```
QueryParameterCollection GetUsingParameters()
```

Возвращает коллекцию параметров, используемых запросом.

```
int Execute()
```

Выполняет запрос. Возвращает количество задействованных запросом записей.

```
int Execute(DBExecutor dbExecutor)
```

Выполняет запрос, используя экземпляр `DBExecutor`. Возвращает количество задействованных запросом записей.

```
QueryCondition Where()
QueryCondition Where(string sourceColumnAlias)
QueryCondition Where(string sourceAlias, string sourceColumnAlias)
QueryCondition Where(Select subSelect)
QueryCondition Where(Query subSelectQuery)
QueryCondition Where(QueryColumnExpression columnExpression)
Query Where(QueryCondition condition)
```

Добавляет к текущему запросу начальное условие.

Параметры

`sourceColumnAlias`	Псевдоним колонки, для которой добавляется условие.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос выборки данных, для результатов которого добавляется условие.
`subSelectQuery`	Подзапрос, для результатов которого добавляется условие.
`columnExpression`	Выражение, для результатов которого добавляется условие.
`condition`	Условие запроса.

```
QueryCondition And()
QueryCondition And(string sourceColumnAlias)
QueryCondition And(string sourceAlias, string sourceColumnAlias)
QueryCondition And(Select subSelect)
QueryCondition And(Query subSelectQuery)
QueryCondition And(QueryParameter parameter)
```

```
QueryCondition And(QueryColumnExpression columnExpression)
Query And(QueryCondition condition)
```

К текущему условию запроса добавляет условие (предикат), используя логическую операцию И.

Параметры

`sourceColumnAlias`	Псевдоним колонки, для которой добавляется предикат.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос выборки данных, используемый в качестве предиката.
`subSelectQuery`	Подзапрос, используемый в качестве предиката.
`parameter`	Параметр, для которого добавляется предикат.
`columnExpression`	Выражение, используемое в качестве предиката.
`condition`	Условие запроса.

```
QueryCondition Or()
QueryCondition Or(string sourceColumnAlias)
QueryCondition Or(string sourceAlias, string sourceColumnAlias)
QueryCondition Or(Select subSelect)
QueryCondition Or(Query subSelectQuery)
QueryCondition Or(QueryParameter parameter)
QueryCondition Or(QueryColumnExpression columnExpression)
Query Or(QueryCondition condition)
```

К текущему условию запроса добавляет условие (предикат), используя логическую операцию ИЛИ.

Параметры

`sourceColumnAlias`	Псевдоним колонки, для которой добавляется предикат.
`sourceAlias`	Псевдоним источника.
`subSelect`	Подзапрос на выборку данных, используемый в качестве предиката.
`subSelectQuery`	Подзапрос, используемый в качестве предиката.
`parameter`	Параметр, для которого добавляется предикат.
`columnExpression`	Выражение, используемое в качестве предиката.
`condition`	Условие запроса.

```
Delete From(string schemaName)
Delete From((ModifyQuerySource source)
```

Добавляет в текущий запрос источник данных. Возвращает текущий экземпляр `Delete`.

Параметры

`schemaName`	Название схемы (таблицы, представления).
`source`	Источник данных.

Класс QueryFunction C#

Сложный

Класс `Terrasoft.Core.DB.QueryFunction` реализует функцию выражения.

Функция выражения реализована в следующих классах:

- `QueryFunction` — базовый класс функции выражения.
- `AggregationQueryFunction` — реализует агрегирующую функцию выражения.
- `IsNullQueryFunction` — заменяет значения `null` замещающим выражением.
- `CreateGuidQueryFunction` — реализует функцию выражения нового идентификатора.
- `CurrentDateTimeQueryFunction` — реализует функцию выражения текущей даты и времени.
- `CoalesceQueryFunction` — возвращает первое выражение из списка аргументов, не равное `null`.
- `DatePartQueryFunction` — реализует функцию выражения части значения типа `Дата/Время`.
- `DateAddQueryFunction` — реализует функцию выражения даты, полученной путем добавления указанного промежутка времени к заданной дате.

- `DateDiffQueryFunction` — реализует функцию выражения разницы дат, полученного путем вычитания заданных дат.
- `CastQueryFunction` — приводит выражение аргумента к заданному типу данных.
- `UpperQueryFunction` — преобразовывает символы выражения аргумента в верхний регистр.
- `CustomQueryFunction` — реализует пользовательскую функцию.
- `DataLengthQueryFunction` — определяет число байтов, использованных для представления выражения.
- `TrimQueryFunction` — удаляет начальные и конечные пробелы из выражения.
- `LengthQueryFunction` — возвращает длину выражения.
- `SubstringQueryFunction` — получает часть строки.
- `ConcatQueryFunction` — формирует строку, которая является результатом объединения строковых значений аргументов функции.
- `WindowQueryFunction` — реализует функцию SQL окна.

Класс QueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Базовый класс функции выражения.

На заметку. Полный перечень методов класса `QueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Методы

`static QueryColumnExpression Negate(QueryFunction operand)`

Возвращает выражение отрицания значения переданной функции.

Параметры

`operand`	Функция выражения.

`static QueryColumnExpression operator -(QueryFunction operand)`

Перегрузка оператора отрицания переданной функции выражения.

Параметры

`operand`	Функция выражения.

`static QueryColumnExpression Add(QueryFunction leftOperand, QueryFunction rightOperand)`

Возвращает выражение арифметического сложения переданных функций выражения.

Параметры

`leftOperand`	Левый операнд в операции сложения.
`rightOperand`	Правый операнд в операции сложения.

`static QueryColumnExpression operator +(QueryFunction leftOperand, QueryFunction rightOperand)`

Перегрузка оператора сложения двух функций выражений.

Параметры

`leftOperand`	Левый операнд в операции сложения.
`rightOperand`	Правый операнд в операции сложения.

`static QueryColumnExpression Subtract(QueryFunction leftOperand, QueryFunction rightOperand)`

Возвращает выражение вычитания правой функции выражения из левой.

Параметры

`leftOperand`	Левый операнд в операции вычитания.
`rightOperand`	Правый операнд в операции вычитания.

`static QueryColumnExpression operator -(QueryFunction leftOperand, QueryFunction rightOperand)`

Перегрузка оператора вычитания правой функции выражения из левой.

Параметры

`leftOperand`	Левый операнд в операции вычитания.
`rightOperand`	Правый операнд в операции вычитания.

`static QueryColumnExpression Multiply(QueryFunction leftOperand, QueryFunction rightOperand)`

Возвращает выражение умножения переданных функций выражений.

Параметры

leftOperand	Левый операнд в операции умножения.
rightOperand	Правый операнд в операции умножения.

```
static QueryColumnExpression operator *(QueryFunction leftOperand, QueryFunction rightOperand)
```

Перегрузка оператора умножения двух функций выражений.

Параметры

leftOperand	Левый операнд в операции умножения.
rightOperand	Правый операнд в операции умножения.

```
static QueryColumnExpression Divide(QueryFunction leftOperand, QueryFunction rightOperand)
```

Возвращает выражение деления левой функции выражения на правую.

Параметры

leftOperand	Левый операнд в операции деления.
rightOperand	Правый операнд в операции деления.

```
static QueryColumnExpression operator /(QueryFunction leftOperand, QueryFunction rightOperand)
```

Перегрузка оператора деления функций выражений.

Параметры

leftOperand	Левый операнд в операции деления.
rightOperand	Правый операнд в операции деления.

```
abstract object Clone()
```

Создает копию текущего экземпляра `QueryFunction`.

```
abstract void BuildSqlText(StringBuilder sb, DBEngine dbEngine)
```

Формирует текст запроса с использованием переданных экземпляра `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запросов к базе данных.

```
virtual void AddUsingParameters(QueryParameterCollection resultParameters)
```

Добавляет переданную коллекцию параметров в аргументы функции.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
QueryColumnExpressionCollection GetQueryColumnExpressions()
```

Возвращает коллекцию выражений колонки запроса для текущей функции запроса.

```
QueryColumnExpression GetQueryColumnExpression()
```

Возвращает выражение колонки запроса для текущей функции запроса.

Класс AggregationQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует агрегирующую функцию выражения.

На заметку. Полный перечень методов и свойств класса `AggregationQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
AggregationQueryFunction()
```

Инициализирует новый экземпляр `AggregationQueryFunction`.

`AggregationQueryFunction(AggregationTypeStrict aggregationType, QueryColumnExpression expressior`

Инициализирует новый экземпляр `AggregationQueryFunction` с заданным типом агрегирующей функции для указанного выражения колонки.

Параметры

`aggregationType`	Тип агрегирующей функции.
`expression`	Выражение колонки, к которому применяется агрегирующая функция.

`AggregationQueryFunction(AggregationTypeStrict aggregationType, IQueryColumnExpressionConvertibl`

Инициализирует новый экземпляр `AggregationQueryFunction` с заданным типом агрегирующей функции для указанного выражения колонки.

Параметры

`aggregationType`	Тип агрегирующей функции.
`expression`	Выражение колонки, к которому применяется агрегирующая функция.

`AggregationQueryFunction(AggregationQueryFunction source)`

Инициализирует новый экземпляр `AggregationQueryFunction`, являющийся клоном переданной агрегирующей функции выражения.

Параметры

`source`	Агрегирующая функция выражения `AggregationQueryFunction`, клон которой создается.

Свойства

`AggregationType` `AggregationTypeStrict`

Тип агрегирующей функции.

`AggregationEvalType` `AggregationEvalType`

Область применения агрегирующей функции.

`Expression QueryColumnExpression`

Выражение аргумента функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запроса `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запросов к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет переданную коллекцию параметров в аргументы функции.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `AggregationQueryFunction`.

`AggregationQueryFunction All()`

Устанавливает для текущей агрегирующей функции область применения [*Ко всем значениям*].

`AggregationQueryFunction Distinct()`

Устанавливает для текущей агрегирующей функции область применения [*К уникальным значениям*].

Класс IsNullQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс заменяет значения `null` замещающим выражением.

На заметку. Полный перечень методов и свойств класса `IsNullQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`IsNullQueryFunction()`

Инициализирует новый экземпляр `IsNullQueryFunction`.

`IsNullQueryFunction(QueryColumnExpression checkExpression, QueryColumnExpression replacementExpr`
`IsNullQueryFunction(IQueryColumnExpressionConvertible checkExpression, IQueryColumnExpressionCor`

Инициализирует новый экземпляр `IsNullQueryFunction` для заданных проверяемого выражения и замещающего выражения.

Параметры

`checkExpression`	Выражение, которое проверяется на равенство `null`.
`replacementExpression`	Выражение, которое возвращается функцией, если `checkExpression` равно `null`.

`IsNullQueryFunction(IsNullQueryFunction source)`

Инициализирует новый экземпляр `IsNullQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Агрегирующая функция выражения `IsNullQueryFunction`, клон которой создается.

Свойства

`CheckExpression` `QueryColumnExpression`

Выражение аргумента функции, которое проверяется на равенство значению `null`.

`ReplacementExpression` `QueryColumnExpression`

Выражение аргумента функции, которое возвращается, если проверяемое выражение равно `null`.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запросов к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет переданную коллекцию параметров в аргументы функции.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `IsNullQueryFunction`.

Класс CreateGuidQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует функцию выражения нового идентификатора.

На заметку. Полный перечень методов класса `CreateGuidQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`CreateGuidQueryFunction()`

Инициализирует новый экземпляр `CreateGuidQueryFunction`.

`CreateGuidQueryFunction(CreateGuidQueryFunction source)`

Инициализирует новый экземпляр `CreateGuidQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `CreateGuidQueryFunction`, клон которой создается.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override object Clone()`

Создает клон текущего экземпляра `CreateGuidQueryFunction`.

Класс CurrentDateTimeQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует функцию выражения текущей даты и времени.

На заметку. Полный перечень методов класса `CurrentDateTimeQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`CurrentDateTimeQueryFunction()`

Инициализирует новый экземпляр `CurrentDateTimeQueryFunction`.

`CurrentDateTimeQueryFunction(CurrentDateTimeQueryFunction source)`

Инициализирует новый экземпляр `CurrentDateTimeQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `CurrentDateTimeQueryFunction`, клон которой создается.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override object Clone()`

Создает клон текущего экземпляра `CurrentDateTimeQueryFunction`.

Класс CoalesceQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс возвращает первое выражение из списка аргументов, не равное `null`.

На заметку. Полный перечень методов и свойств класса `CoalesceQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`CoalesceQueryFunction()`

Инициализирует новый экземпляр `CoalesceQueryFunction`.

`CoalesceQueryFunction(CoalesceQueryFunction source)`

Инициализирует новый экземпляр `CoalesceQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `CoalesceQueryFunction`, клон которой создается.

`CoalesceQueryFunction(QueryColumnExpressionCollection expressions)`

Инициализирует новый экземпляр `CoalesceQueryFunction` для переданной коллекции выражений колонок.

Параметры

`expressions`	Коллекция выражений колонок запроса.

`CoalesceQueryFunction(QueryColumnExpression[] expressions)`
`CoalesceQueryFunction(IQueryColumnExpressionConvertible[] expressions)`

Инициализирует новый экземпляр `CoalesceQueryFunction` для переданного массива выражений колонок.

Параметры

`expressions`	Массив выражений колонок запроса.

Свойства

`Expressions` `QueryColumnExpressionCollection`

Коллекция выражений аргументов функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

```
override object Clone()
```

Создает клон текущего экземпляра `CoalesceQueryFunction`.

```
override void AddUsingParameters(QueryParameterCollection resultParameters)
```

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

Класс DatePartQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует функцию выражения части значения типа `Дата/Время`.

На заметку. Полный перечень методов и свойств класса `DatePartQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
DatePartQueryFunction()
```

Инициализирует новый экземпляр `DatePartQueryFunction`.

```
DatePartQueryFunction(DatePartQueryFunctionInterval interval, QueryColumnExpression expression)
DatePartQueryFunction(DatePartQueryFunctionInterval interval, IQueryColumnExpressionConvertible
```

Инициализирует новый экземпляр `DatePartQueryFunction` с заданным выражением колонки типа `Дата/Время` и указанной частью даты.

Параметры

`interval`	Часть даты.
`expression`	Выражение колонки типа `Дата/Время`.

`DatePartQueryFunction(DatePartQueryFunction source)`

Инициализирует новый экземпляр `DatePartQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `DatePartQueryFunction`, клон которой создается.

Свойства

`Expression` `QueryColumnExpression`

Выражение аргумента функции.

`Interval` `DatePartQueryFunctionInterval`

Часть даты, возвращаемая функцией.

`UseUtcOffset` `bool`

Использование смещения всеобщего скоординированного времени (UTC) относительно заданного местного времени.

`UtcOffset` `int?`

Смещение всеобщего скоординированного времени (UTC).

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

```
override void AddUsingParameters(QueryParameterCollection resultParameters)
```

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
override object Clone()
```

Создает клон текущего экземпляра `DatePartQueryFunction`.

Класс DateAddQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует функцию выражения даты, полученной путем добавления указанного промежутка времени к заданной дате.

На заметку. Полный перечень методов и свойств класса `DateAddQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
DateAddQueryFunction()
```

Инициализирует новый экземпляр `DateAddQueryFunction`.

```
DateAddQueryFunction(DatePartQueryFunctionInterval interval, int number, QueryColumnExpression e
DateAddQueryFunction(DatePartQueryFunctionInterval interval, IQueryColumnExpressionConvertible r
DateAddQueryFunction(DatePartQueryFunctionInterval interval, int number, IQueryColumnExpressionC
```

Инициализирует экземпляр `DateAddQueryFunction` с заданными параметрами.

Параметры

`interval`	Часть даты, к которой добавляется временной промежуток.
`number`	Значение, которое добавляется к `interval`.
`expression`	Выражение колонки, содержащей исходную дату.

`DateAddQueryFunction(DateAddQueryFunction source)`

Инициализирует экземпляр `DateAddQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Экземпляр функции `DateAddQueryFunction`, клон которой создается.

Свойства

`Expression` `QueryColumnExpression`

Выражение колонки, содержащей исходную дату.

`Interval` `DatePartQueryFunctionInterval`

Часть даты, к которой добавляется временной промежуток.

`Number` `int`

Добавляемый временной промежуток.

`NumberExpression` `QueryColumnExpression`

Выражение, содержащие добавляемый временной промежуток.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

```
override void AddUsingParameters(QueryParameterCollection resultParameters)
```

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
override object Clone()
```

Создает клон текущего экземпляра `DateAddQueryFunction`.

Класс DateDiffQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует функцию выражения разницы дат, полученного путем вычитания заданных дат.

На заметку. Полный перечень методов и свойств класса `DateDiffQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
DateDiffQueryFunction(DateDiffQueryFunctionInterval interval, QueryColumnExpression startDateExp
DateDiffQueryFunction(DateDiffQueryFunctionInterval interval, IQueryColumnExpressionConvertible
```

Инициализирует экземпляр `DateDiffQueryFunction` с заданными параметрами.

Параметры

`interval`	Единица измерения разницы дат.
`startDateExpression`	Выражение колонки, содержащей начальную дату.
`endDateExpression`	Выражение колонки, содержащей конечную дату.

`DateDiffQueryFunction(DateDiffQueryFunction source)`

Инициализирует экземпляр `DateDiffQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Экземпляр функции `DateDiffQueryFunction`, клон которой создается.

Свойства

`StartDateExpression` `QueryColumnExpression`

Выражение колонки, содержащей начальную дату.

`EndDateExpression` `QueryColumnExpression`

Выражение колонки, содержащей конечную дату.

`Interval` `DateDiffQueryFunctionInterval`

Единица измерения разницы дат, возвращаемая функцией.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
override object Clone()
```

Создает клон текущего экземпляра `DateDiffQueryFunction`.

Класс CastQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс приводит выражение аргумента к заданному типу данных.

На заметку. Полный перечень методов и свойств класса `CastQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
CastQueryFunction(QueryColumnExpression expression, DBDataValueType castType)
CastQueryFunction(IQueryColumnExpressionConvertible expression, DBDataValueType castType)
```

Инициализирует новый экземпляр `CastQueryFunction` с заданными выражением колонки и целевым типом данных.

Параметры

`expression`	Выражение колонки запроса.
`castType`	Целевой тип данных.

```
CastQueryFunction(CastQueryFunction source)
```

Инициализирует новый экземпляр `CastQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `CastQueryFunction`, клон которой создается.

Свойства

```
Expression QueryColumnExpression
```

Выражение аргумента функции.

`CastType DBDataValueType`

Целевой тип данных.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `CastQueryFunction`.

Класс UpperQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс преобразовывает символы выражения аргумента в верхний регистр.

На заметку. Полный перечень методов и свойств класса `UpperQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`UpperQueryFunction()`

Инициализирует новый экземпляр `UpperQueryFunction`.

```
UpperQueryFunction(QueryColumnExpression expression)
UpperQueryFunction(IQueryColumnExpressionConvertible expression)
```

Инициализирует новый экземпляр `UpperQueryFunction` для заданного выражения колонки.

Параметры

`expression`	Выражение колонки запроса.

`UpperQueryFunction(UpperQueryFunction source)`

Инициализирует новый экземпляр `UpperQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `UpperQueryFunction`, клон которой создается.

Свойства

`Expression`

Выражение аргумента функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

```
override void AddUsingParameters(QueryParameterCollection resultParameters)
```

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
override object Clone()
```

Создает клон текущего экземпляра `UpperQueryFunction`.

Класс CustomQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует пользовательскую функцию.

На заметку. Полный перечень методов и свойств класса `CustomQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
CustomQueryFunction()
```

Инициализирует новый экземпляр `CustomQueryFunction`.

```
CustomQueryFunction(string functionName, QueryColumnExpressionCollection expressions)
```

Инициализирует новый экземпляр `CustomQueryFunction` для заданной функции и переданной коллекции выражений колонок.

Параметры

`functionName`	Имя функции.
`expressions`	Коллекция выражений колонок запроса.

