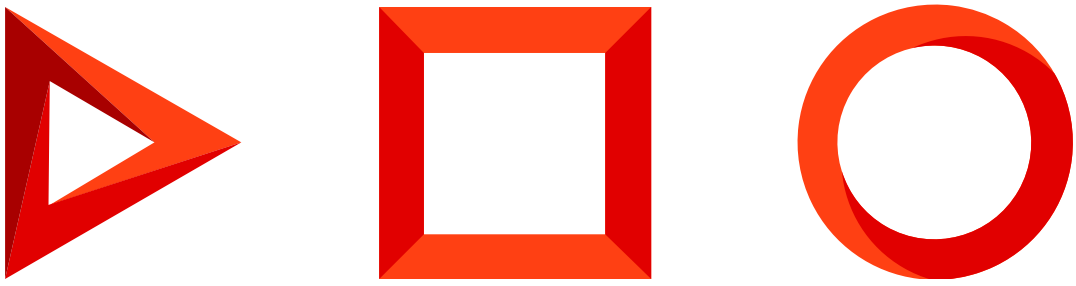
Creatio Academy

Управление элементами интерфейса

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

Механизм Feature Toggle

Средний

Feature toggle — техника разработки программных продуктов. **Назначение** Feature toggle — возможность подключения дополнительной функциональности в работающем приложении.

В Creatio Feature toggle использует непрерывную интеграцию, которая позволяет сохранить работоспособность приложения и скрыть функциональность на стадии разработки. В исходном коде приложения присутствует блок с дополнительной функциональностью и условный оператор, который определяет состояние подключения функциональности.

Понятие Feature toggle описано на Википедии.

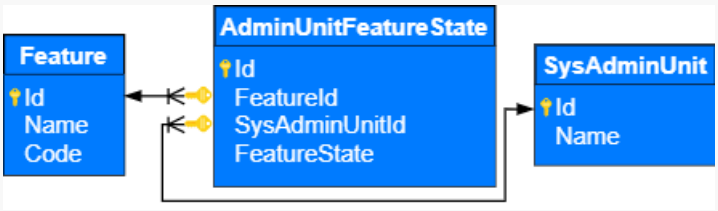
Хранить сведения о функциональности

Сведения о функциональности хранятся в базе данных приложения.

Таблицы базы данных, которые хранят информацию о функциональности:

- `[Feature]` — перечень функциональности, которая доступна для подключения/отключения. По умолчанию таблица пуста.
- `[AdminUnitFeatureState]` (колонка `[FeatureState]`) — информация о состоянии функциональности (подключена/отключена). Таблица `[AdminUnitFeatureState]` связывает таблицы `[Feature]` и `[SysAdminUnit]` базы данных. Таблица `[SysAdminUnit]` содержит информацию о пользователях и группах пользователей приложения.

Диаграмма взаимосвязи таблиц базы данных, которые хранят информацию о функциональности, представлена на рисунке ниже.



Основные колонки таблицы `[Feature]` базы данных приведены в таблице ниже.

Основные колонки таблицы `[Feature]`

Колонка	Тип	Описание
`[Id]`	`uniqueidentifier`	Уникальный идентификатор записи.
`[Name]`	`nvarchar(250)`	Имя функциональности.
`[Code]`	`nvarchar( 50 )`	Код функциональности.

Основные колонки таблицы `[AdminUnitFeatureState]` базы данных приведены в таблице ниже.

Основные колонки таблицы `[AdminUnitFeatureState]`

Колонка	Тип	Описание
`[Id]`	`uniqueidentifier`	Уникальный идентификатор записи.
`[FeatureId]`	`uniqueidentifier`	Уникальный идентификатор записи из таблицы `[Feature]`.
`[SysAdminUnitId]`	`uniqueidentifier`	Уникальный идентификатор записи пользователя.
`[FeatureState]`	`int`	Состояние функциональности (`1` — подключена, `0` — отключена).

Подключить дополнительную функциональность

Способы подключения дополнительной функциональности:

- Для текущего пользователя приложения.
- Для всех пользователей приложения.

Подключить функциональность для текущего пользователя приложения

1. Перейдите на страницу [*Функциональность*] ([*Features*]), которая используется для точечного подключения функциональности.

   Шаблон адреса страницы функциональности

   ```
   [Адрес приложения]/0/Nui/ViewModule.aspx#BaseSchemaModuleV2/FeaturesPage
   ```