```
CustomQueryFunction(string functionName, QueryColumnExpression[] expressions)
CustomQueryFunction(string functionName, IQueryColumnExpressionConvertible[] expressions)
```

Инициализирует новый экземпляр `CustomQueryFunction` для заданной функции и переданного массива выражений колонок.

Параметры

`functionName`	Имя функции.
`expressions`	Массив выражений колонок запроса.

`CustomQueryFunction(CustomQueryFunction source)`

Инициализирует новый экземпляр `CustomQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `CustomQueryFunction`, клон которой создается.

Свойства

`Expressions` `QueryColumnExpressionCollection`

Коллекция выражений аргументов функции.

`FunctionName` `string`

Имя функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет заданные параметры в коллекцию.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `CustomQueryFunction`.

Класс DataLengthQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс определяет число байтов, использованных для представления выражения.

На заметку. Полный перечень методов и свойств класса `DataLengthQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`DataLengthQueryFunction()`

Инициализирует новый экземпляр `DataLengthQueryFunction`.

`DataLengthQueryFunction(QueryColumnExpression expression)`

Инициализирует новый экземпляр `DataLengthQueryFunction` для заданного выражения колонки.

Параметры

`expression`	Выражение колонки запроса.

`DataLengthQueryFunction(IQueryColumnExpressionConvertible columnNameExpression)`

Инициализирует новый экземпляр `DataLengthQueryFunction` для заданного выражения колонки.

Параметры

`columnNameExpression`	Выражение колонки запроса.

`DataLengthQueryFunction(DataLengthQueryFunction source)`

Инициализирует новый экземпляр `DataLengthQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `DataLengthQueryFunction`, клон которой создается.

Свойства

`Expression QueryColumnExpression`

Выражение аргумента функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет в аргументы функции переданную коллекцию параметров.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `DataLengthQueryFunction`.

Класс TrimQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс удаляет начальные и конечные пробелы из выражения.

На заметку. Полный перечень методов и свойств класса `TrimQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
TrimQueryFunction(QueryColumnExpression expression)
TrimQueryFunction(IQueryColumnExpressionConvertible expression)
```

Инициализирует новый экземпляр `TrimQueryFunction` для заданного выражения колонки.

Параметры

`expression`	Выражение колонки запроса.

```
TrimQueryFunction(TrimQueryFunction source)
```

Инициализирует новый экземпляр `TrimQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `TrimQueryFunction`, клон которой создается.

Свойства

```
Expression QueryColumnExpression
```

Выражение аргумента функции.

Методы

```
override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)
```

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

```
override void AddUsingParameters(QueryParameterCollection resultParameters)
```

Добавляет в аргументы функции переданную коллекцию параметров.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
override object Clone()
```

Создает клон текущего экземпляра `TrimQueryFunction`.

Класс LengthQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс возвращает длину выражения.

На заметку. Полный перечень методов и свойств класса `LengthQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
LengthQueryFunction()
```

Инициализирует новый экземпляр `LengthQueryFunction`.

```
LengthQueryFunction(QueryColumnExpression expression)
LengthQueryFunction(IQueryColumnExpressionConvertible expression)
```

Инициализирует новый экземпляр `LengthQueryFunction` для заданного выражения колонки.

Параметры

`expression`	Выражение колонки запроса.

`LengthQueryFunction(LengthQueryFunction source)`

Инициализирует новый экземпляр `LengthQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `LengthQueryFunction`, клон которой создается.

Свойства

`Expression QueryColumnExpression`

Выражение аргумента функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет в аргументы функции переданную коллекцию параметров.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `LengthQueryFunction`.

Класс SubstringQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс получает часть строки.

На заметку. Полный перечень методов и свойств класса `SubstringQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
SubstringQueryFunction(QueryColumnExpression expression, int start, int length)
SubstringQueryFunction(IQueryColumnExpressionConvertible expression, int start, int length)
```

Инициализирует новый экземпляр `SubstringQueryFunction` для заданного выражения колонки, начальной позиции и длины подстроки.

Параметры

`expression`	Выражение колонки запроса.
`start`	Начальная позиция подстроки.
`length`	Длина подстроки.

```
SubstringQueryFunction(SubstringQueryFunction source)
```

Инициализирует новый экземпляр `SubstringQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `SubstringQueryFunction`, клон которой создается.

Свойства

`Expression` `QueryColumnExpression`

Выражение аргумента функции.

`StartExpression` `QueryColumnExpression`

Начальная позиция подстроки.

`LengthExpression` `QueryColumnExpression`

Длина подстроки.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет в аргументы функции переданную коллекцию параметров.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `SubstringQueryFunction`.

Класс ConcatQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс формирует строку, которая является результатом объединения строковых значений аргументов функции.

На заметку. Полный перечень методов и свойств класса `ConcatQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`ConcatQueryFunction(QueryColumnExpressionCollection expressions)`

Инициализирует новый экземпляр `ConcatQueryFunction` для переданной коллекции выражений.

Параметры

`expressions`	Коллекция выражений колонок запроса.

`ConcatQueryFunction(ConcatQueryFunction source)`

Инициализирует новый экземпляр `ConcatQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `ConcatQueryFunction`, клон которой создается.

Свойства

`Expressions QueryColumnExpressionCollection`

Коллекция выражений аргументов функции.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет в аргументы функции переданную коллекцию параметров.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

```
override object Clone()
```

Создает клон текущего экземпляра `ConcatQueryFunction`.

Класс WindowQueryFunction C#

Пространство имен `Terrasoft.Core.DB`.

Класс реализует функцию SQL окна.

> **На заметку.** Полный перечень методов и свойств класса `WindowQueryFunction`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

```
WindowQueryFunction(QueryFunction innerFunction)
```

Реализует функцию SQL окна.

Параметры

`innerFunction`	Вложенная функция.

```
WindowQueryFunction(QueryFunction innerFunction, QueryColumnExpression partitionByExpression = r
```

Реализует функцию SQL окна.

Параметры

`innerFunction`	Вложенная функция.
`partitionByExpression`	Выражение для разделения запроса.
`orderByExpression`	Выражение для сортировки запроса.

```
WindowQueryFunction(WindowQueryFunction source) : this( source.InnerFunction, source.PartitionBy
```

Инициализирует новый экземпляр `WindowQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `WindowQueryFunction`, клон которой создается.

Свойства

`InnerFunction` `QueryFunction`

Функция для применения.

`PartitionByExpression` `QueryColumnExpression`

Разделение по пунктам.

`OrderByExpression` `QueryColumnExpression`

Сортировать по пункту.

Методы

`override void BuildSqlText(StringBuilder sb, DBEngine dbEngine)`

Формирует текст запроса с использованием заданных экземпляров `StringBuilder` и построителя запросов `DBEngine`.

Параметры

`sb`	Экземпляр `StringBuilder`, с помощью которого формируется текст запроса.
`dbEngine`	Экземпляр построителя запроса к базе данных.

`override void AddUsingParameters(QueryParameterCollection resultParameters)`

Добавляет в аргументы функции переданную коллекцию параметров.

Параметры

`resultParameters`	Коллекция параметров запроса, которые добавляются в аргументы функции.

`override object Clone()`

Создает клон текущего экземпляра `WindowQueryFunction`.

Доступ к данным через ORM

Средний

Способы доступа к базе данных, которые предоставляют back-end компоненты ядра:

- Доступ через ORM-модель.
- Прямой доступ.

В этой статье будет рассмотрено выполнение запросов к базе данных через ORM-модель.

ORM (Object-relational mapping) — технология, которая позволяет работать с данными, полученными из базы данных, путем использования объектно-ориентированных языков программирования. **Назначение** ORM — связывание объектов, реализованных в коде, с записями в таблицах базы данных.

Классы, которые реализуют работу с данными через ORM-модель данных:

- `Terrasoft.Core.Entities.EntitySchemaQuery` — построение запросов на получение записей из таблиц базы данных с учетом прав доступа текущего пользователя.
- `Terrasoft.Core.Entities.Entity` — доступ к сущности, которая представляет собой запись в таблице базы данных.

Для доступа к данным рекомендуется использовать ORM-модель, хотя прямой доступ к базе данных также реализован в back-end компонентах ядра. Выполнение запросов к базе данных через прямой доступ подробно описано в статье Прямой доступ к данным.

Сформировать путь к колонке относительно корневой схемы

Основой механизма построения запроса на выборку с применением `EntitySchemaQuery` является корневая схема. **Корневая схема** — это таблица в базе данных, относительно которой строятся пути к колонкам в запросе, в том числе к колонкам присоединяемых таблиц. Для использования в запросе колонки из таблицы базы данных необходимо корректно задать путь к этой колонке.

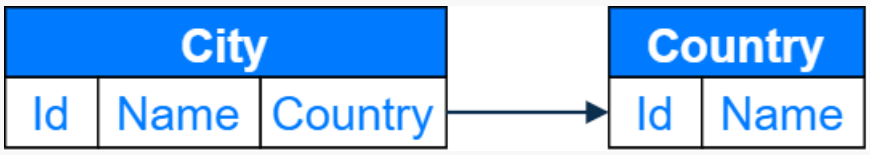
При построении путей к колонкам применяется **принцип связей через справочные поля**. Имя колонки, добавляемой в запрос, строится в виде цепочки взаимосвязанных звеньев. Каждое звено представляет собой "контекст" конкретной схемы, которая связывается с предыдущей по справочной колонке.

Шаблон формирования пути к колонке из схемы N:
`КонтекстСхемы1.[...].КонтекстСхемыN.ИмяКолонкиИзСправочнойСхемы`.

Сформировать путь к колонке по прямым связям

Шаблон формирования пути к колонке по **прямым связям**:
`ИмяСправочнойКолонки.ИмяКолонкиСхемыИзСправочнойСхемы`.

Прямые связи используются, когда справочная колонка для связи присутствует в основной схеме и ссылается на присоединяемую схему. Например, есть корневая схема `[City]` со справочной колонкой `[Country]`, которая через колонку `[Id]` связана со справочной схемой `[Country]`.



Путь к колонке с наименованием страны, которой принадлежит город по прямым связям: `Country.Name`. Здесь:

- `Country` — имя справочной колонки корневой схемы `[City]` (ссылается на схему `[Country]`).
- `Name` — имя колонки из справочной схемы `[Country]`.

Сформировать путь к колонке по обратным связям

Отличие присоединения по обратным связям от присоединения по прямым связям — справочное поле для присоединения должно быть у присоединяемой сущности, а не у основной.

Шаблон формирования пути к колонке по **обратным связям**:
`[ИмяПрисоединяемойСхемы:ИмяКолонкиДляСвязиПрисоединяемойСхемы:ИмяКолонкиДляСвязиТекущейСхемы].ИмяКолонкиПрисоединяемойСхемы`
.

Путь к колонке с названием контрагента, у которого в поле `[Город]` указана выбираемая в запросе запись `[City]` по обратным связям: `[Account:City:Id].Name`. Здесь:

- `Account` — имя присоединяемой схемы.
- `City` — имя колонки схемы `[Account]` для установки связи присоединяемой схемы.
- `Id` — имя справочной колонки схемы `[City]` для установки связи текущей схемы.
- `Name` — значение справочной колонки схемы `[Account]`.

Если в качестве колонки для связи у текущей схемы выступает колонка `[Id]`, то ее можно не указывать: `[ИмяПрисоединяемойСхемы:ИмяКолонкиДляСвязиПрисоединяемойСхемы].ИмяКолонкиПрисоединяемойСхемы`. Например, `[Account:City].Name`.

Получить данные из базы данных

Классы, которые реализуют получение данных из базы данных:

- `Terrasoft.Core.DB.Select` — получение данных из базы данных через прямой доступ. Подробнее читайте в статье Прямой доступ к данным.
- `Terrasoft.Core.Entities.EntitySchemaQuery` — получение данных из базы данных через ORM-модель.

Назначение класса `Terrasoft.Core.Entities.EntitySchemaQuery` — построение запросов на выборку записей из таблиц базы данных. Максимальное количество записей, которые можно получить по запросу, задается настройкой `MaxEntityRowCount` (по умолчанию — 20 000). Изменить значение настройки можно в файле `...\Terrasoft.WebApp\Web.config`.

> **Важно.** Не рекомендуется изменять настройку `MaxEntityRowCount`. Изменение настройки может привести к проблемам производительности.

После создания и конфигурирования экземпляра класса будет построен `SELECT`-запрос к базе данных приложения. В запрос можно добавить колонки, фильтры и условия ограничений. Также можно задать параметры для постраничного вывода результатов выполнения запроса. Используя класс `Terrasoft.Core.Entities.EntitySchemaQueryOptions`, можно задать параметры построения иерархического запроса. Передача одного и того же экземпляра `EntitySchemaQueryOptions` в качестве параметра метода `GetEntityCollection()` соответствующего запроса позволяет получить результат выполнения различных запросов.

Особенности класса `Terrasoft.Core.Entities.EntitySchemaQuery`:

- В результирующем запросе учитываются права доступа текущего пользователя.
- Класс позволяет управлять правами доступа текущего пользователя на таблицы, присоединенные в запрос с помощью SQL-оператора `JOIN`.
- Класс позволяет работать с данными хранилища кэша или произвольного хранилища, которое определено пользователем.

При выполнении запроса данные, полученные из базы данных, помещаются в кэш. В качестве кэша запроса может выступать произвольное хранилище, которое реализует интерфейс `Terrasoft.Core.Store.ICacheStore`. По умолчанию используется кэш Creatio уровня сессии с локальным хранением данных. Кэш запроса определяется свойством `Cache` экземпляра класса `EntitySchemaQuery`. С помощью свойства `CacheItemName` задается ключ доступа к кэшу запроса. Уровни хранилищ Creatio подробно описаны в статье Хранилища данных и кэш.

Результат выполнения запроса — экземпляр `Terrasoft.Nui.ServiceModel.DataContract.EntityCollection` или коллекция экземпляров класса `Terrasoft.Core.Entities.Entity`. Каждый экземпляр `Entity` в коллекции представляет собой строку набора данных, возвращаемого запросом.

Управлять присоединенными таблицами

Класс `EntitySchemaQuery` позволяет указать тип присоединения схемы к запросу. Для добавления в запрос колонки из присоединяемой схемы используется оператор `JOIN`.

Шаблон формирования типа соединения к колонке присоединяемой схемы:
`СпецсимволТипаСоединенияИмяКолонкиДляСвязиПрисоединяемойСхемы`.

Описание типов соединения

Тип соединения	Спецсимвол типа соединения	Пример использования
`INNER JOIN`	`=`	`=Country.Name`
`LEFT OUTER JOIN`	`>`	`>Country.Name`
`RIGHT OUTER JOIN`	`<`	`<Country.Name`
`FULL OUTER JOIN`	`<>`	`<>Country.Name`
`CROSS JOIN`	`*`	`*Country.Name`

По умолчанию используется тип присоединения `LEFT OUTER JOIN`.

Ниже представлен пример добавления колонок в запрос с использованием разных типов соединения.

Пример добавления колонок в запрос

```
/* Создание экземпляра запроса с корневой схемой City. */
var esqResult = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "City");

/* К запросу будет добавлена схема Country с типом присоединения LEFT OUTER JOIN. */
esqResult.AddColumn("Country.Name");

/* К запросу будет добавлена схема Country с типом присоединения INNER JOIN. */
esqResult.AddColumn("=Country.Name");

/* К запросу будут присоединены схема Country с типом присоединения LEFT OUTER JOIN и схема Cont
esqResult.AddColumn(">Country.<CreatedBy.Name");
```

В результате будет сформирован SQL-запрос.

SQL-запрос

```
SELECT
    [Country].[Name] [Country.Name],
    [Country1].[Name] [Country1.Name],
    [CreatedBy].[Name] [CreatedBy.Name]
FROM
    [dbo].[City] [City]
    LEFT OUTER JOIN [dbo].[Country] [Country] ON ([Country].[Id] = [City].[CountryId])
    INNER JOIN [dbo].[Country] [Country1] ON ([Country1].[Id] = [City].[CountryId])
    LEFT OUTER JOIN [dbo].[Country] [Country2] ON ([Country2].[Id] = [City].[CountryId])
    RIGHT OUTER JOIN [dbo].[Contact] [CreatedBy] ON ([CreatedBy].[Id] = [Country2].[CreatedById]
```

Если запрос содержит корневую схему и присоединяемые схемы, которые администрируются по записям, то можно применить права доступа текущего пользователя. Варианты применения прав доступа по записям к присоединенным схемам заданы перечислением `Terrasoft.Core.DB.QueryJoinRightLevel`.

Варианты применения прав доступа к присоединяемым схемам запроса:

- Всегда применяются.
- Применяются, если в присоединяемой схеме запроса используются колонки, отличные от первичной и первичной для отображения. Чаще всего это колонки `[Id]` и `[Name]`.
- Не применяются.

Порядок применения прав доступа определяется значением свойства `JoinRightState` запроса. Значение этого свойства по умолчанию определяется системной настройкой [*Способ администрирования связанных объектов*] (код `QueryJoinRightLevel`). Если значение этой системной настройки не задано, то права доступа применяются, если в присоединяемой схеме запроса используются колонки, отличные от первичной и первичной для отображения.

Управлять фильтрами в запросе

Фильтр — набор условий, применяемых при отображении данных запроса. В терминах SQL фильтр представляет собой отдельный предикат (условие) оператора `WHERE`.

Структура фильтра

```
Filter = {[AggregationType] {<LeftExpression> | <LeftExpressionColumnPath>}
    <ComparisonType>
    {{<RightExpression> | {<RightExpressionColumnPath>,...}} | {<Macros>, [MacrosValue]}}
}
```

Для создания простого фильтра в `EntitySchemaQuery` используется метод `CreateFilter()`, который возвращает экземпляр типа `EntitySchemaQueryFilter`. Для этого метода в `EntitySchemaQuery` реализован ряд перегрузок. Это позволяет создавать фильтры с разными исходными параметрами. В `EntitySchemaQuery` реализованы методы создания фильтров специального вида.

Экземпляр `EntitySchemaQuery` имеет свойство `Filters`, которое представляет собой коллекцию фильтров данного запроса (экземпляр класса `EntitySchemaQueryFilterCollection`). Экземпляр класса `EntitySchemaQueryFilterCollection` представляет собой типизированную коллекцию элементов `IEntitySchemaQueryFilterItem`.

Алгоритм добавления фильтра в запрос:

- Создайте экземпляр фильтра для данного запроса (метод `CreateFilter()`, методы создания фильтров специального вида).
- Добавьте созданный экземпляр фильтра в коллекцию фильтров запроса (метод `Add()` коллекции).

По умолчанию фильтры, добавляемые в коллекцию `Filters`, объединяются между собой логической операцией `AND`. При реализации логической операции `OR` необходимо использовать свойство `LogicalOperation` коллекции `Filters`. Это свойство принимает значения перечисления `LogicalOperationStrict` и позволяет указать логическую операцию, которой необходимо объединять фильтры.

В запросах `EntitySchemaQuery` реализована возможность управления фильтрами, участвующими в построении результирующего набора данных. Каждый элемент коллекции `Filters` имеет свойство `IsEnabled`, которое определяет, участвует ли данный элемент в построении результирующего запроса (`true` — участвует, `false` — не участвует). Аналогичное свойство `IsEnabled` также определено для коллекции `Filters`. Установив это свойство в `false`, можно отключить фильтрацию для запроса, при этом коллекция фильтров запроса останется неизменной. Таким образом, изначально сформировав коллекцию фильтров запроса, в дальнейшем можно использовать различные комбинации, не внося изменений в саму коллекцию.

Управлять сущностью базы данных

`Terrasoft.Core.Entities.Entity` — класс, который реализует работу с сущностью базы данных.

Назначение класса `Terrasoft.Core.Entities.Entity` — доступ к объекту, который является записью в таблице базы данных. Класс также можно использовать для CRUD-операций над соответствующими записями.

Управлять сущностями базы данных

Сложный

На заметку. Примеры 1-5, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

Пример 1

Пример. Получить значение колонки схемы [*City*] с именем `Name`.

Метод `GetEntityColumnData`

```
public string GetEntityColumnData()
{
    var result = "";
    /* Создание запроса к схеме City, добавление в запрос колонки Name. */
    var esqResult = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "City");
    var colName = esqResult.AddColumn("Name");
    /* Выполнение запроса к базе данных и получение объекта с заданным идентификатором. UId объе
```

```
    var entity = esqResult.GetEntity(UserConnection, new Guid("100B6B13-E8BB-DF11-B00F-001D60E93
    /* Получение значения колонки объекта. */
    result += entity.GetColumnValue(colName.Name).ToString();
    return result;
}
```

Пример 2

Пример. Получить коллекцию имен колонок схемы [*City*].

Метод `GetEntityColumns`

```
public IEnumerable<string> GetEntityColumns()
{
    /* Создание объекта строки данных схемы City (по идентификатору схемы, полученному из базы д
    var entity = new Entity(UserConnection, new Guid("5CA90B6A-93E7-4448-BEFE-AB5166EC2CFE"));
    /* Получение из базы данных объекта с заданным идентификатором. UId объекта можно получить и
    entity.FetchFromDB(new Guid("100B6B13-E8BB-DF11-B00F-001D60E938C6"),true);
    /* Получение коллекции имен колонок объекта. */
    var result = entity.GetColumnValueNames();
    return result;
}
```

Пример 3

Пример. Удалить из базы данных записи схемы [*Order*].

Метод `DeleteEntity`

```
public bool DeleteEntity()
{
    /* Создание запроса к схеме Order, добавление в запрос всех колонок схемы. */
    var esqResult = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "Order");
    esqResult.AddAllSchemaColumns();
    /* Выполнение запроса к базе данных и получение объекта с заданным идентификатором. UId объе
    var entity = esqResult.GetEntity(UserConnection, new Guid("e3bfa32f-3fe9-4bae-9332-16c162c51
    /* Удаление объекта из базы данных. */
    entity.Delete();
    /* Проверка, существует ли в базе данных объект с заданным идентификатором. */
    var result = entity.ExistInDB(new Guid("e3bfa32f-3fe9-4bae-9332-16c162c51e0d"));
```

```
    return result;
}
```

Пример 4

Пример. Изменить статус заказа.

Метод `UpdateEntity`

```
public bool UpdateEntity()
{
    /* Создание запроса к схеме Order, добавление в запрос всех колонок схемы. */
    var esqResult = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "Order");
    esqResult.AddAllSchemaColumns();
    /* Выполнение запроса к базе данных и получение объекта с заданным идентификатором. UId объе
    var entity = esqResult.GetEntity(UserConnection, new Guid("58be5223-715d-4b16-a5c4-e3d4ec041
    /* Создание объекта строки данных схемы OrderStatus. */
    var statusSchema = UserConnection.EntitySchemaManager.GetInstanceByName("OrderStatus");
    var newStatus = statusSchema.CreateEntity(UserConnection);
    /* Получение из базы данных объекта с заданным названием. */
    newStatus.FetchFromDB("Name", "4. Completed");
    /* Присваивает колонке StatusId новое значение. */
    entity.SetColumnValue("StatusId", newStatus.GetTypedColumnValue<Guid>("Id"));
    /* Сохранение измененного объекта в базе данных. */
    var result = entity.Save();
    return result;
}
```

Пример 5

Пример. Добавить город с указанным названием, привязав его к указанной стране.

Метод `UpdateEntity`

```
public bool InsertEntity(string city, string country)
{
    city = city ?? "unknown city";
    country = country ?? "unknown country";
    var citySchema = UserConnection.EntitySchemaManager.GetInstanceByName("City");
    var entity = citySchema.CreateEntity(UserConnection);
```

```
    entity.FetchFromDB("Name", city);
    /* Устанавливает для колонок объекта значения по умолчанию. */
    entity.SetDefColumnValues();
    var contryEntity = new Entity(UserConnection, new Guid("09FCE1F8-515C-4296-95CD-8CD93F79A6CF
    contryEntity.FetchFromDB("Name", country);
    /* Присваивает колонке Name переданное название города. */
    entity.SetColumnValue("Name", city);
    /* Присваивает колонке CountryId UId переданной страны. */
    entity.SetColumnValue("CountryId", contryEntity.GetTypedColumnValue<Guid>("Id"));
    var result = entity.Save();
    return result;
}
```

Пример 6

Пример. Создать контакт с именем "User01".

```
EntitySchema contactSchema = UserConnection.EntitySchemaManager.GetInstanceByName("Contact");
Entity contactEntity = contactSchema.CreateEntity(UserConnection);
contactEntity.SetDefColumnValues();
contactEntity.SetColumnValue("Name", "User01");
contactEntity.Save();
```

Пример 7

Пример. Изменить имя контакта на "User02".

```
EntitySchema entitySchema = UserConnection.EntitySchemaManager.GetInstanceByName("Contact");
Entity entity = entitySchema.CreateEntity(UserConnection);
if (!entity.FetchFromDB(some_id) {
    return false;
}
entity.SetColumnValue("Name", "User02");
return entity.Save();
```

Пример 8

Пример. Удалить контакт с именем "User02".

```
EntitySchema entitySchema = UserConnection.EntitySchemaManager.GetInstanceByName("Contact");
Entity entity = entitySchema.CreateEntity(UserConnection);
var fetchConditions = new Dictionary<string, object> {
    {"Name", "User02"}
};
if (entity.FetchFromDB(fetchConditions)) {
    entity.Delete();
}
```

Получить данные из базы данных с учетом прав пользователя

Сложный

На заметку. Примеры, приведенные в этой статье, реализованы в веб-сервисе. Пакет с реализацией веб-сервиса прикреплен в блоке "Ресурсы".

Пример 1

Пример. Создать экземпляр `EntitySchemaQuery`.

Метод `CreateESQ`

```
public string CreateESQ()
{
    var result = "";
    /* Получение экземпляра менеджера схем объектов. */
    EntitySchemaManager esqManager = SystemUserConnection.EntitySchemaManager;
    /* Получение экземпляра схемы, которая будет установлена в качестве корневой для создаваемог
    var rootEntitySchema = esqManager.GetInstanceByName("City") as EntitySchema;
    /* Создание экземпляра EntitySchemaQuery, у которого в качестве корневой схемы установлена с
    var esqResult = new EntitySchemaQuery(rootEntitySchema);
    /* Добавление колонок, которые будут выбираться в результирующем запросе. */
    esqResult.AddColumn("Id");
    esqResult.AddColumn("Name");
    /* Получение экземпляра Select, ассоциированного с созданным запросом EntitySchemaQuery. */
    Select selectEsq = esqResult.GetSelectQuery(SystemUserConnection);
```

```
    /* Получение текста результирующего запроса созданного экземпляра EntitySchemaQuery. */
    result = selectEsq.GetSqlText();
    return result;
}
```

Пример 2

Пример. Создать клон экземпляра `EntitySchemaQuery`.

Метод `CreateESQClone`

```
public string CreateESQClone()
{
    var result = "";
    EntitySchemaManager esqManager = SystemUserConnection.EntitySchemaManager;
    var esqSource = new EntitySchemaQuery(esqManager, "Contact");
    esqSource.AddColumn("Id");
    esqSource.AddColumn("Name");
    /* Создание экземпляра EntitySchemaQuery, являющегося клоном экземпляра esqSource. */
    var esqClone = new EntitySchemaQuery(esqSource);
    result = esqClone.GetSelectQuery(SystemUserConnection).GetSqlText();
    return result;
}
```

Пример 3

Пример. Получить результат выполнения запроса.

Метод `GetEntitiesExample`

```
public string GetEntitiesExample()
{
    var result = "";
    /* Создание запроса к схеме City, добавление в запрос колонки Name. */
    var esqResult = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "City");
    var colName = esqResult.AddColumn("Name");

    /* Выполнение запроса к базе данных и получение всей результирующей коллекции объектов. */
    var entities = esqResult.GetEntityCollection(UserConnection);
    for (int i=0; i < entities.Count; i++) {
```

```
        result += entities[i].GetColumnValue(colName.Name).ToString();
        result += "\n";
    }

    /* Выполнение запроса к базе данных и получение объекта с заданным идентификатором. */
    var entity = esqResult.GetEntity(UserConnection, new Guid("100B6B13-E8BB-DF11-B00F-001D60E93
    result += "\n";
    result += entity.GetColumnValue(colName.Name).ToString();
    return result;
}
```

Пример 4

Пример. Использовать кэш запроса `EntitySchemaQuery`.

Метод `UsingCacheExample`

```
public Collection<string> UsingCacheExample()
{
    /* Создание запроса к схеме City, добавление в запрос колонки Name. */
    var esqResult = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "City");
    esqResult.AddColumn("Name");

    /* Определение ключа, под которым в кэше будут храниться результаты выполнения запроса. В ка
    esqResult.CacheItemName = "EsqResultItem";

    /* Коллекция, в которую будут помещены результаты выполнения запроса. */
    var esqCityNames = new Collection<string>();

    /* Коллекция, в которую будут помещаться закэшированные результаты выполнения запроса. */
    var cachedEsqCityNames = new Collection<string>();

    /* Выполнение запроса к базе данных и получение результирующей коллекции объектов.
    После выполнения этой операции результаты запроса будут помещены в кэш. */
    var entities = esqResult.GetEntityCollection(UserConnection);

    /* Обработка результатов выполнения запроса и заполнение коллекции esqCityNames. */
    foreach (var entity in entities)
    {
        esqCityNames.Add(entity.GetTypedColumnValue<string>("Name"));
    }

    /* Получение ссылки на кэш запроса esqResult по ключу CacheItemName в виде таблицы данных в
    var esqCacheStore = esqResult.Cache[esqResult.CacheItemName] as DataTable;
```

```
    /* Заполнение коллекции cachedEsqCityNames значениями из кэша запроса. */
    if (esqCacheStore != null)
    {
        foreach (DataRow row in esqCacheStore.Rows)
        {
            cachedEsqCityNames.Add(row[0].ToString());
        }
    }
    return cachedEsqCityNames;
}
```

Пример 5

Пример. Использовать дополнительные настройки запроса `EntitySchemaQuery`.

Метод `UsingCacheExample`

```
public Collection<string> ESQOptionsExample()
{
    /* Создание экземпляра запроса с корневой схемой City. */
    var esqCities = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "City");
    esqCities.AddColumn("Name");

    /* Создание запроса с корневой схемой Country. */
    var esqCountries = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "Country");
    esqCountries.AddColumn("Name");

    /* Создание экземпляра настроек для возврата запросом первых 5 строк. */
    var esqOptions = new EntitySchemaQueryOptions()
    {
        PageableDirection = PageableSelectDirection.First,
        PageableRowCount = 5,
        PageableConditionValues = new Dictionary<string, object>()
    };

    /* Получение коллекции городов, которая будет содержать первые 5 городов результирующего наб
    var cities = esqCities.GetEntityCollection(UserConnection, esqOptions);

    /* Получение коллекции стран, которая будет содержать первые 5 стран результирующего набора
    var countries = esqCountries.GetEntityCollection(UserConnection, esqOptions);
        var esqStringCollection = new Collection<string>();
    foreach (var entity in cities)
    {
```

```
            esqStringCollection.Add(entity.GetTypedColumnValue<string>("Name"));
        }
        foreach (var entity in countries)
        {
            esqStringCollection.Add(entity.GetTypedColumnValue<string>("Name"));
        }
        return esqStringCollection;
    }
```

Класс EntitySchemaQuery C#

Сложный

Пространство имен `Terrasoft.Core.Entities`.

Класс `Terrasoft.Core.Entities.EntitySchemaQuery` предназначен для построения запросов выборки записей из таблиц базы данных с учетом прав доступа текущего пользователя. В результате создания и конфигурирования экземпляра этого класса будет построен запрос в базу данных приложения в виде SQL-выражения `SELECT`. В запрос можно добавить требуемые колонки, фильтры и условия ограничений.

На заметку. Полный перечень методов и свойств класса `EntitySchemaQuery`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaQuery(EntitySchema rootSchema)`

Создает экземпляр класса, в котором в качестве корневой схемы устанавливается переданный экземпляр `EntitySchema`. В качестве менеджера схем устанавливается менеджер переданного экземпляра корневой схемы.

`EntitySchemaQuery(EntitySchemaManager entitySchemaManager, string sourceSchemaName)`

Создает экземпляр класса с указанным `EntitySchemaManager` и корневой схемы с именем, переданным в качестве аргумента.

`EntitySchemaQuery(EntitySchemaQuery source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`Cache` `Terrasoft.Core.Store.ICacheStore`

Кэш запроса.

`CacheItemName` `string`

Имя элемента кэша.

`CanReadUncommitedData` `bool`

Определяет, попадут ли в результаты запроса данные, для которых не завершена транзакция.

`Caption` `Terrasoft.Common.LocalizableString`

Заголовок.

`ChunkSize` `int`

Количество строк запроса в одном чанке.

`Columns` `Terrasoft.Core.Entities.EntitySchemaQueryColumnCollection`

Коллекция колонок текущего запроса к схеме объекта.

`DataValueTypeManager` `DataValueTypeManager`

Менеджер значений типов данных.

`EntitySchemaManager` `Terrasoft.Core.Entities.EntitySchemaManager`

Менеджер схем объектов.

`Filters` `Terrasoft.Core.Entities.EntitySchemaQueryFilterCollection`

Коллекция фильтров текущего запроса к схеме объекта.

`HideSecurityValue` `bool`

Определяет, будут ли скрыты значения зашифрованных колонок.

`IgnoreDisplayValues` `bool`

Определяет, будут ли в запросе использоваться отображаемые значения колонок.

`IsDistinct` `bool`

Определяет, убирать ли дубли в результирующем наборе данных.

`IsInherited` `bool`

Определяет, является ли запрос унаследованным.

`JoinRightState` `QueryJoinRightLevel`

Определяет условие наложения прав при использовании связанных таблиц, если схема администрируется по записям.

`Manager` `Terrasoft.Core.IManager`

Менеджер схем.

`ManagerItem` `Terrasoft.Core.IManagerItem`

Элемент менеджера.

`Name` `string`

Имя.

`ParentCollection` `Terrasoft.Core.Entities.EntitySchemaQueryCollection`

Коллекция запросов, которой принадлежит текущий запрос к схеме объекта.

`ParentEntitySchema` `Terrasoft.Core.Entities.EntitySchema`

Родительская схема запроса.

`PrimaryQueryColumn` `Terrasoft.Core.Entities.EntitySchemaQueryColumn`

Колонка, созданная по первичной колонке корневой схемы. Заполняется при первом обращении.

`QueryOptimize` `bool`

Разрешает использование оптимизации запроса.

`RootSchema` `Terrasoft.Core.Entities.EntitySchema`

Корневая схема.

`RowCount` `int`

Количество строк, возвращаемых запросом.

`SchemaAliasPrefix` `string`

Префикс, используемый для создания псевдонимов схем.

`SkipRowCount` `int`

Количество строк, которые необходимо пропустить при возврате результата запроса.

`UseAdminRights` `bool`

Определяет будут ли учитываться права при построении запроса получения данных.

`UseLocalization` `bool`

Определяет, будут ли использоваться локализованные данные.

`UseOffsetFetchPaging` `bool`

Определяет возможность постраничного возврата результата запроса.

`UseRecordDeactivation` `bool`

Определяет, будут ли данные исключены из фильтрации.

`AdminUnitRoleSources` `int`

Целочисленное свойство, которое соответствует критериям фильтрации записей по источнику вхождения пользователя в роли. Значение по умолчанию: `0`.

Чтобы сформировать `AdminUnitRoleSources`, необходимо с помощью побитового `ИЛИ` "`|`" перечислить следующие константы из серверного класса `AdminUnitRoleSources`:

- `ExplicitEntry`.
- `Delegated`.
- `FuncRoleFromOrgRole`.
- `UpHierarchy`.
- `AsManager`.

- `All`.
- `None`.

В результате, отработает следующее правило: возвращать запись только, если у пользователя есть хоть одна роль, которой доступна запись, и пользователь входит в эту роль в соответствии с источниками, указанными в условиях фильтрации.

Методы

```
void AddAllSchemaColumns(bool skipSystemColumns)
```

В коллекцию колонок текущего запроса к схеме объекта добавляет все колонки корневой схемы.

```
EntitySchemaQueryColumn AddColumn(string columnPath, AggregationTypeStrict aggregationType, out
void AddColumn(EntitySchemaQueryColumn queryColumn)
EntitySchemaQueryColumn AddColumn(string columnPath)
EntitySchemaQueryColumn AddColumn(EntitySchemaQueryFunction function)
EntitySchemaQueryColumn AddColumn(object parameterValue, DataValueType parameterDataValueType)
EntitySchemaQueryColumn AddColumn(EntitySchemaQuery subQuery)
```

Создает и добавляет колонку в текущий запрос к схеме объекта.

Параметры

`columnPath`	Путь к колонке схемы относительно корневой схемы.
`aggregationType`	Тип агрегирующей функции. В качестве параметра передаются значения перечисления типов агрегирующей функции `Terrasoft.Common.AggregationTypeStrict`.
`subQuery`	Ссылка на созданный подзапрос, помещенный в колонку.
`queryColumn`	Экземпляр `EntitySchemaQueryColumn`, добавляемый в коллекцию колонок текущего запроса.
`function`	Экземпляр функции `EntitySchemaQueryFunction`.
`parameterValue`	Значение параметра, добавляемого в запрос в качестве колонки.
`parameterDataValueType`	Тип значения параметра, добавляемого в запрос в качестве колонки.

```
void ClearCache()
```

Очищает кэш текущего запроса.

```
static void ClearDefCache(string cacheItemName)
```

Удаляет из кэша запроса элемент с заданным именем `cacheItemName`.

```
object Clone()
```

Создает клон текущего экземпляра `EntitySchemaQuery`.

```
EntitySchemaQueryExpression CreateAggregationEntitySchemaExpression(string leftExprColumnPath, A
```

Возвращает выражение агрегирующей функции с заданным типом агрегации из перечисления `Terrasoft.Common.AggregationTypeStrict` для колонки, расположенной по заданному пути `leftExprColumnPath`.

```
static EntitySchemaQueryExpression CreateParameterExpression(object parameterValue)
static EntitySchemaQueryExpression CreateParameterExpression(object parameterValue, DataValueTyp
static EntitySchemaQueryExpression CreateParameterExpression(object parameterValue, string displ
```

Создает выражение для параметра запроса.

Параметры

`parameterValue`	Значение параметра.
`valueType`	Тип значения параметра.
`displayValue`	Значение для отображения параметра.

```
static IEnumerable CreateParameterExpressions(DataValueType valueType, params object[] parameter
static IEnumerable CreateParameterExpressions(DataValueType valueType, IEnumerable<object> param
```

Создает коллекцию выражений для параметров запроса с определенным типом данных `DataValueType`.

```
static EntitySchemaQueryExpression CreateSchemaColumnExpression(EntitySchemaQuery parentQuery, E
static EntitySchemaQueryExpression CreateSchemaColumnExpression(EntitySchema rootSchema, string
EntitySchemaQueryExpression CreateSchemaColumnExpression(string columnPath, bool useCoalesceFunc
```

Возвращает выражение колонки схемы объекта.

Параметры

parentQuery	Запрос к схеме объекта, для которого создается выражение колонки.
rootSchema	Корневая схема.
columnPath	Путь к колонке относительно корневой схемы.
useCoalesceFunctionForMultiLookup	Признак, использовать ли для колонки типа справочник функцию `COALESCE`. Необязательный параметр, по умолчанию равен `true`.
useDisplayValue	Признак, использовать ли для колонки значение для отображения. Необязательный параметр, по умолчанию равен `false`.

```
Enumerable CreateSchemaColumnExpressions(params string[] columnPaths)
IEnumerable CreateSchemaColumnExpressions(IEnumerable columnPaths, bool useCoalesceFunctionForMu
```

Возвращает коллекцию выражений колонок запроса к схеме объекта по заданной коллекции путей к колонкам `columnPaths`.

```
IEnumerable CreateSchemaColumnExpressionsWithoutCoalesce(params string[] columnPaths)
```

Возвращает коллекцию выражений колонок запроса к схеме объекта по заданному массиву путей к колонкам. При этом, если колонка имеет тип множественный справочник, к ее значениям не применяется функция `COALESCE`.

```
static EntitySchemaQueryExpression CreateSchemaColumnQueryExpression(string columnPath, EntitySc
static EntitySchemaQueryExpression CreateSchemaColumnQueryExpression(string columnPath, EntitySc
```

Возвращает выражение запроса к схеме объекта по заданным пути к колонке, корневой схеме и экземпляру колонки схемы. При этом для колонки можно определить, какой тип ее значения использовать в выражении — хранимое значение или значение для отображения.

```
EntitySchemaQueryExpression CreateSubEntitySchemaExpression(string leftExprColumnPath)
```

Возвращает выражение подзапроса к схеме объекта для колонки, расположенной по заданному пути `leftExprColumnPath`.

```
EntitySchemaAggregationQueryFunction CreateAggregationFunction(AggregationTypeStrict aggregatior
```

Возвращает экземпляр агрегирующей функции `EntitySchemaAggregationQueryFunction` с заданным типом агрегации из перечисления `Terrasoft.Common.AggregationTypeStrict` для колонки по указанному пути относительно корневой схемы `columnPath`.

```
EntitySchemaCaseNotNullQueryFunction CreateCaseNotNullFunction(params EntitySchemaCaseNotNullQue
```

Возвращает экземпляр `CASE`-функции `EntitySchemaCaseNotNullQueryFunction` для заданного массива выражений условий `EntitySchemaCaseNotNullQueryFunctionWhenItem`.

```
EntitySchemaCaseNotNullQueryFunctionWhenItem CreateCaseNotNullQueryFunctionWhenItem(string whenC
```

Возвращает экземпляр выражения для sql-конструкции вида `WHEN <Выражение_1> IS NOT NULL THEN <Выражение_2>`.

Параметры

whenColumnPath	Путь к колонке, содержащей выражение предложения `WHEN`.
thenParameterValue	Путь к колонке, содержащей выражение предложения `THEN`.

```
EntitySchemaCastQueryFunction CreateCastFunction(string columnPath, DBDataValueType castType)
```

Возвращает экземпляр `CAST`-функции `EntitySchemaCastQueryFunction` для выражения колонки, расположенной по заданному пути относительно корневой схемы `columnPath`, и указанным целевым типом данных `DBDataValueType`.

```
EntitySchemaCoalesceQueryFunction CreateCoalesceFunction(params string[] columnPaths)
static EntitySchemaCoalesceQueryFunction CreateCoalesceFunction(EntitySchemaQuery parentQuery, E
static EntitySchemaCoalesceQueryFunction CreateCoalesceFunction(EntitySchema rootSchema, params
```

Возвращает экземпляр функции, возвращающей первое не равное `null` выражение из списка аргументов, для заданных колонок.

Параметры

columnPaths	Массив путей к колонкам относительно корневой схемы.
parentQuery	Запрос к схеме объекта, для которого создается экземпляр функции.
rootSchema	Корневая схема.

```
EntitySchemaConcatQueryFunction CreateConcatFunction(params EntitySchemaQueryExpression[] expres
```

Возвращает экземпляр функции для формирования строки, являющейся результатом объединения строковых значений аргументов функции для заданного массива выражений `EntitySchemaQueryExpression`.

`EntitySchemaDatePartQueryFunction CreateDatePartFunction(EntitySchemaDatePartQueryFunctionInterv`

Возвращает экземпляр `DATEPART`-функции `EntitySchemaDatePartQueryFunction`, определяющей интервал даты, заданный перечислением `EntitySchemaDatePartQueryFunctionInterval` (месяц, день, час, год, день недели...), для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateDayFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, определяющей интервал даты [*День*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateHourFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, возвращающей часть даты [*Час*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateHourMinuteFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, возвращающей часть даты [*Минута*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateMonthFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, возвращающей часть даты [*Месяц*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateWeekdayFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, возвращающей часть даты [*День недели*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateWeekFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, возвращающей часть даты [*Неделя*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaDatePartQueryFunction CreateYearFunction(string columnPath)`

Возвращает экземпляр функции `EntitySchemaDatePartQueryFunction`, возвращающей часть даты [*Год*] для значения колонки, расположенной по указанному пути относительно корневой схемы.

`EntitySchemaIsNullQueryFunction CreateIsNullFunction(string checkColumnPath, string replacementC`

Возвращает экземпляр функции `EntitySchemaIsNullQueryFunction` для колонок с проверяемым и замещающим значениями, которые расположены по заданным путям относительно корневой схемы.

```
EntitySchemaLengthQueryFunction CreateLengthFunction(string columnPath)
EntitySchemaLengthQueryFunction CreateLengthFunction(params EntitySchemaQueryExpression[] expres
```

Создание экземпляра функции `LEN` (функция для возврата длины выражения) для выражения колонки по заданному пути относительно корневой схемы или для заданного массива выражений.

```
EntitySchemaTrimQueryFunction CreateTrimFunction(string columnPath)
EntitySchemaTrimQueryFunction CreateTrimFunction(params EntitySchemaQueryExpression[] expressior
```

Возвращает экземпляр функции `TRIM` (функция для удаления начальных и конечных пробелов из выражения) для выражения колонки по заданному пути относительно корневой схемы или для заданного массива выражений.

```
EntitySchemaUpperQueryFunction CreateUpperFunction(string columnPath)
```

Возвращает экземпляр функции `EntitySchemaUpperQueryFunction`, для преобразования символов выражения аргумента к верхнему регистру, для выражения колонки по заданному пути относительно корневой схемы.

```
EntitySchemaCurrenddateQueryFunction CreateCurrenddateFunction()
```

Возвращает экземпляр функции `EntitySchemaCurrenddateQueryFunction`, определяющей текущую дату.

```
EntitySchemaCurrenddateTimeQueryFunction CreateCurrenddateTimeFunction()
```

Возвращает экземпляр функции `EntitySchemaCurrenddateTimeQueryFunction`, определяющей текущие дату и время.

```
EntitySchemaCurrentTimeQueryFunction CreateCurrentTimeFunction()
```

Возвращает экземпляр функции `EntitySchemaCurrentTimeQueryFunction`, определяющей текущее время.

```
EntitySchemaCurrentUserAccountQueryFunction CreateCurrentUserAccountFunction()
```

Возвращает экземпляр функции `EntitySchemaCurrentUserAccountQueryFunction`, определяющей идентификатор контрагента текущего пользователя.

```
EntitySchemaCurrentUserContactQueryFunction CreateCurrentUserContactFunction()
```

Возвращает экземпляр функции `EntitySchemaCurrentUserContactQueryFunction`, определяющей

идентификатор контакта текущего пользователя.

`EntitySchemaCurrentUserQueryFunction CreateCurrentUserFunction()`

Возвращает экземпляр функции `EntitySchemaCurrentUserQueryFunction`, определяющей текущего пользователя.

`EntitySchemaQueryFilter CreateExistsFilter(string rightExpressionColumnPath)`

Создает фильтр сравнения типа [*Существует по заданному условию*] и устанавливает в качестве проверяемого значения выражение колонки, расположенной по пути `rightExpressionColumnPath`.

```
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, EntitySchemaQuery
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, EntitySchemaQuery
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, EntitySchemaQuery
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExpres
EntitySchemaQueryFilter CreateFilter(FilterComparisonType comparisonType, string leftExprColumnP
EntitySchemaQueryFilter CreateFilter(FilterComparisonType comparisonType, string leftExprColumnP
EntitySchemaQueryFilter CreateFilter(FilterComparisonType comparisonType, string leftExprColumnP
EntitySchemaQueryFilter CreateFilter(FilterComparisonType comparisonType, string leftExprColumnP
EntitySchemaQueryFilter CreateFilter(FilterComparisonType comparisonType, string leftExprColumnP
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExprCo
IEntitySchemaQueryFilterItem CreateFilter(FilterComparisonType comparisonType, string leftExprCo
```

Создает фильтр запроса для выборки записей по определенным условиям.

Параметры

comparisonType	Тип сравнения из перечисления `Terrasoft.Core.Entities.FilterComparisonType`.
leftExpressionColumnPath	Путь к колонке, содержащей выражение левой части фильтра.
leftExpression	Выражение в левой части фильтра.
leftExprAggregationType	Тип агрегирующей функции.
leftExprSubQuery	Параметр, в котором возвращается подзапрос для выражения в левой части фильтра (если он не равен `null`) либо подзапрос для первого выражения в правой части фильтра (если выражение левой части фильтра равно `null`).
rightExpressionColumnPaths	Массив путей к колонкам, содержащим выражения правой части фильтра.
rightExpression	Выражение в правой части фильтра.
rightExpressionValue	Экземпляр функции выражения в правой части фильтра (тип параметра `EntitySchemaQueryFunction`) или выражение подзапроса в правой части фильтра (тип параметра `EntitySchemaQuery`).
rightValue	Значение, которое обрабатывается макросом в правой части фильтра.
rightExprParameterValue	Значение параметра, к которому применяется агрегирующая функция в правой части фильтра.
macrosType	Тип макроса из перечисления `Terrasoft.Core.Entities.EntitySchemaQueryMacrosType`.
daysCount	Значение, к которому применяется макрос в правой части фильтра. Необязательный параметр, по умолчанию равен 0.

```
IEntitySchemaQueryFilterItem CreateFilterWithParameters(FilterComparisonType comparisonType, boo
IEntitySchemaQueryFilterItem CreateFilterWithParameters(FilterComparisonType comparisonType, str
IEntitySchemaQueryFilterItem CreateFilterWithParameters(FilterComparisonType comparisonType, str
static IEntitySchemaQueryFilterItem CreateFilterWithParameters(EntitySchemaQuery parentQuery, En
static IEntitySchemaQueryFilterItem CreateFilterWithParameters(EntitySchema rootSchema, FilterCo
```

Создает параметризированный фильтр для выборки записей по определенным условиям.

Параметры

`parentQuery`	Родительский запрос, для которого создается фильтр.
`rootSchema`	Корневая схема.
`comparisonType`	Тип сравнения из перечисления `Terrasoft.Core.Entities.FilterComparisonType`.
`useDisplayValue`	Признак типа значения колонки, которое используется в фильтре: `true` - значение для отображения; `false` - хранимое значение.
`leftExpressionColumnPath`	Путь к колонке, содержащей выражение левой части фильтра.
`rightExpressionParameterValues`	Коллекция выражений параметров в правой части фильтра.

`IEntitySchemaQueryFilterItem CreateIsNotNullFilter(string leftExpressionColumnPath)`

Создает фильтр сравнения типа [*Не является null в базе данных*], устанавливая в качестве проверяемого значения выражение колонки, расположенной по указанному в параметре `leftExpressionColumnPath` пути.

`IEntitySchemaQueryFilterItem CreateIsNullFilter(string leftExpressionColumnPath)`

Создает фильтр сравнения типа [*Является null в базе данных*], устанавливая в качестве условия проверки выражение колонки, расположенной по указанному в параметре `leftExpressionColumnPath` пути.

`EntitySchemaQueryFilter CreateNotExistsFilter(string rightExpressionColumnPath)`

Создает фильтр сравнения типа [*Не существует по заданному условию*] и устанавливает в качестве проверяемого значения выражение колонки, расположенной по пути `rightExpressionColumnPath`.

`DataTable GetDataTable(UserConnection userConnection)`

Возвращает результат выполнения текущего запроса к схеме объекта в виде таблицы данных в памяти, используя пользовательское подключение `UserConnection`.

`static int GetDayOfWeekNumber(UserConnection userConnection, DayOfWeek dayOfWeek)`

Возвращает порядковый номер дня недели для объекта `System.DayOfWeek` с учетом региональных

настроек.

```
Entity GetEntity(UserConnection userConnection, object primaryColumnValue)
```

Возвращает экземпляр Entity по первичному ключу `primaryColumnValue`, используя пользовательское подключение `UserConnection`.

```
EntityCollection GetEntityCollection(UserConnection userConnection, EntitySchemaQueryOptions opt
EntityCollection GetEntityCollection(UserConnection userConnection)
```

Возвращает коллекцию экземпляров `Entity`, представляющих результаты выполнения текущего запроса, используя пользовательское подключение `UserConnection` и заданные дополнительные настройки запроса `EntitySchemaQueryOptions`.

```
EntitySchema GetSchema()
```

Возвращает экземпляр схемы объекта `EntitySchema` текущего экземпляра `EntitySchemaQuery`.

```
Select GetSelectQuery(UserConnection userConnection)
Select GetSelectQuery(UserConnection userConnection, EntitySchemaQueryOptions options)
```

Возвращает экземпляр запроса на выборку данных, используя пользовательское подключение `UserConnection` и заданные дополнительные настройки запроса `EntitySchemaQueryOptions`.

```
EntitySchemaQueryColumnCollection GetSummaryColumns()
EntitySchemaQueryColumnCollection GetSummaryColumns(IEnumerable<string> columnNames)
```

Возвращает коллекцию выражений колонок запроса, для которых вычисляются итоговые значения.

```
Entity GetSummaryEntity(UserConnection userConnection, EntitySchemaQueryColumnCollection summary
Entity GetSummaryEntity(UserConnection userConnection)
Entity GetSummaryEntity(UserConnection userConnection, IEnumerable<string> columnNames)
Entity GetSummaryEntity(UserConnection userConnection, params string[] columnNames)
```

Возвращает экземпляр `Entity` для результата, возвращаемого запросом на выборку итоговых значений.

Параметры

`userConnection`	Пользовательское подключение.
`summaryColumns`	Коллекция колонок запроса, для которых выбираются итоговые значения.
`columnNames`	Коллекция имен колонок.

```
Select GetSummarySelectQuery(UserConnection userConnection, EntitySchemaQueryColumnCollection su
Select GetSummarySelectQuery(UserConnection userConnection)
Select GetSummarySelectQuery(UserConnection userConnection, IEnumerable<string> columnNames)
Select GetSummarySelectQuery(UserConnection userConnection, params string[] columnNames)
```

Строит запрос на выборку итоговых значений для заданной коллекции колонок текущего экземпляра `EntitySchemaQuery`.

Параметры

`userConnection`	Пользовательское подключение.
`summaryColumns`	Коллекция колонок запроса, для которых выбираются итоговые значения.
`columnNames`	Коллекция имен колонок.

```
T GetTypedColumnValue(Entity entity, string columnName)
```

Возвращает типизированное значение колонки с именем `columnName` из переданного экземпляра `Entity`.

```
void LoadDataTableData(UserConnection userConnection, DataTable dataTable)
void LoadDataTableData(UserConnection userConnection, DataTable dataTable, EntitySchemaQueryOpti
```

Загружает результат выполнения текущего запроса к схеме объекта в объект `System.Data.DataTable`, используя пользовательское подключение `UserConnection` и заданные дополнительные настройки запроса `EntitySchemaQueryOptions`.

```
void RemoveColumn(string columnName)
```

Удаляет колонку с именем `columnName` из коллекции колонок текущего запроса.

```
void ResetSchema()
```

Очищает схему текущего экземпляра `EntitySchemaQuery`.

`void ResetSelectQuery()`

Очищает запрос на выборку для текущего запроса к схеме объекта.

`void SetLocalizationCultureId(System.Guid cultureId)`

Устанавливает идентификатор локальной культуры.

Класс Entity C#

Сложный

Пространство имен `Terrasoft.Core.Entities`.

Класс `Terrasoft.Core.Entities.Entity` предназначен для доступа к объекту, который представляет собой запись в таблице базы данных.

На заметку. Полный перечень методов и свойств класса `Entity`, его родительских классов, а также реализуемых им интерфейсов, можно найти в Библиотеке .NET классов.

Конструкторы

`Entity(UserConnection userConnection)`

Создает новый экземпляр класса `Entity` для заданного пользовательского подключения `UserConnection`.

`Entity(UserConnection userConnection, Guid schemaUId)`

Создает новый экземпляр класса `Entity` для заданного пользовательского подключения `UserConnection` и схемы заданной идентификатором `schemaUId`.

`Entity(Entity source)`

Создает экземпляр класса, являющийся клоном экземпляра, переданного в качестве аргумента.

Свойства

`ChangeType` `EntityChangeType`

Тип изменения состояния объекта (добавлен, изменен, удален, без изменений).

`EntitySchemaManager` `EntitySchemaManager`

Экземпляр менеджера схемы объекта.

`EntitySchemaManagerName` `string`

Имя менеджера схемы объекта.

`HasColumnValues` `bool`

Определяет, имеет ли объект хотя бы одну колонку.

`HierarchyColumnValue` `Guid`

Значение колонки связи с родительской записью для иерархических объектов.

`InstanceUId` `Guid`

Идентификатор экземпляра объекта.

`IsDeletedFromDB` `bool`

Определяет, удален ли объект из базы данных.

`IsInColumnValueChanged` `bool`

Определяет, выполняется ли обработка события `ColumnValueChanged`.

`IsInColumnValueChanging` `bool`

Определяет, выполняется ли обработка события `ColumnValueChanging`.

`IsInDefColumnValuesSet` `bool`

Определяет, выполняется ли обработка события `DefColumnValuesSet`.

`IsInDeleted` `bool`

Определяет, выполняется ли обработка события `Deleted`.

`IsInDeleting` `bool`

Определяет, выполняется ли обработка события `Deleting`.

`IsInInserted` `bool`

Определяет, выполняется ли обработка события `Inserted`.

`IsInInserting` `bool`

Определяет, выполняется ли обработка события `Inserting`.

`IsInLoaded` `bool`

Определяет, выполняется ли обработка события `Loaded`.

`IsInLoading` `bool`

Определяет, выполняется ли обработка события `Loading`.

`IsInSaved` `bool`

Определяет, выполняется ли обработка события `Saved`.

`IsInSaveError` `bool`

Определяет, выполняется ли обработка события `SaveError`.

`IsInSaving` `bool`

Определяет, выполняется ли обработка события `Saving`.

`IsInUpdated` `bool`

Определяет, выполняется ли обработка события `Updated`.

`IsInUpdating` `bool`

Определяет, выполняется ли обработка события `Updating`.

`IsInValidating` `bool`

Определяет, выполняется ли обработка события `Validating`.

`IsSchemaInitialized` `bool`

Определяет, является ли схема объекта проинициализированной.

`LicOperationPrefix` `string`

Префикс лицензируемой операции.

`LoadState` `EntityLoadState`

Состояние загрузки объекта.

`PrimaryColumnValue` `Guid`

Идентификатор первичной колонки.

`PrimaryDisplayColumnValue` `string`

Значение для отображения первичной колонки.

`Process` `Process`

Встроенный процесс объекта.

`Schema` `EntitySchema`

Экземпляр схемы объекта.

`SchemaName` `string`

Имя схемы объекта.

`StoringState` `StoringObjectState`

Состояние объекта (изменен, добавлен, удален, без изменений).

`UseAdminRights` `bool`

Определяет, будут ли учитываться права при вставке, обновлении, удалении и получении данных.

`UseDefRights` `bool`

Определяет, использовать ли права по умолчанию на объект.

`UseLazyLoad` `bool`

Определяет, использовать ли ленивую первоначальную загрузку данных объекта.

`UserConnection` `UserConnection`

Пользовательское подключение.

`ValidationMessages` `EntityValidationMessageCollection`

Коллекция сообщений, выводимых при проверке объекта.

`ValueListSchemaManager` `ValueListSchemaManager`

Экземпляр менеджера перечислений объекта.

`ValueListSchemaManagerName` `string`

Имя менеджера перечислений объекта.

Методы

```
void AddDefRights()
void AddDefRights(Guid primaryColumnValue)
void AddDefRights(IEnumerable<Guid> primaryColumnValues)
```

Для данного объекта устанавливает права по умолчанию.

Параметры

`primaryColumnValue`	Идентификатор значения права доступа.
`primaryColumnValues`	Массив идентификаторов значений прав доступа.

```
virtual object Clone()
```

Создает клон текущего экземпляра `Entity`.

```
Insert CreateInsert(bool skipLookupColumnValues)
```

Создает запрос на добавление данных в базу.

Параметры

skipLookupColumnValues	Признак добавления данных с учетом справочных колонок. По умолчанию установлено значение `false`.

```
Update CreateUpdate(bool skipLookupColumnValues)
```

Создает запрос на обновление данных в базе.

Параметры

skipLookupColumnValues	Параметр, определяющий необходимость добавления в базу данных колонок типа справочник. Если параметр равен `true`, то колонки типа справочник не будут добавлены в базу. Значение по умолчанию — `false`.

```
virtual bool Delete()
virtual bool Delete(object keyValue)
```

Удаляет из базы данных запись объекта.

Параметры

keyValue	Значение ключевого поля.

```
bool DeleteWithCancelProcess()
```

Удаляет из базы данных запись объекта и отменяет запущенный процесс.

```
static Entity DeserializeFromJson(UserConnection userConnection, string jsonValue)
```

Создает объект типа `Entity`, используя пользовательское подключение `userConnection`, и заполняет значения его полей из указанной строки формата JSON `jsonValue`.

Параметры

jsonValue	Строка формата JSON.
userConnection	Пользовательское подключение.

```
bool ExistInDB(EntitySchemaColumn conditionColumn, object conditionValue)
bool ExistInDB(string conditionColumnName, object conditionValue)
bool ExistInDB(object keyValue)
```

```
bool ExistInDB(Dictionary<string,object> conditions)
```

Определяет, существует ли в базе данных запись, отвечающая заданному условию запроса `conditionValue` к заданной колонке схемы объекта `conditionColumn` либо с заданным первичным ключом `keyValue`.

Параметры

`conditionColumn`	Колонка, для которой задается условие выборки.
`conditionColumnName`	Название колонки, для которой задается условие выборки.
`conditionValue`	Значение колонки условия для выбираемых данных.
`conditions`	Набор условий фильтрации выборки записей объекта.
`keyValue`	Значение ключевого поля.

```
bool FetchFromDB(EntitySchemaColumn conditionColumn, object conditionValue, bool useDisplayValue
bool FetchFromDB(string conditionColumnName, object conditionValue, bool useDisplayValues)
bool FetchFromDB(object keyValue, bool useDisplayValues)
bool FetchFromDB(Dictionary<string,object> conditions, bool useDisplayValues)
bool FetchFromDB(EntitySchemaColumn conditionColumn, object conditionValue, IEnumerable<EntitySc
bool FetchFromDB(string conditionColumnName, object conditionValue, IEnumerable<string>columnNam
```

По заданному условию загружает объект из базы данных.

Параметры

`conditionColumn`	Колонка, для которой задается условие выборки.
`conditionColumnName`	Название колонки, для которой задается условие выборки.
`conditionValue`	Значение колонки условия для выбираемых данных.
`columnsToFetch`	Список колонок, которые будут выбраны.
`columnNamesToFetch`	Список названий колонок, которые будут выбраны.
`conditions`	Набор условий фильтрации выборки записей объекта.
`keyValue`	Значение ключевого поля.
`useDisplayValues`	Признак получения в запросе первичных отображаемых значений. Если параметр равен `true`, в запросе будут возвращены первичные отображаемые значения.

```
bool FetchPrimaryColumnFromDB(object keyValue)
```

По заданному условию `keyValue` загружает из базы данных объект с первичной колонкой.

Параметры

keyValue	Значение ключевого поля.

```
bool FetchPrimaryInfoFromDB(EntitySchemaColumn conditionColumn, object conditionValue)
bool FetchPrimaryInfoFromDB(string conditionColumnName, object conditionValue)
```

По заданному условию загружает из базы данных объект с первичными колонками, включая колонку, первичную для отображения.

Параметры

conditionColumn	Колонка, для которой задается условие выборки.
conditionColumnName	Название колонки, для которой задается условие выборки.
conditionValue	Значение колонки условия для выбираемых данных.

```
byte[] GetBytesValue(string valueName)
```

Возвращает значение заданной колонки объекта в виде массива байт.

Параметры

valueName	Имя колонки объекта.

```
IEnumerable<EntityColumnValue> GetChangedColumnValues()
```

Возвращает коллекцию имен колонок объекта, которые были изменены.

```
string GetColumnDisplayValue(EntitySchemaColumn column)
```

Возвращает значение для отображения свойства объекта, соответствующее заданной колонке схемы объекта.

Параметры

column	Определенная колонка схемы объекта.

```
object GetColumnOldValue(string valueName)
object GetColumnOldValue(EntitySchemaColumn column)
```

Возвращает предыдущее значение заданного свойства объекта.

Параметры

column	Определенная колонка схемы объекта.
valueName	Имя колонки объекта.

```
virtual object GetColumnValue(string valueName)
virtual object GetColumnValue(EntitySchemaColumn column)
```

Возвращает значение колонки объекта с заданным именем, соответствующее переданной колонке схемы объекта.

Параметры

column	Определенная колонка схемы объекта.
valueName	Имя колонки объекта.

```
IEnumerable<string> GetColumnValueNames()
```

Возвращает коллекцию имен колонок объекта.

```
virtual bool GetIsColumnValueLoaded(string valueName)
bool GetIsColumnValueLoaded(EntitySchemaColumn column)
```

Возвращает признак, определяющий, загружено ли заданное свойство объекта.

Параметры

column	Определенная колонка схемы объекта.
valueName	Имя колонки объекта.

```
virtual MemoryStream GetStreamValue(string valueName)
```

Возвращает преобразованное в экземпляр типа `System.IO.MemoryStream` значение переданной колонки схемы объекта.

Параметры

`valueName`	Имя колонки объекта.

```
TResult GetTypedColumnValue<TResult>(EntitySchemaColumn column)
```

Возвращает типизированное значение свойства объекта, соответствующее заданной колонке схемы объекта.

Параметры

`column`	Определенная колонка схемы объекта.

```
TResult GetTypedOldColumnValue<TResult>(string valueName)
TResult GetTypedOldColumnValue<TResult>(EntitySchemaColumn column)
```

Возвращает типизированное предыдущее значение свойства объекта, соответствующее заданной колонке схемы объекта.

Параметры

`column`	Определенная колонка схемы объекта.
`valueName`	Имя колонки объекта.

```
virtual bool InsertToDB(bool skipLookupColumnValues, bool validateRequired)
```

Добавляет запись текущего объекта в базу данных.

Параметры

`skipLookupColumnValues`	Параметр, определяющий необходимость добавления в базу данных колонок типа справочник. Если параметр равен `true`, то колонки типа справочник не будут добавлены в базу. Значение по умолчанию — `false`.
`validateRequired`	Параметр, определяющий необходимость проверки заполнения обязательных значений. Значение по умолчанию — `true`.

```
bool IsColumnValueLoaded(string valueName)
bool IsColumnValueLoaded(EntitySchemaColumn column)
```

Определяет, загружено ли значение свойства объекта с заданным именем.

Параметры

column	Определенная колонка схемы объекта.
valueName	Имя колонки объекта.

```
virtual bool Load(DataRow dataRow)
virtual bool Load(DataRow dataRow, Dictionary<string,string> columnMap)
virtual bool Load(IDataReader dataReader)
virtual bool Load(IDataReader dataReader, IDictionary<string,string> columnMap)
virtual bool Load(object dataSource)
virtual bool Load(object dataSource, IDictionary<string,string> columnMap)
```

Заполняет объект переданными данными.

Параметры

columnMap	Свойства объекта, заполняемые данными.
dataRow	Экземпляр `System.Data.DataRow`, из которого загружаются данные в объект.
dataReader	Экземпляр `System.Data.IDataReader`, из которого загружаются данные.
dataSource	Экземпляр `System.Object`, из которого загружаются данные.

```
void LoadColumnValue(string columnValueName, IDataReader dataReader, int fieldIndex, int binaryP
void LoadColumnValue(string columnValueName, IDataReader dataReader, int fieldIndex)
void LoadColumnValue(string columnValueName, object value)
void LoadColumnValue(EntitySchemaColumn column, object value)
```

Для свойства с заданным именем загружает его значение из переданного экземпляра.

Параметры

`binaryPackageSize`	Размер загружаемого значения.
`column`	Колонка схемы объекта.
`columnValueName`	Имя свойства объекта.
`dataReader`	Экземпляр `System.Data.IDataReader`, из которого загружается значение свойства.
`fieldIndex`	Индекс загружаемого из `System.Data.IDataReader` поля.
`value`	Загружаемое значение свойства.

```
static Entity Read(UserConnection userConnection, DataReader dataReader)
```

Возвращает значение текущего свойства типа `Entity` из потока ввода.

Параметры

`dataReader`	Экземпляр `System.Data.IDataReader`, из которого загружается значение свойства.
`userConnection`	Пользовательское подключение.

```
void ReadData(DataReader reader)
void ReadData(DataReader reader, EntitySchema schema)
```

Считывает данные из схемы объекта в заданный объект типа `System.Data.IDataReader`.

Параметры

`reader`	Экземпляр `System.Data.IDataReader`, в который загружаются данные схемы объекта.
`schema`	Схема объекта.

```
void ResetColumnValues()
```

Для всех свойств объекта отменяет изменения.

```
void ResetOldColumnValues()
```

Для всех свойств объекта отменяет изменения, устанавливая предыдущее значение.

```
bool Save(bool validateRequired)
```

Сохраняет объект в базе данных.

Параметры

`validateRequired`	Определяет необходимость проверки заполнения обязательных значений. Значение по умолчанию — `true`.

```
static string SerializeToJson(Entity entity)
```

Преобразует объект `entity` в строку формата `JSON`.

Параметры

`entity`	Экземпляр `Entity`.

```
virtual void SetBytesValue(string valueName, byte[] streamBytes)
```

Устанавливает для заданного свойства объекта переданное значение типа `System.Byte`.

Параметры

`streamBytes`	Значение типа `System.Byte`, которое устанавливается в заданную колонку объекта.
`valueName`	Имя колонки объекта.

```
bool SetColumnBothValues(EntitySchemaColumn column, object value, string displayValue)
bool SetColumnBothValues(string columnValueName, object value, string displayColumnValueName, st
```

Устанавливает свойству объекта, соответствующему заданной колонке схемы, переданные значение `value` и значение для отображения `displayValue`.

Параметры

`column`	Колонка схемы объекта.
`displayValue`	Загружаемое значение для отображения.
`displayColumnValueName`	Имя колонки, содержащей значение для отображения.
`value`	Загружаемое значение колонки.

```
bool SetColumnValue(string valueName, object value)
bool SetColumnValue(EntitySchemaColumn column, object value)
```

Устанавливает заданной колонке схемы переданное значение `value`.

Параметры

`column`	Колонка схемы объекта.
`value`	Загружаемое значение колонки.
`valueName`	Имя колонки объекта.

```
void SeddefColumnValue(string columnValueName, object defValue)
void SeddefColumnValue(string columnValueName)
```

Устанавливает значение по умолчанию свойству с заданным именем.

Параметры

`columnValueName`	Имя колонки объекта.
`defValue`	Значение по умолчанию.

```
void SeddefColumnValues()
```

Для всех свойств объекта устанавливает значения по умолчанию.

```
bool SetStreamValue(string valueName, Stream value)
```

Устанавливает для заданного свойства объекта переданное значение типа `System.IO.Stream`.

Параметры

`value`	Загружаемое значение колонки.
`valueName`	Имя колонки объекта.

```
virtual bool UpdateInDB(bool validateRequired)
```

Обновляет запись объекта в базе данных.

Параметры

`validateRequired`	Определяет необходимость проверки заполнения обязательных значений. Значение по умолчанию — `true`.

```
bool Validate()
```

Проверяет заполнение обязательных полей.

```
static void Write(DataWriter dataWriter, Entity entity, string propertyName)
static void Write(DataWriter dataWriter, Entity entity, string propertyName, bool couldConvertFo
```

Осуществляет запись значения типа `Entity` в поток вывода с заданными именем.

Параметры

`couldConvertForXml`	Разрешить преобразование для xml-сериализации.
`dataWriter`	Экземпляр класса `Terrasoft.Common.DataWriter`, предоставляющий методы последовательной записи значений в поток вывода.
`entity`	Значение для записи типа `Entity`.
`propertyName`	Имя объекта.

```
void Write(DataWriter dataWriter, string propertyName)
```

Осуществляет запись данных в поток вывода с заданным именем.

Параметры

dataWriter	Экземпляр класса `Terrasoft.Common.DataWriter`, предоставляющий методы последовательной записи значений в поток вывода.
propertyName	Имя свойства.

```
void WriteData(DataWriter writer)
void WriteData(DataWriter writer, EntitySchema schema)
```

Осуществляет запись в поток вывода для указанной либо текущей схемы объекта.

Параметры

schema	Схема объекта.
writer	Экземпляр класса `Terrasoft.Common.DataWriter`, предоставляющий методы последовательной записи значений в поток вывода.

События

```
event EventHandler<EntityColumnAfterEventArgs> ColumnValueChanged
```

Обработчик события, возникающего после изменения значения колонки объекта.

Обработчик события получает аргумент типа `EntityColumnAfterEventArgs`.

Свойства `EntityColumnAfterEventArgs` предоставляющие сведения, относящиеся к событию:

- `ColumnValueName`;
- `DisplayColumnValueName`.

```
event EventHandler<EntityColumnBeforeEventArgs> ColumnValueChanging
```

Обработчик события, возникающего перед изменением значения колонки объекта.

Обработчик события получает аргумент типа `EntityColumnBeforeEventArgs`.

Свойства `EntityColumnBeforeEventArgs` предоставляющие сведения, относящиеся к событию:

- `ColumnStreamValue`.
- `ColumnValue`.
- `ColumnValueName`.
- `DisplayColumnValue`.
- `DisplayColumnValueName`.

`event EventHandler<EventArgs> DefColumnValuesSet`

Обработчик события, возникающего после установки значений по умолчанию полей объекта.

`event EventHandler<EntityAfterEventArgs> Deleted`

Обработчик события, возникающего после удаления объекта.

Обработчик события получает аргумент типа `EntityAfterEventArgs`.

Свойства `EntityAfterEventArgs` предоставляющие сведения, относящиеся к событию:

- `ModifiedColumnValues`.
- `PrimaryColumnValue`.

`event EventHandler<EntityBeforeEventArgs> Deleting`

Обработчик события, возникающего перед удалением объекта.

Обработчик события получает аргумент типа `EntityBeforeEventArgs`.

Свойства `EntityBeforeEventArgs` предоставляющие сведения, относящиеся к событию:

- `AdditionalCondition`.
- `IsCanceled`.
- `KeyValue`.

`event EventHandler<EntityAfterEventArgs> Inserted`

Обработчик события, возникающего после вставки объекта.

Обработчик события получает аргумент типа `EntityAfterEventArgs`.

Свойства `EntityAfterEventArgs` предоставляющие сведения, относящиеся к событию:

- `ModifiedColumnValues`.
- `PrimaryColumnValue`.

`event EventHandler<EntityBeforeEventArgs> Inserting`

Обработчик события, возникающего перед вставкой объекта.

Обработчик события получает аргумент типа `EntityBeforeEventArgs`.

Свойства `EntityBeforeEventArgs` предоставляющие сведения, относящиеся к событию:

- `AdditionalCondition`.
- `IsCanceled`.
- `KeyValue`.

`event EventHandler<EntityAfterLoadEventArgs> Loaded`

Обработчик события, возникающего после загрузки объекта.

Обработчик события получает аргумент типа `EntityAfterLoadEventArgs`.

Свойства `EntityAfterLoadEventArgs` предоставляющие сведения, относящиеся к событию:

- `ColumnMap`.
- `DataSource`.

`event EventHandler<EntityBeforeLoadEventArgs> Loading`

Обработчик события, возникающего перед загрузкой объекта.

Обработчик события получает аргумент типа `EntityBeforeLoadEventArgs`.

Свойства `EntityBeforeLoadEventArgs` предоставляющие сведения, относящиеся к событию:

- `ColumnMap`.
- `DataSource`.
- `IsCanceled`.

`event EventHandler<EntityAfterEventArgs> Saved`

Обработчик события, возникающего после сохранения объекта.

Обработчик события получает аргумент типа `EntityAfterEventArgs`.

Свойства `EntityAfterEventArgs` предоставляющие сведения, относящиеся к событию:

- `ModifiedColumnValues`.
- `PrimaryColumnValue`.

`event EventHandler<EntitySaveErrorEventArgs> SaveError`

Обработчик события, возникающего при ошибке сохранения объекта.

Обработчик события получает аргумент типа `EntitySaveErrorEventArgs`.

Свойства `EntitySaveErrorEventArgs` предоставляющие сведения, относящиеся к событию:

- `Exception`.
- `IsHandled`.

`event EventHandler<EntityBeforeEventArgs> Saving`

Обработчик события, возникающего перед сохранением объекта.

Обработчик события получает аргумент типа `EntityBeforeEventArgs`.

Свойства `EntityBeforeEventArgs` предоставляющие сведения, относящиеся к событию:

- `AdditionalCondition`.
- `IsCanceled`.
- `KeyValue`.

`event EventHandler<EntityAfterEventArgs> Updated`

Обработчик события, возникающего после обновления объекта.

Обработчик события получает аргумент типа `EntityAfterEventArgs`.

Свойства `EntityAfterEventArgs` предоставляющие сведения, относящиеся к событию:

- `ModifiedColumnValues`.
- `PrimaryColumnValue`.

`event EventHandler<EntityBeforeEventArgs>Updating`

Обработчик события, возникающего перед обновлением объекта.

Обработчик события получает аргумент типа `EntityBeforeEventArgs`.

Свойства `EntityBeforeEventArgs` предоставляющие сведения, относящиеся к событию:

- `AdditionalCondition`.
- `IsCanceled`.
- `KeyValue`.

`event EventHandler<EntityValidationEventArgs> Validating`

Обработчик события, возникающего при проверке объекта.

Обработчик события получает аргумент типа `EntityValidationEventArgs`.

Свойства `EntityValidationEventArgs` предоставляющие сведения, относящиеся к событию:

- `Messages`.

Класс EntityMapper C#

Сложный

Класс `Terrasoft.Configuration.EntityMapper` — это утилитный класс конфигурации, который находится в пакете [*FinAppLending*] продукта Lending. `EntityMapper` позволяет сопоставлять данные одной сущности (`Entity`) с другой по правилам, определенным в конфигурационном файле. Использование подхода сопоставления данных разных сущностей позволяет избежать появления однообразного кода.

В продукте Lending существует два объекта, содержащих одинаковые колонки. Это объекты [*Физ. лицо*

] ([*Contact*]) и [*Анкета*] ([*AppForm*]). Также существует несколько деталей, относящихся к объекту [*Физ. лицо*] ([*Contact*]) и имеющих похожие детали, относящиеся к [*Анкета*] ([*AppForm*]). Очевидно, что при заполнении анкеты должна быть возможность по колонке [*Id*] объекта [*Физ. лицо*] ([*Contact*]) получить список всех его колонок и значений, а также список нужных деталей с их колонками и значениями, и сопоставить эти данные с данными, связанными с анкетой. После этого можно автоматически заполнить поля анкеты сопоставленными данными. Таким образом можно существенно уменьшить затраты на ручной ввод одинаковых данных.

Идея сопоставления данных разных сущностей реализована в следующих классах:

- `EntityMapper` — реализует логику сопоставления.
- `EntityResult` — определяет в каком виде вернется сопоставленная сущность.
- `MapConfig` — представляет набор правил для сопоставления.
- `DetailMapConfig` — используется для установки списка правил сопоставления деталей и связанных с ними сущностей.
- `RelationEntityMapConfig` — содержит правила для сопоставления связанных сущностей.
- `EntityFilterMap` — представляет из себя фильтр для запроса в базу данных.

Класс EntityMapper C#

Пространство имен `Terrasoft.Configuration`.

Класс реализует логику сопоставления.

Методы

```
virtual EntityResult GetMappedEntity(Guid recId, MapConfig config)
```

Возвращает сопоставленные данные для двух объектов `Entity`.

Параметры

`recId`	`GUID` записи в базе данных.
`config`	Экземпляр класса `MapConfig`, представляющий из себя набор правил сопоставления.

```
virtual Dictionary<string, object> GetColumnsValues(Guid recordId, MapConfig config, Dictionary<
```

Получает из базы данных главную сущность и сопоставляет ее колонки и значения по правилам, указанным в объекте `config`.

Параметры

`recordId`	`GUID` записи в базе данных.
`config`	Экземпляр класса `MapConfig`, представляющий из себя набор правил сопоставления.
`result`	Словарь колонок и их значений уже сопоставленной сущности.

```
virtual Dictionary<string, object> GetRelationEntityColumnsValues(List<RelationEntityMapConfig>
```

Получает из базы данных связанные сущности и сопоставляет их с основными сущностями.

Параметры

`relations`	Список правил для получения связанных записей.
`dictionaryToMerge`	Словарь с колонками и их значениями.
`columnName`	Название родительской колонки.
`entitylookup`	Объект, содержащий название и Id записи в базе.

Класс EntityResult C#

Пространство имен `Terrasoft.Configuration`.

Класс определяет в каком виде вернется сопоставленная сущность.

Свойства

`Columns` `Dictionary<string, object>`

Словарь с названиями колонок основной сущности и их значениями.

`Details` `Dictionary<string, List<Dictionary<string, object>>>`

Словарь названий деталей со списком их колонок и значений.

Класс MapConfig C#

Пространство имен `Terrasoft.Configuration`.

Класс представляет набор правил для сопоставления.

Свойства

`SourceEntityName` `string`

Название сущности в базе данных.

`Columns` `Dictionary<string, object>`

Словарь с названиями колонок одной сущности и сопоставляемыми колонками другой сущности.

`DetailsConfig` `List<DetailMapConfig>`

Список конфигурационных объектов с правилами для деталей.

`CleanDetails` `List<string>`

Список названий деталей для очистки их значений.

`RelationEntities` `List<RelationEntityMapConfig>`

Список конфигурационных объектов с правилами сопоставления связанных записей с главной сущностью.

Класс DetailMapConfig C#

Пространство имен `Terrasoft.Configuration`.

Класс используется для установки списка правил сопоставления деталей и связанных с ними сущностей.

Свойства

`DetailName` `string`

Название детали (для обеспечения уникальности экземпляра детали).

`SourceEntityName` `string`

Название сущности в базе данных.

`Columns` `Dictionary<string, object>`

Словарь с названиями колонок одной сущности и сопоставляемыми колонками другой сущности.

`Filters` `List<EntityFilterMap>`

Список конфигурационных объектов с правилами фильтрации для более точных выборок из базы данных.

`RelationEntities` `List<RelationEntityMapConfig>`

Список конфигурационных объектов с правилами сопоставления связанных записей с главной сущностью.

Класс RelationEntityMapConfig C#

Пространство имен `Terrasoft.Configuration`.

Класс содержит правила для сопоставления связанных сущностей.

Свойства

`ParentColumnName` `string`

Название родительской колонки, при нахождении которой будет срабатывать логика получения и сопоставления данных по сущности.

`SourceEntityName` `string`

Название сущности в базе данных.

`Columns` `Dictionary<string, object>`

Словарь с названиями колонок одной сущности и сопоставляемыми колонками другой сущности.

`Filters` `List<EntityFilterMap>`

Список конфигурационных объектов с правилами фильтрации для более точных выборок из базы данных.

`RelationEntities` `List<RelationEntityMapConfig>`

Список конфигурационных объектов с правилами сопоставления связанных записей с главной сущностью.

Класс EntityFilterMap C#

Пространство имен `Terrasoft.Configuration`.

Класс представляет из себя фильтр для запроса в базу данных.

Свойства

`ColumnName` `string`

Название колонки, при нахождении которой будет срабатывать логика фильтрации.

`Value` `object`

Значение, с которым необходимо сравнение.

Класс EntitySchemaQueryFunction C#

Сложный

Класс `Terrasoft.Core.Entities.EntitySchemaQueryFunction` реализует функцию выражения.

Идея функции выражения реализована в следующих классах:

- `EntitySchemaQueryFunction` — базовый класс функции выражения запроса к схеме объекта.
- `EntitySchemaAggregationQueryFunction` — реализует агрегирующую функцию выражения.
- `EntitySchemaIsNullQueryFunction` — заменяет значения `null` замещающим выражением.
- `EntitySchemaCoalesceQueryFunction` — возвращает первое выражение из списка аргументов, не равное `null`.
- `EntitySchemaCaseNotNullQueryFunctionWhenItem` — класс, описывающий выражение условия sql-оператора `CASE`.
- `EntitySchemaCaseNotNullQueryFunctionWhenItems` — коллекция выражений условий sql-оператора `CASE`.
- EntitySchemaStartOfCurrentHourQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression expression, int offset = 0) : this(parentQuery, offset)
- `EntitySchemaCaseNotNullQueryFunction` — возвращает одно из множества возможных значений в зависимости от указанных условий.
- `EntitySchemaSystemValueQueryFunction` — возвращает выражение системного значения.
- `EntitySchemaCurrentDateTimeQueryFunction` — реализует функцию выражения текущей даты и времени.
- `EntitySchemaBaseCurrentDateQueryFunction` — базовый класс функции выражения для базовой даты.
- `EntitySchemaCurrentDateQueryFunction` — реализует функцию выражения текущей даты.
- `EntitySchemaDateToCurrentYearQueryFunction` — реализует функцию выражения даты начала текущей недели.
- `EntitySchemaStartOfCurrentWeekQueryFunction` — реализует функцию, которая конвертирует выражение даты в такую же дату текущего года.
- `EntitySchemaStartOfCurrentMonthQueryFunction` — реализует функцию выражения даты начала текущего месяца.
- `EntitySchemaStartOfCurrentQuarterQueryFunction` — реализует функцию выражения даты начала

текущего квартала.

- `EntitySchemaStartOfCurrentHalfYearQueryFunction` — реализует функцию выражения даты начала текущего полугодия.
- `EntitySchemaStartOfCurrentYearQueryFunction` — реализует функцию выражения даты начала текущего года.
- `EntitySchemaBaseCurrentDateTimeQueryFunction` — базовый класс функции выражения базовых даты и времени.
- `EntitySchemaStartOfCurrentHourQueryFunction` — реализует функцию выражения начала текущего часа.
- `EntitySchemaCurrentTimeQueryFunction` — реализует функцию выражения текущего времени.
- `EntitySchemaCurrentUserQueryFunction` — реализует функцию выражения текущего пользователя.
- `EntitySchemaCurrentUserContactQueryFunction` — реализует функцию контакта текущего пользователя.
- `EntitySchemaCurrentUserAccountQueryFunction` — реализует функцию выражения контрагента текущего пользователя.
- `EntitySchemaDatePartQueryFunction` — реализует функцию запроса для части даты.
- `EntitySchemaUpperQueryFunction` — преобразовывает символы выражения аргумента к верхнему регистру.
- `EntitySchemaCastQueryFunction` — приводит выражение аргумента к заданному типу данных.
- `EntitySchemaTrimQueryFunction` — удаляет начальные и конечные пробелы из выражения.
- `EntitySchemaLengthQueryFunction` — возвращает длину выражения.
- `EntitySchemaConcatQueryFunction` — формирует строку, которая является результатом объединения строковых значений аргументов функции.
- `EntitySchemaWindowQueryFunction` — реализует функцию SQL окна.

Класс EntitySchemaQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Базовый класс функции выражения запроса к схеме объекта.

> **На заметку.** Полный перечень методов класса `EntitySchemaQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Методы

```
abstract QueryColumnExpression CreateQueryColumnExpression(DBSecurityEngine dbSecurityEngine)
```

Возвращает выражение колонки запроса для текущей функции, сформированное с учетом заданных прав доступа.

Параметры

dbSecurityEngine	Объект `Terrasoft.Core.DB.DBSecurityEngine`, определяющий права доступа.

`abstract DataValueType GetResultDataValueType(DataValueTypeManager dataValueTypeManager)`

Возвращает тип данных возвращаемого функцией результата, используя переданный менеджер типов данных.

Параметры

dataValueTypeManager	Менеджер типов данных.

`abstract bool GetIsSupportepataValueType(DataValueType dataValueType)`

Определяет, имеет ли возвращаемый функцией результат указанный тип данных.

Параметры

dataValueType	Тип данных.

`abstract string GetCaption()`

Возвращает заголовок функции выражения.

`virtual EntitySchemaQueryExpressionCollection GetArguments()`

Возвращает коллекцию выражений аргументов функции.

`void CheckIsSupportepataValueType(DataValueType dataValueType)`

Проверяет, имеет ли возвращаемый функцией результат указанный тип данных. В противном случае генерируется исключение.

Параметры

dataValueType	Тип данных.

Класс EntitySchemaAggregationQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует агрегирующую функцию выражения.

На заметку. Полный перечень методов и свойств класса `EntitySchemaAggregationQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaAggregationQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует экземпляр `EntitySchemaAggregationQueryFunction` заданного типа агрегирующей функции для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaAggregationQueryFunction(AggregationTypeStrict aggregationType, EntitySchemaQuery pa`

Инициализирует экземпляр `EntitySchemaAggregationQueryFunction` заданного типа агрегирующей функции для заданного запроса к схеме объекта.

Параметры

`aggregationType`	Тип агрегирующей функции.
`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaAggregationQueryFunction(AggregationTypeStrict aggregationType, EntitySchemaQueryExp`

Инициализирует новый экземпляр `EntitySchemaAggregationQueryFunction` для заданных типа агрегирующей функции, выражения и запроса к схеме объекта.

Параметры

`aggregationType`	Тип агрегирующей функции.
expression	Выражение запроса.
parentQuery	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaAggregationQueryFunction(EntitySchemaAggregationQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaAggregationQueryFunction`, являющийся клоном переданного экземпляра агрегирующей функции выражения.

Параметры

`source`	Экземпляр агрегирующей функции выражения, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`AggregationType` `AggregationTypeStrict`

Тип агрегирующей функции.

`AggregationEvalType` `AggregationEvalType`

Область применения агрегирующей функции.

`Expression` `EntitySchemaQueryExpression`

Выражение аргумента агрегирующей функции.

Методы

`override void WriteMetaData(DataWriter writer)`

Выполняет сериализацию агрегирующей функции, используя заданный экземпляр `Terrasoft.Common.DataWriter`.

Параметры

`writer`	Экземпляр `Terrasoft.Common.DataWriter`, с помощью которого выполняется сериализация.

`override QueryColumnExpression CreateQueryColumnExpression(DBSecurityEngine dbSecurityEngine)`

Возвращает выражение колонки запроса для агрегирующей функции, сформированное с учетом заданных прав доступа.

Параметры

dbSecurityEngine	Объект `Terrasoft.Core.DB.DBSecurityEngine`, определяющий права доступа.

`override EntitySchemaQueryExpressionCollection GetArguments()`

Возвращает коллекцию выражений аргументов агрегирующей функции.

`override DataValueType GetResultDataValueType(DataValueTypeManager dataValueTypeManager)`

Возвращает тип данных возвращаемого агрегирующей функцией результата, используя заданный менеджер типов данных.

Параметры

dataValueTypeManager	Менеджер типов данных.

`override bool GetIsSupportepataValueType(DataValueType dataValueType)`

Определяет, имеет ли возвращаемый агрегирующей функцией результат указанный тип данных.

Параметры

dataValueType	Тип данных.

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaAggregationQueryFunction`.

`EntitySchemaAggregationQueryFunction All()`

Устанавливает для текущей агрегирующей функции область применения [*Ко всем значениям*].

`EntitySchemaAggregationQueryFunction Distinct()`

Устанавливает для текущей агрегирующей функции область применения [*К уникальным значениям*].

Класс EntitySchemaIsNullQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс заменяет значения `null` замещающим выражением.

На заметку. Полный перечень методов и свойств класса `EntitySchemaIsNullQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaIsNullQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует экземпляр `EntitySchemaIsNullQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaIsNullQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression check`

Инициализирует новый экземпляр `EntitySchemaIsNullQueryFunction` для заданных запроса к схеме объекта, проверяемого выражения и замещающего выражения.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`checkExpression`	Выражение, которое проверяется на равенство `null`.
`replacementExpression`	Выражение, которое возвращается функцией, если `checkExpression` равно `null`.

`EntitySchemaIsNullQueryFunction(EntitySchemaIsNullQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaIsNullQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Экземпляр функции `EntitySchemaIsNullQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`CheckExpression` `EntitySchemaQueryExpression`

Выражение аргумента функции, которое проверяется на равенство значению `null`.

`ReplacementExpression` `EntitySchemaQueryExpression`

Выражение аргумента функции, которое возвращается, если проверяемое выражение равно `null`.

Методы

`override void WriteMetaData(DataWriter writer)`

Выполняет сериализацию функции выражения, используя переданный экземпляр `DataWriter`.

Параметры

`writer`	Экземпляр `DataWriter`, с помощью которого выполняется сериализация функции выражения.

`override QueryColumnExpression CreateQueryColumnExpression(DBSecurityEngine dbSecurityEngine)`

Возвращает выражение колонки запроса для текущей функции, сформированное с учетом заданных прав доступа.

Параметры

`dbSecurityEngine`	Объект `Terrasoft.Core.DB.DBSecurityEngine`, определяющий права доступа.

`override EntitySchemaQueryExpressionCollection GetArguments()`

Возвращает коллекцию выражений аргументов функции.

`override DataValueType GetResultDataValueType(DataValueTypeManager dataValueTypeManager)`

Возвращает тип данных возвращаемого функцией результата, используя переданный менеджер

типов данных.

Параметры

`dataValueTypeManager`	Менеджер типов данных.

Класс EntitySchemaCoalesceQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс возвращает первое выражение из списка аргументов, не равное `null`.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCoalesceQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCoalesceQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaCoalesceQueryFunction` для заданного запроса к схеме объекта.

Параметры

`aggregationType`	Тип агрегирующей функции.
`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaCoalesceQueryFunction(EntitySchemaCoalesceQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaCoalesceQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaCoalesceQueryFunction`, клон которой создается.

Свойства

`QueryAlias string`

Псевдоним функции в sql-запросе.

`Expressions` `EntitySchemaQueryExpressionCollection`

Коллекция выражений аргументов функции.

`HasExpressions` `bool`

Признак, определяющий наличие хотя бы одного элемента в коллекции выражений аргументов функции.

Методы

`override bool GetIsSupportepataValueType(DataValueType dataValueType)`

Определяет, имеет ли возвращаемый функцией результат указанный тип данных.

Параметры

`dataValueType`	Тип данных.

Класс EntitySchemaCaseNotNullQueryFunctionWhenItem C#

Пространство имен `Terrasoft.Core.Entities`.

Класс, описывающий выражение условия sql-оператора `CASE`.

На заметку. Полный перечень методов класса `EntitySchemaCaseNotNullQueryFunctionWhenItem`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCaseNotNullQueryFunctionWhenItem()`

Инициализирует новый экземпляр `EntitySchemaCaseNotNullQueryFunctionWhenItem`.

`EntitySchemaCaseNotNullQueryFunctionWhenItem(EntitySchemaQueryExpression whenExpression, EntityS`

Инициализирует экземпляр `EntitySchemaCaseNotNullQueryFunctionWhenItem` для заданных выражений предложений `WHEN` и `THEN`.

Параметры

whenExpression	Выражение предложения `WHEN` условия.
thenExpression	Выражение предложения `THEN` условия.

`EntitySchemaCaseNotNullQueryFunctionWhenItem(EntitySchemaCaseNotNullQueryFunctionWhenItem source`

Инициализирует экземпляр `EntitySchemaCaseNotNullQueryFunctionWhenItem`, являющийся клоном переданной функции.

Параметры

source	Функция `EntitySchemaCaseNotNullQueryFunctionWhenItem`, клон которой создается.

Свойства

`WhenExpression` `EntitySchemaQueryExpression`

Выражение предложения `WHEN`.

`ThenExpression` `EntitySchemaQueryExpression`

Выражение предложения `THEN`.

Класс EntitySchemaCaseNotNullQueryFunctionWhenItems C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует коллекцию выражений условий sql-оператора `CASE`.

На заметку. Полный перечень методов класса `EntitySchemaCaseNotNullQueryFunctionWhenItems`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCaseNotNullQueryFunctionWhenItems()`

Инициализирует экземпляр `EntitySchemaCaseNotNullQueryFunctionWhenItems`.

`EntitySchemaCaseNotNullQueryFunctionWhenItems(EntitySchemaCaseNotNullQueryFunctionWhenItems sour`

Инициализирует новый экземпляр `EntitySchemaCaseNotNullQueryFunctionWhenItems`, являющийся клоном клоном переданной коллекции условий.

Параметры

`source`	Коллекция условий, клон которой создается.

Класс EntitySchemaCaseNotNullQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс возвращает одно из множества возможных значений в зависимости от указанных условий.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCaseNotNullQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`CurrentDateTimeQueryFunction()`

Инициализирует новый экземпляр `CurrentDateTimeQueryFunction`.

`EntitySchemaCaseNotNullQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaCaseNotNullQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaCaseNotNullQueryFunction(EntitySchemaCaseNotNullQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaCaseNotNullQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaCaseNotNullQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`WhenItems` `EntitySchemaCaseNotNullQueryFunctionWhenItems`

Коллекция условий функции выражения.

`HasWhenItems` `bool`

Признак, имеет ли функция хотя бы одно условие.

`ElseExpression` `EntitySchemaQueryExpression`

Выражение предложения `ELSE`.

Методы

`void SpecifyQueryAlias(string queryAlias)`

Определяет для текущей функции выражения заданный псевдоним в результирующем sql-запросе.

Параметры

`queryAlias`	Псевдоним, определяемый для текущей функции.

Класс EntitySchemaSystemValueQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс возвращает выражение системного значения.

На заметку. Полный перечень методов и свойств класса `EntitySchemaSystemValueQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`SystemValueName` `string`

Имя системного значения.

Класс EntitySchemaCurrentDateTimeQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения текущей даты и времени.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCurrentDateTimeQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCurrentDateTimeQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует экземпляр `EntitySchemaCurrentDateTimeQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaCurrentDateTimeQueryFunction(EntitySchemaCurrentDateTimeQueryFunction source)`

Инициализирует экземпляр `EntitySchemaCurrentDateTimeQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Экземпляр функции `EntitySchemaCurrentDateTimeQueryFunction`, клон которой создается.

Свойства

`SystemValueName` `string`

Имя системного значения.

Методы

```
override string GetCaption()
```

Возвращает заголовок функции выражения.

```
override object Clone()
```

Создает клон текущего экземпляра `EntitySchemaCurrentDateTimeQueryFunction`.

Класс EntitySchemaBaseCurrentDateQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Базовый класс функции выражения для базовой даты.

На заметку. Полный перечень методов и свойств класса `EntitySchemaBaseCurrentDateQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Свойства

```
SystemValueName string
```

Имя системного значения.

```
Offset int
```

Смещение.

Класс EntitySchemaCurrentDateQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения текущей даты.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCurrentDateQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaCurrentDateQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) : this(parer
EntitySchemaCurrentDateQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression
```

Инициализирует экземпляр `EntitySchemaCurrentDateQueryFunction` с указанным смещением относительно базовой даты для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`offset`	Смещение в днях относительно контрольной даты. Значение по умолчанию - `0`.
`expression`	Выражение запроса.

`EntitySchemaCurrentDateQueryFunction(EntitySchemaCurrentDateQueryFunction source)`

Инициализирует экземпляр `EntitySchemaCurrentDateQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Экземпляр функции `EntitySchemaCurrentDateQueryFunction`, клон которой создается.

Методы

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaCurrentDateQueryFunction`.

Класс EntitySchemaDateToCurrentYearQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию, которая конвертирует выражение даты в такую же дату текущего года.

На заметку. Полный перечень методов и свойств класса `EntitySchemaDateToCurrentYearQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaDateToCurrentYearQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaDateToCurrentYearQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaDateToCurrentYearQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpre`

Инициализирует новый экземпляр `EntitySchemaDateToCurrentYearQueryFunction` для заданного запроса к схеме объекта и переданного выражения даты.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`expression`	Выражение запроса.

`EntitySchemaDateToCurrentYearQueryFunction(EntitySchemaDateToCurrentYearQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaDateToCurrentYearQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaDateToCurrentYearQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`Expression` `EntitySchemaQueryExpression`

Выражение аргументов функции.

Класс EntitySchemaStartOfCurrentWeekQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения текущей даты.

На заметку. Полный перечень методов и свойств класса `EntitySchemaStartOfCurrentWeekQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaStartOfCurrentWeekQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) : thi
EntitySchemaStartOfCurrentWeekQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpr
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentWeekQueryFunction` с указанным смещением относительно базовой даты для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`offset`	Смещение в днях относительно контрольной даты. Значение по умолчанию - `0`.
`expression`	Выражение запроса.

```
EntitySchemaStartOfCurrentWeekQueryFunction(EntitySchemaStartOfCurrentWeekQueryFunction source)
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentWeekQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Экземпляр функции `EntitySchemaStartOfCurrentWeekQueryFunction`, клон которой создается.

Методы

```
override string GetCaption()
```

Возвращает заголовок функции выражения.

```
override object Clone()
```

Создает клон текущего экземпляра `EntitySchemaStartOfCurrentWeekQueryFunction`.

Класс EntitySchemaStartOfCurrentMonthQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения даты начала текущего месяца.

На заметку. Полный перечень методов и свойств класса `EntitySchemaStartOfCurrentMonthQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaStartOfCurrentMonthQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) : th
EntitySchemaStartOfCurrentMonthQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExp
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentMonthQueryFunction` с указанным смещением относительно базовой даты для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`offset`	Смещение в днях относительно контрольной даты. Значение по умолчанию - `0`.
`expression`	Выражение запроса.

```
EntitySchemaStartOfCurrentMonthQueryFunction(EntitySchemaStartOfCurrentMonthQueryFunction source
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentMonthQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Экземпляр функции `EntitySchemaStartOfCurrentMonthQueryFunction`, клон которой создается.

Методы

```
override string GetCaption()
```

Возвращает заголовок функции выражения.

```
override object Clone()
```

Создает клон текущего экземпляра `EntitySchemaStartOfCurrentMonthQueryFunction`.

Класс EntitySchemaStartOfCurrentQuarterQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения даты начала текущего месяца.

На заметку. Полный перечень методов и свойств класса `EntitySchemaStartOfCurrentQuarterQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaStartOfCurrentQuarterQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) :
EntitySchemaStartOfCurrentQuarterQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryE
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentQuarterQueryFunction` с указанным смещением относительно базовой даты для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`offset`	Смещение в днях относительно контрольной даты. Значение по умолчанию - `0`.
`expression`	Выражение запроса

```
EntitySchemaStartOfCurrentQuarterQueryFunction(EntitySchemaStartOfCurrentQuarterQueryFunction sc
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentQuarterQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Экземпляр функции `EntitySchemaStartOfCurrentQuarterQueryFunction`, клон которой создается.

Методы

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaStartOfCurrentQuarterQueryFunction`.

Класс EntitySchemaStartOfCurrentHalfYearQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения даты начала текущего полугодия.

На заметку. Полный перечень методов и свойств класса `EntitySchemaStartOfCurrentHalfYearQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaStartOfCurrentHalfYearQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) :`
`EntitySchemaStartOfCurrentHalfYearQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQuery`

Инициализирует экземпляр `EntitySchemaStartOfCurrentHalfYearQueryFunction` с указанным смещением относительно базовой даты для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`offset`	Смещение в днях относительно контрольной даты. Значение по умолчанию - `0`.
`expression`	Выражение запроса.

`EntitySchemaStartOfCurrentHalfYearQueryFunction(EntitySchemaStartOfCurrentHalfYearQueryFunction`

Инициализирует экземпляр `EntitySchemaStartOfCurrentHalfYearQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Экземпляр функции `EntitySchemaStartOfCurrentHalfYearQueryFunction`, клон которой создается.

Методы

```
override string GetCaption()
```

Возвращает заголовок функции выражения.

```
override object Clone()
```

Создает клон текущего экземпляра `EntitySchemaStartOfCurrentHalfYearQueryFunction`.

Класс EntitySchemaStartOfCurrentYearQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения даты начала текущего года.

На заметку. Полный перечень методов и свойств класса `EntitySchemaStartOfCurrentYearQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaStartOfCurrentYearQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) : thi
EntitySchemaStartOfCurrentYearQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpr
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentYearQueryFunction` с указанным смещением относительно базовой даты для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`offset`	Смещение в днях относительно контрольной даты. Значение по умолчанию - `0`.
`expression`	Выражение запроса.

```
EntitySchemaStartOfCurrentYearQueryFunction(EntitySchemaStartOfCurrentYearQueryFunction source)
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentYearQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

`source`	Экземпляр функции `EntitySchemaStartOfCurrentYearQueryFunction`, клон которой создается.

Методы

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaStartOfCurrentHalfYearQueryFunction`.

Класс EntitySchemaBaseCurrentDateTimeQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Базовый класс функции выражения базовых даты и времени.

На заметку. Полный перечень методов и свойств класса `EntitySchemaBaseCurrentDateTimeQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Свойства

`SystemValueName string`

Имя системного значения.

Класс EntitySchemaStartOfCurrentHourQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения начала текущего часа.

На заметку. Полный перечень методов и свойств класса `EntitySchemaStartOfCurrentHourQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaStartOfCurrentHourQueryFunction(EntitySchemaQuery parentQuery, int offset = 0) : bas
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentHourQueryFunction`, который является частью `parentQuery` и указан `offset` относительно базовой даты.

Параметры

parentQuery	Экземпляр `EntitySchemaQuery`.
offset	Смещение в часах относительно базовой даты.

```
EntitySchemaStartOfCurrentHourQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpr
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentHourQueryFunction`, который является частью `parentQuery`, имеет указанные аргументы `expression` и `offset` относительно базовой даты.

Параметры

parentQuery	Экземпляр `EntitySchemaQuery`.
expression	Выражение аргумента функции.
offset	Смещение в часах относительно базовой даты.

```
EntitySchemaStartOfCurrentHourQueryFunction(EntitySchemaStartOfCurrentHourQueryFunction source)
```

Инициализирует экземпляр `EntitySchemaStartOfCurrentHourQueryFunction`, являющийся клоном переданной функции выражения.

Параметры

source	Экземпляр функции `EntitySchemaStartOfCurrentHourQueryFunction`, клон которой создается.

Методы

```
override string GetCaption()
```

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaStartOfCurrentHourQueryFunction`.

Класс EntitySchemaCurrentTimeQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения текущего времени.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCurrentTimeQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCurrentTimeQueryFunction(EntitySchemaQuery parentQuery) : base(parentQuery)`

Инициализирует новый экземпляр `EntitySchemaCurrentTimeQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaCurrentTimeQueryFunction(EntitySchemaCurrentTimeQueryFunction source) : base(source)`

Инициализирует новый экземпляр `EntitySchemaCurrentTimeQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaCurrentTimeQueryFunction`, клон которой создается.

Свойства

`SystemValueName string`

Имя системного значения.

Методы

```
override string GetCaption()
```

Возвращает заголовок функции выражения.

```
override object Clone()
```

Создает клон текущего экземпляра `EntitySchemaCurrentTimeQueryFunction`.

Класс EntitySchemaCurrentUserQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения текущего пользователя.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCurrentUserQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

```
EntitySchemaCurrentUserQueryFunction(EntitySchemaQuery parentQuery) : base(parentQuery)
```

Инициализирует новый экземпляр `EntitySchemaCurrentUserQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

```
EntitySchemaCurrentUserQueryFunction(EntitySchemaCurrentUserQueryFunction source) : base(source)
```

Инициализирует новый экземпляр `EntitySchemaCurrentUserQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaCurrentUserQueryFunction`, клон которой создается.

Свойства

```
SystemValueName string
```

~~SystemValueName string~~

Имя системного значения.

Методы

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaCurrentUserQueryFunction`.

Класс EntitySchemaCurrentUserContactQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения контакта текущего пользователя.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCurrentUserContactQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCurrentUserContactQueryFunction(EntitySchemaQuery parentQuery) : base(parentQuery)`

Инициализирует новый экземпляр `EntitySchemaCurrentUserContactQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaCurrentUserContactQueryFunction(EntitySchemaCurrentUserContactQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaCurrentUserContactQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaCurrentUserContactQueryFunction`, клон которой создается.

Свойства

`SystemValueName` `string`

Имя системного значения.

Методы

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaCurrentUserContactQueryFunction`.

Класс EntitySchemaCurrentUserAccountQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию выражения контрагента текущего пользователя.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCurrentUserAccountQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCurrentUserAccountQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaCurrentUserAccountQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaCurrentUserAccountQueryFunction(EntitySchemaCurrentUserAccountQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaCurrentUserAccountQueryFunction`, являющийся клоном переданной функции.

Параметры

source	Функция `EntitySchemaCurrentUserAccountQueryFunction`, клон которой создается.

Свойства

`SystemValueName` `string`

Имя системного значения.

Класс EntitySchemaDatePartQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию запроса для части даты.

На заметку. Полный перечень методов и свойств класса `EntitySchemaDatePartQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaDatePartQueryFunction(EntitySchemaQuery parentQuery) : base(parentQuery)`

Инициализирует новый экземпляр `EntitySchemaDatePartQueryFunction` для заданного запроса к схеме объекта.

Параметры

parentQuery	Экземпляр `EntitySchemaQuery`.

`EntitySchemaDatePartQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaDatePartQueryFuncti`

Инициализирует новый экземпляр `EntitySchemaDatePartQueryFunction`, который является частью `parentQuery` с указанной частью даты `interval` для запроса к схеме сущности и выражению запроса `expression`.

Параметры

`parentQuery`	Экземпляр `EntitySchemaQuery`.
`interval`	Часть даты.
`expression`	Выражение запроса.

`EntitySchemaDatePartQueryFunction(EntitySchemaDatePartQueryFunction source) : base(source)`

Инициализирует новый экземпляр `EntitySchemaDatePartQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaDatePartQueryFunction`, клон которой создается.

Свойства

`QueryAlias string`

Псевдоним функции в sql-запросе.

`EntitySchemaDatePartQueryFunctionInterval Interval`

Часть даты, возвращаемая функцией.

`EntitySchemaQueryExpression Expression`

Выражение аргумента функции.

Методы

`override void WriteMetaData(DataWriter writer)`

Выполняет сериализацию функции, используя заданный экземпляр `Terrasoft.Common.DataWriter`.

Параметры

`writer`	Экземпляр `Terrasoft.Common.DataWriter`, с помощью которого выполняется сериализация.

`override QueryColumnExpression CreateQueryColumnExpression(DBSecurityEngine dbSecurityEngine)`

Возвращает выражение колонки запроса для текущей функции, сформированное с учетом заданных прав доступа.

Параметры

`dbSecurityEngine`	Объект `Terrasoft.Core.DB.DBSecurityEngine`, определяющий права доступа.

`override DataValueType GetResultDataValueType(DataValueTypeManager dataValueTypeManager)`

Возвращает тип данных возвращаемого функцией результата, используя переданный менеджер типов данных.

Параметры

`dataValueTypeManager`	Менеджер типов данных.

`override bool GetIsSupportedDataValueType(DataValueType dataValueType)`

Определяет, имеет ли возвращаемый функцией результат указанный тип данных.

Параметры

`dataValueType`	Тип данных.

`override string GetCaption()`

Возвращает заголовок функции выражения.

`override EntitySchemaQueryExpressionCollection GetArguments()`

Возвращает коллекцию выражений аргументов функции.

`override object Clone()`

Создает клон текущего экземпляра `EntitySchemaUpperQueryFunction`.

Класс EntitySchemaUpperQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс преобразовывает символы выражения аргумента к верхнему регистру.

На заметку. Полный перечень методов и свойств класса `EntitySchemaUpperQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaUpperQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaUpperQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaUpperQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression expres`

Инициализирует новый экземпляр `EntitySchemaUpperQueryFunction` для заданного запроса к схеме объекта и переданного выражения даты.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`expression`	Выражение запроса.

`EntitySchemaUpperQueryFunction(EntitySchemaUpperQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaUpperQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaUpperQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`Expression` `EntitySchemaQueryExpression`

Выражение аргументов функции.

Класс EntitySchemaCastQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс приводит выражение аргумента к заданному типу данных.

На заметку. Полный перечень методов и свойств класса `EntitySchemaCastQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaCastQueryFunction(EntitySchemaQuery parentQuery, DBDataValueType castType)`

Инициализирует новый экземпляр `EntitySchemaCastQueryFunction` для заданного запроса к схеме объекта с указанным целевым типом данных.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`castType`	Целевой тип данных.

`EntitySchemaCastQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression express`

Инициализирует новый экземпляр `EntitySchemaCastQueryFunction` с заданными выражением и целевым типом данных.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`expression`	Выражение запроса.
`castType`	Целевой тип данных.

`EntitySchemaCastQueryFunction(EntitySchemaCastQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaCastQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaCastQueryFunction`, клон которой создается.

Свойства

`QueryAlias string`

Псевдоним функции в sql-запросе.

`Expression EntitySchemaQueryExpression`

Выражение аргумента функции.

`CastType DBDataValueType`

Целевой тип данных.

Класс EntitySchemaTrimQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс удаляет начальные и конечные пробелы из выражения.

На заметку. Полный перечень методов и свойств класса `EntitySchemaTrimQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaTrimQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaTrimQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaTrimQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression express`

Инициализирует новый экземпляр `EntitySchemaTrimQueryFunction` для заданного запроса к схеме объекта и переданного выражения даты.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`expression`	Выражение запроса.

`EntitySchemaTrimQueryFunction(EntitySchemaTrimQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaTrimQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaTrimQueryFunction`, клон которой создается.

Свойства

`QueryAlias string`

Псевдоним функции в sql-запросе.

`Expression EntitySchemaQueryExpression`

Выражение аргументов функции.

Класс EntitySchemaLengthQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс возвращает длину выражения.

На заметку. Полный перечень методов и свойств класса `EntitySchemaLengthQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaLengthQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaLengthQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaLengthQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression expre`

Инициализирует новый экземпляр `EntitySchemaLengthQueryFunction` для заданного запроса к схеме объекта и переданного выражения даты.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`expression`	Выражение запроса.

`EntitySchemaLengthQueryFunction(EntitySchemaLengthQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaLengthQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaLengthQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`Expression` `EntitySchemaQueryExpression`

Выражение аргументов функции.

Класс EntitySchemaConcatQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс формирует строку, которая является результатом объединения строковых значений аргументов функции.

На заметку. Полный перечень методов и свойств класса `EntitySchemaConcatQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaConcatQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaConcatQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaConcatQueryFunction(EntitySchemaQuery parentQuery, EntitySchemaQueryExpression[] exp`

Инициализирует новый экземпляр `EntitySchemaConcatQueryFunction` для заданных массива выражений и запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`expressions`	Массив выражений.

`EntitySchemaConcatQueryFunction(EntitySchemaConcatQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaConcatQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaConcatQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`Expressions` `EntitySchemaQueryExpressionCollection`

Коллекция выражений аргументов функции.

`HasExpressions` `bool`

Признак, определяющий наличие хотя бы одного элемента в коллекции выражений аргументов функции.

Класс EntitySchemaWindowQueryFunction C#

Пространство имен `Terrasoft.Core.Entities`.

Класс реализует функцию SQL окна.

На заметку. Полный перечень методов и свойств класса `EntitySchemaWindowQueryFunction`, его родительских классов, а также реализуемых им интерфейсов, можно найти в документации ".NET библиотеки классов ядра платформы".

Конструкторы

`EntitySchemaWindowQueryFunction(EntitySchemaQuery parentQuery)`

Инициализирует новый экземпляр `EntitySchemaWindowQueryFunction` для заданного запроса к схеме объекта.

Параметры

`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.

`EntitySchemaWindowQueryFunction(EntitySchemaQueryExpression function, EntitySchemaQuery esq)`

Инициализирует новый экземпляр `EntitySchemaWindowQueryFunction` для заданного запроса к схеме объекта.

Параметры

`function`	Вложенная функция запроса.
`esq`	Запрос к схеме объекта.

`EntitySchemaWindowQueryFunction(EntitySchemaQueryExpression function, EntitySchemaQuery esq, Ent`

Инициализирует новый экземпляр `EntitySchemaWindowQueryFunction` для заданного запроса к схеме объекта.

Параметры

`function`	Вложенная функция запроса.
`parentQuery`	Запрос к схеме объекта, которому принадлежит функция.
`partitionBy`	Выражение для разделения запроса.
`orderBy`	Выражение для сортировки запроса.

`EntitySchemaWindowQueryFunction(EntitySchemaQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaWindowQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaQueryFunction`, клон которой создается.

`EntitySchemaWindowQueryFunction(EntitySchemaWindowQueryFunction source)`

Инициализирует новый экземпляр `EntitySchemaWindowQueryFunction`, являющийся клоном переданной функции.

Параметры

`source`	Функция `EntitySchemaWindowQueryFunction`, клон которой создается.

Свойства

`QueryAlias` `string`

Псевдоним функции в sql-запросе.

`InnerFunction` `EntitySchemaQueryExpression`

Функция для применения.

`PartitionByExpression` `EntitySchemaQueryExpression`

Разделение по пунктам.

`OrderByExpression` `EntitySchemaQueryExpression`

Сортировать по пункту.

Класс EntitySchemaQueryOptions C#

Сложный

Пространство имен `Terrasoft.Core.Entities`.

Класс `Terrasoft.Core.Entities.EntitySchemaQueryOptions` предназначен для настроек запроса к схеме объекта.

На заметку. Полный перечень методов и свойств класса `EntitySchemaQueryOptions`, его родительских классов, а также реализуемых им интерфейсов можно найти в Библиотеке .NET классов.

Конструкторы

`EntitySchemaQueryOptions`

Инициализирует экземпляр класса. В конструкторе свойству `PageableRowCount` по умолчанию устанавливается значение 14.

Свойства

`PageableRowCount` `int`

Количество записей страницы результирующего набора данных, возвращаемого запросом.

`PageableDirection` `Terrasoft.Core.DB.PageableSelectDirection`

Направление постраничного вывода.

Возможные значения (`Terrasoft.Core.DB.PageableSelectDirection`)

`Prior`	Предыдущая страница.
`First`	Первая страница.
`Current`	Текущая страница.
`Next`	Следующая страница.

`PageableConditionValues` `Dictionary<string, object>`

Значения условий постраничного вывода.

`HierarchicalMaxDepth` `int`

Максимальный уровень вложенности иерархического запроса.

`HierarchicalColumnName` `string`

Имя колонки, которая используется для построения иерархического запроса.

`HierarchicalColumnValue` `Guid`

Начальное значение иерархической колонки, от которого будет строиться иерархия.

Сложные Select-запросы

Средний

Некоторые сложные запросы к базе данных могут нагружать ресурсы сервера базы данных на 100% в течение продолжительного времени. Это приводит к затруднению или невозможности работы других пользователей.

Виды сложных запросов:

- Неоптимально составленные запросы в динамических группах, блоках итогов.
- Сложные аналитические выборки в блоках итогов.

Способы выполнения сложных `Select`-запросов:

- С использованием отдельного пула запросов (доступно для Microsoft SQL Server Enterprise Edition).
- С использованием read-only реплики.

Использование вышеуказанных способов выполнения сложных `Select`-запросов позволяет:

- Ограничить ресурсы, которые выделяются сервером базы данных, на обработку сложных `Select`-запросов.
- Уменьшить влияние обработки сложных `Select`-запросов на работу других пользователей и частей системы.

Отдельный пул запросов

Назначение отдельного пула запросов — обработка сложных `Select`-запросов, которые не являются частью транзакции и вынесены в отдельный пул.

1. Настроить соединение отдельного пула запросов

MS SQL Server позволяет ограничивать выделяемые ресурсы с помощью встроенного инструмента — Resource Governor. Ранжирование соединений в Resource Governor базируется на информации о подключении, а не о конкретном запросе. Работу инструмента сложно увидеть на незагруженном сервере и на "коротких" запросах. Эффект от использования Resource Governor наблюдается, когда сервер баз данных загружен на 100%, а сложный запрос выполняется продолжительное время.

Чтобы **настроить соединение отдельного пула запросов**, откройте конфигурационный файл `ConnectionStrings.config` и для свойства `App` или `Application Name` допишите суффикс `_Limited`. Эта настройка позволяет использовать специальное соединение, которое разделит запросы на простые и потенциально сложные.

Виды соединений:

- Если в строке соединения конфигурационного файла `ConnectionStrings.config` для свойства `App` **не указано значение**, то для соединений будут использованы значения по умолчанию. Значение по умолчанию для общего соединения — "`.Net SqlClient DataProvider`", значение по умолчанию для соединения отдельного пула запросов — "`.Net SqlClient DataProvider_Limited`".
- Если в строке соединения конфигурационного файла `ConnectionStrings.config` для свойства `App` **указано значение** "`creatio`", то значение свойства для соединения отдельного пула запросов будет заменено значением "`creatio_Limited`".

Пример настройки свойства `App` строки соединения

```
<add name="db" connectionString="App=creatio; Data Source=dbserver\mssql2016; Initial Catalog
```

Таким образом, при загрузке дашбордов или фильтрации разделов с помощью динамических групп приложение создает дополнительные соединения с базой данных. В отличие от основных соединений, эти соединения в названиях содержат суффикс `_Limited`.

Важно. При использовании отдельного пула запросов не выполняется ограничение ресурсов. Использование суффикса `_Limited` позволяет выполнять ранжирование соединений средствами Resource Governor.

2. Включить функциональность отдельного пула запросов

Чтобы **включить функциональность отдельного пула запросов**, в файле `..\Terrasoft.WebApp\Web.config` установите значение `true` для ключа `UseQueryKinds` элемента `<appSettings>`. Ключ `UseQueryKinds` обеспечивает отправку запросов из дашбордов и динамических групп в соединения, которые в названии содержат суффикс `_Limited`.

`..\Terrasoft.WebApp\Web.config`

```
<add key="UseQueryKinds" value="true" />
```

3. Настроить инструмент Resource Governor

Настройка инструмента Resource Governor подразумевает настройку групп и пула.

Чтобы **настроить инструмент Resource Governor**, выполните SQL-скрипт.

Пример настройки групп и пула

```
ALTER RESOURCE POOL poolLimited WITH (
    MAX_CPU_PERCENT = 20,
    MIN_CPU_PERCENT = 0
    -- REQUEST_MAX_MEMORY_GRANT_PERCENT = value
    -- REQUEST_MAX_CPU_TIME_SEC = value
    -- REQUEST_MEMORY_GRANT_TIMEOUT_SEC = value
    -- MAX_DOP = value
    -- GROUP_MAX_REQUESTS = value
);
GO
--- Create a workload group for off-hours processing
--- and configure the relative importance.
    CREATE WORKLOAD GROUP groupLimited WITH (IMPORTANCE = LOW) USING poolLimited
GO ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```

Настройка инструмента подробно описана в официальной документации Resource Governor.

Для каждого нового соединения выполняется функция-классификатор, которая возвращает название группы.

Пример функции-классификатора

```
USE [master]
GO

ALTER FUNCTION [dbo].[fnProtoClassifier]()
```

```
    RETURNS sysname
    WITH SCHEMABINDING
AS
BEGIN
    IF(app_name() like '%_Limited')
    BEGIN
        RETURN N'groupLimited'
    END
    RETURN N'default'
END;
```

Read-only реплика

Начиная с версии 7.18.4, Creatio позволяет читать данные из read-only реплики. **Назначение** read-only реплики — обработка сложных `Select`-запросов.

Запросы, перенаправление которых позволяет настроить Creatio:

- Пользовательские `SelectQuery`-запросов из интерфейса приложения.
- `Select`-запросы с back-end части приложения, например, в элементе процесса [*Задание-сценарий*] ([*Script-task*]).

Как и для отдельного пула запросов, на read-only реплику позволяется направлять только `Select`-запросы, которые не являются частью транзакции. Creatio позволяет использовать только одну read-only реплику.

Настройка read-only реплики описана в статье Ускорить обработку сложных запросов к базе данных.

Выполнить сложный `Select`-запрос

Чтобы **выполнить** `Select` **-запрос**, получите специальный `DBExecutor`, передав в качестве дополнительного параметра значение `Limited` из перечисления `QueryKind`.

В приведенном ниже примере `QueryKind` — это аргумент метода `EnsureDBConnection()`. Значение аргумента приходит в клиентском `ESQ`-запросе, устанавливается в серверный `ESQ`-запрос и в `Select`-запрос.

Получение `DBExecutor` в зависимости от полученного `QueryKind`

```
using (DBExecutor executor = userConnection.EnsureDBConnection(QueryKind)) {
    /* ... */
};
```

Вызов `EnsureDBConnection(QueryKind.General)` эквивалентен вызову `EnsureDBConnection()` без указания `QueryKind`.

Таким образом, если при создании экземпляра класса `Terrasoft.EntitySchemaQuery` во front-end части приложения установить признак `QueryKind.Limited`, то это значение будет передано на сервер и запросу

будет обеспечен специальный `DBExecutor`, который использует отдельный пул запросов.

Пример установки признака `QueryKind.Limited` клиентскому `ESQ`-запросу в схеме `ChartModule`

```
...
getChartDataESQ: function() {
    return this.Ext.create("Terrasoft.EntitySchemaQuery", {
        rootSchema: this.entitySchema,
        queryKind: Terrasoft.QueryKind.LIMITED
    });
},
...
```

Важно. Если в программном коде присутствуют вложенные вызовы `userConnection.EnsureDBConnection(QueryKind)`, то необходимо убедиться, что на всех уровнях вложенности используется одно и то же значение `QueryKind`.

Копирование иерархических данных

Сложный

Для копирования записей таблицы базы данных и записей связанных таблиц используется **копирование иерархических данных**. В Creatio иерархическое копирование данных доступно к использованию для копирования таблиц `[ProductHierarchyDataStructureObtainer]` и `[ProductConditionHierarchyDataStructureObtainer]`. Чтобы скопировать данные с других таблиц, необходимо выполнить кастомизацию иерархического копирования данных.

Иерархическое копирование можно использовать в пользовательском веб-сервисе, например, при копировании данных из внешнего сервиса в приложение Creatio или при подключении к внешней базе данных.

Например, есть раздел [*Продукты*] ([*Products*]), который сформирован на основе таблицы `[Product]` базы данных. Страница продукта содержит пользовательские детали. При иерархическом копировании записи раздела будут скопированы и данные записи, и данные связанных деталей.

При копировании учитываются права доступа к таблицам. Например, пользователь не имеет прав на чтение и добавление записей в таблицу `[Contact]`. При вызове копирования иерархических данных запись не будет скопирована и приложение вернет сообщение о невозможности выполнения данной операции.

Структура и алгоритм работы копирования иерархических данных

Составляющие копирования иерархических данных представлены в таблице.

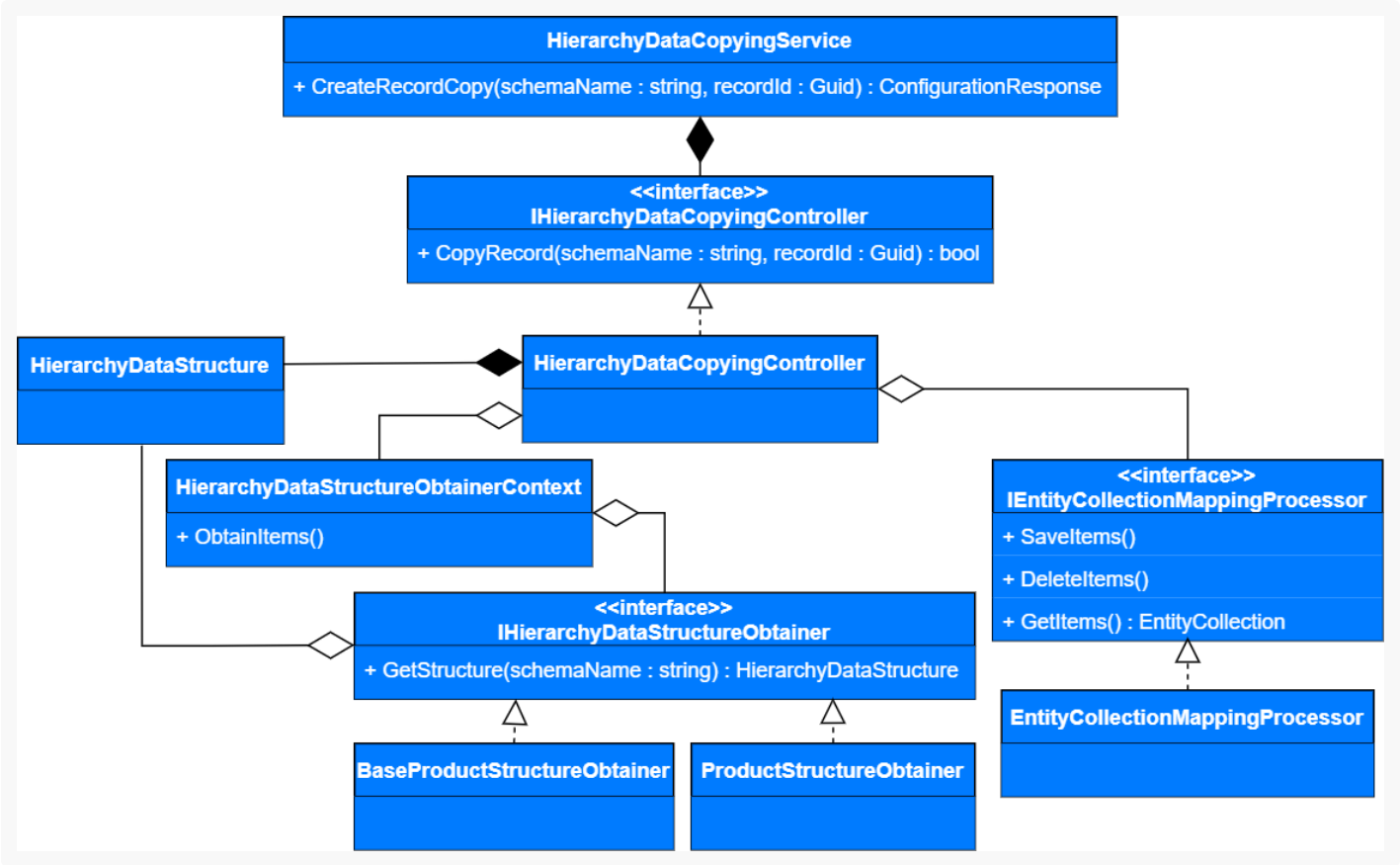
Составляющие копирования иерархических данных

Название	Описание	Классы и интерфейсы
Контроллер	Контролирует процесс копирования	`IHierarchyDataCopyingController` `HierarchyDataCopyingController` — контроль создания копии записи таблицы и связанных данных из базы данных.

Получатель	Получает из базы данных структуру текущей таблицы и связанных таблиц	`HierarchyDataStructureObtainerContext` — выбор алгоритма для получения иерархической структуры таблицы и связанных таблиц.
		`IHierarchyDataStructureObtainer`
		`HierarchyDataStructureObtainer` — получение иерархической структуры таблицы и связанных таблиц.
		`BaseHierarchyDataStructureObtainer`
		`ProductHierarchyDataStructureObtainer` — получение иерархической структуры таблицы `[Product]` и связанных таблиц.

Контейнер	Сохраняет структуру текущей таблицы и связанных таблиц	`HierarchyDataStructure` — контейнер для сохранения информации о структуре иерархических данных.
Мапер	Работает со структурой	`IEntityCollectionMappingProcessor` `EntityCollectionMappingProcessor` — получение структуры из базы данных, копирование.

Диаграмма классов копирования иерархических данных представлено на рисунке ниже.



Алгоритм работы копирования иерархических данных:

1. Класс сервиса вызывает контроллер процесса копирования и передает в него название таблицы и идентификатор записи, данные которой будут копироваться.
2. Контроллер начинает поэтапное создание копии:
   a. Получает структуру таблицы и связанных таблиц в унифицированной форме.
   b. Сохраняет полученную структуру таблицы в унифицированной форме.
3. Контроллер копирует записи в соответствии со структурой, полученной на этапе сохранения.

Унифицированная форма — это сохранение полученной структуры таблицы в объект с типом `HierarchyDataStructure`. Если таблица имеет связанные таблицы (колонка по внешнему ключу ссылается на запись другой таблицы), то они помещаются в созданный объект (в коллекцию из объектов аналогичного типа). При необходимости расширения или обновления механизма получения структуры использование унифицированной формы позволяет обработать новую структуру без дополнительных изменений кода в контроллере.

Шаблон унифицированной формы, которая используется при сохранении полученной структуры таблицы, приведен ниже.

Шаблон унифицированной формы структуры таблицы

```
/* Class holds structure of hierarchical data. */
public class HierarchyDataStructure
```

```
{
    public string SchemaName;
    public List<string> Columns;

    /* If current structure object does not have a parent foreign table name than here need to b
    public string ParentColumnName;

    /* List of child structures. */
    public List<HierarchyDataStructure> Structures;

    /* List filters. */
    public HierarchyDataStructureFilterGroup Filters;
}
```

Кастомизировать копирование иерархических данных

Кастомизация копирования иерархических данных позволяет:

- Добавить пользовательскую реализацию получателя данных (класс `HierarchyDataStructureObtainer`).
- Изменить реализацию получателя данных (класс `HierarchyDataStructureObtainer`).
- Изменить реализацию контроллера (класс `HierarchyDataCopyingController`).
- Добавить пользовательскую реализацию копирования иерархических данных.

Добавить пользовательскую реализацию получателя данных

Способы добавления пользовательской реализации получателя данных (класс `HierarchyDataStructureObtainer`):

- Через базовый интерфейс.
- Через наследование базового класса.

Добавить пользовательскую реализацию получателя данных через базовый интерфейс

1. Создайте класс, который реализует интерфейс `IHierarchyDataStructureObtainer`. Шаблон названия класса: `[НазваниеОбъектаКопирования]HierarchyDataStructureObtainer`.
2. Добавьте пользовательскую реализацию метода интерфейса `ObtainStructure()`. Обязательно укажите модификатор `virtual`.

Пример реализации получателя данных через базовый интерфейс содержится в пакете [*ProductBankCustomerJourney*] —> классы `ProductHierarchyDataStructureObtainer` и `ProductConditionHierarchyDataStructureObtainer`.

Добавить пользовательскую реализацию получателя данных через наследование

базового класса

Под базовой реализацией необходимо понимать стандартное копирование записи без связанных записей. Получатель реализован в классе `BaseHierarchyDataStructureObtainer` базового пакета [*NUI*].

1. Создайте класс, который реализует интерфейс `BaseHierarchyDataStructureObtainer` (пакет [*NUI*] —> класс `BaseHierarchyDataStructureObtainer`). Шаблон названия класса: `[НазваниеОбъектаКопирования]HierarchyDataStructureObtainer`.
2. Расширьте базовую реализацию получателя.

Пример реализации получателя данных через наследование базового класса содержится в пакете [*ProductBankCustomerJourney*] —> класс `ProductHierarchyDataStructureObtainer`).

Изменить реализацию получателя данных

1. Создайте класс, который замещает один из классов `BaseHierarchyDataStructureObtainer` (пакет [*NUI*]), `ProductConditionHierarchyDataStructureObtainer` (пакет [*ProductBankCustomerJourney*]), `ProductHierarchyDataStructureObtainer` (пакет [*ProductBankCustomerJourney*]).
2. В замещающий класс добавьте пользовательскую реализацию замещающего метода `ObtainStructure()` базового класса.

Изменить реализацию контроллера

1. Создайте класс, который реализует интерфейс `IHierarchyDataCopyingController`. Шаблон названия класса: `[НазваниеОбъекта]HierarchyDataController`.
2. В метод интерфейса `CopyRecord` добавьте пользовательский алгоритм копирования.

   Один шаг алгоритма должен содержать вызов одного метода другого класса. Шаг алгоритма также должен включать в себя создание объекта класса, который необходимо вызвать, или минимальную подготовку данных, которые будут передаваться в метод.

Добавить пользовательскую реализацию копирования иерархических данных

1. Добавьте реализацию контроллера процесса копирования (класс `HierarchyDataCopyingController`). Контроллер должен поэтапно вызывать получателя структуры (класс `HierarchyDataStructureObtainer`), обработчика структуры (класс `EntityCollectionMappingProcessor`), контейнера структуры (класс `HierarchyDataStructure`).
2. Добавьте реализацию получателя иерархической структуры данных (класс `HierarchyDataStructureObtainer`).
3. Создайте класс, который реализует интерфейс. Шаблон названия класса: `[НазваниеОбъекта]HierarchyDataProcessor`.

   Рекомендуется добавить интерфейс для класса обработчика. Это позволяет добавить другую реализацию и заменить существующую, а также используется для унификации всех обработчиков.
4. Создайте класс, реализующий интерфейс `IEntityCollectionMappingHandler`.

5. В метод контроллера `CopyRecord` добавьте вызовы методов получателя структуры (класс `HierarchyDataStructureObtainer`), обработчика структуры (класс `EntityCollectionMappingProcessor`), контейнера структуры (класс `HierarchyDataStructure`).
6. Создайте в пользовательском классе объект класса `HierarchyDataCopyingController`.

Пример создания объекта контроллера

```
var copyController = ClassFactory.Get<HierarchyDataCopyingController>(new ConstructorArgument
```

7. Вызовите метод копирования `copyController`.

Пример вызова метода копирования

```
copyController.CopyRecord(schemaName, recordId);
```

Вызвать копирование иерархических данных

Копирование иерархических данных можно вызвать из front-end и из back-end части.

Вызвать иерархическое копирование из front-end части

Чтобы **вызвать копирование иерархических данных из front-end части**, используйте метод `callService()`.

Пример вызова содержится в пакете [*ProductBankCustomerJourney*] —> схема `ProductConditionDetailV2` —> метод `callCopyRecordService()`.

Пример вызова сервиса копирования из front-end части

```
/**
* Call service that creates records copy.
* @protected
*/
callCopyRecordService: function() {
    this.showBodyMask();
    var config = this.getCopyRecordConfig();
    this.callService(config, this.copyRecordServiceCallback, this);
}
```

Вызвать иерархическое копирование из back-end части

Чтобы **вызвать копирование иерархических данных из back-end части**, в пользовательском классе создайте объект класса `HierarchyDataCopyingController`.

Пример вызова сервиса копирования из back-end части

```
var copyController = ClassFactory.Get<HierarchyDataCopyingController>(new ConstructorArgument("U
```

Чтобы **работать с данными таблиц по маппингу колонок**:

1. В пользовательском классе создайте объект класса маппера, который реализует интерфейс `IEntityCollectionMappingHandler`.

   Пример создания объекта класса маппера

   ```
   var entityCollectionMappingHandler = ClassFactory.Get<IEntityCollectionMappingHandler>(new Co
   ```

2. Вызовите методы маппера через объект.

   Пример вызова метода копирования

   ```
   entityCollectionMappingHandler.CopyItems(data.SchemaName, columns, filterGroup, relatedColumn
   ```

   На заметку. Создание объекта по названию интерфейса позволяет разработчику заменить существующую реализацию маппера на пользовательскую. При изменении реализации существующего маппера будет перекомпилирован только класс маппера. Все классы, которые используют данную реализацию, не требуют перекомпиляции. Подробнее о создании объектов с использованием механизма внедрения зависимостей описано в статье Замещение конфигурационных элементов.

Чтобы **получить структуру определенной таблицы** в виде объекта с типом `HierarchyDataStructure`:

1. В пользовательском классе создайте объект класса `HierarchyDataStructureObtainerContext`.

   Пример создания получателя структуры таблицы

   ```
   var _hierarchyDataStructureObtainer = ClassFactory.Get<HierarchyDataStructureObtainerContext>
   ```

2. Получите структуру определенной таблицы.

   Способы получения структуры:

   - Вызовите метод `ObtainStructureByObtainerStrategy` и передайте ему параметр `schemaName` — название таблицы, запись которой необходимо скопировать.
   - Вызовите реализацию существующего получателя структуры — `ProductHierarchyDataStructureObtainer` или `ProductConditionHierarchyDataStructureObtainer`.