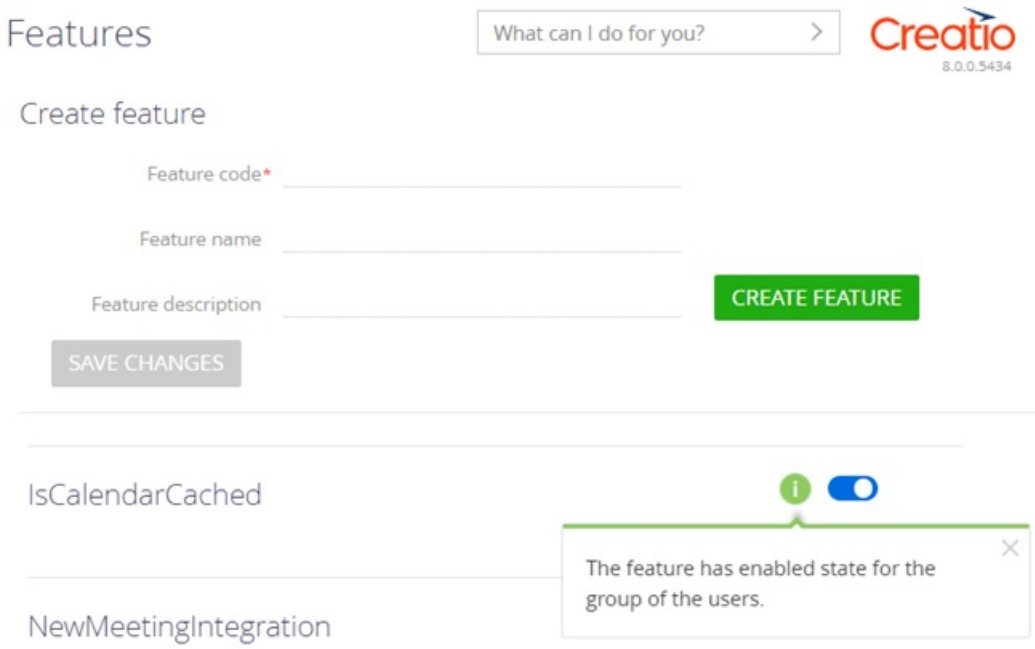
   Пример адреса страницы функциональности

```
http://mycreatio.com/0/Nui/ViewModule.aspx#BaseSchemaModuleV2/FeaturesPage
```

2. Переключите соответствующий переключатель функциональности, которую планируется подключить.
3. Нажмите кнопку [*Сохранить изменения*] ([*Save changes*]).

Всплывающая подсказка отображает статус функциональности для группы пользователей, в которую входит текущий пользователь. Подключение функциональности для текущего пользователя не меняет ее статус для группы пользователей. Т. е. для текущего пользователя создается дополнительное правило приоритет которого выше, чем приоритет правила для группы пользователей, в которую входит текущий пользователь.

Пример страницы [*Функциональность*] ([*Features*]) приведен на рисунке ниже.



Подключить функциональность для всех пользователей приложения

Чтобы **подключить функциональность для всех пользователей приложения**, выполните SQL-запрос к базе данных, который приведен ниже.

Пример SQL-запроса

MSSQL

```
DECLARE @featureCode varchar(max) = 'NewEmailSettingsWithoutAutoSynchronization', @featureId uni
set @featureId = (select top 1 Id from Feature where Code = @featureCode);
IF @featureId is null
    BEGIN
```

```
        insert into Feature
            (Name, Code)
        values
            (@featureCode, @featureCode);
        set @featureId = (select top 1 Id from Feature where Code = @featureCode);
    END;
delete from AdminUnitFeatureState where FeatureId = @featureId;
insert into AdminUnitFeatureState
    (SysAdminUnitId, FeatureState, FeatureId)
    values
        ('A29A3BA5-4B0D-DE11-9A51-005056C00008', 1, @featureId),
        ('720B771C-E7A7-4F31-9CFB-52CD21C3739F', 1, @featureId);
```

PostgreSQL

```
do $$
    DECLARE
        featureCode varchar(100) = 'EmailIntegrationV2';
        featureId UUID;
    BEGIN
        featureId = (select "Id" from "Feature" where "Code" = featureCode limit 1);
        IF featureId is null THEN
            insert into "Feature"
                ("Name", "Code")
            values
                (featureCode, featureCode);
            featureId = (select "Id" from "Feature" where "Code" = featureCode limit 1);
        end IF;
    delete from "AdminUnitFeatureState" where "FeatureId" = featureId;
    insert into "AdminUnitFeatureState"
        ("SysAdminUnitId", "FeatureState", "FeatureId")
        values
            ('A29A3BA5-4B0D-DE11-9A51-005056C00008', 1, featureId),
            ('720B771C-E7A7-4F31-9CFB-52CD21C3739F', 1, featureId);
end $$;
```

Oracle

```
DECLARE
    existObject NUMBER := 0;
    v_featurecode VARCHAR2(50) := 'LoadAttachmentsInSameProcess';
    v_featureid VARCHAR2(38);
BEGIN
    SELECT COUNT(1) INTO existObject FROM "Feature" WHERE "Code" = v_featurecode;
    if existObject = 0 THEN
```

```
        INSERT INTO "Feature" ("Name", "Code") VALUES (:v_featurecode, :v_featurecode);
    END IF;
SELECT "Id" INTO v_featureid FROM "Feature" WHERE "Code" = v_featurecode AND rownum <= 1;
DELETE FROM "AdminUnitFeatureState" WHERE "FeatureId" = v_featureid;
INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId")
    VALUES ('{A29A3BA5-4B0D-DE11-9A51-005056C00008}', 1, v_featureid);
END;
/
```

`featureCode` — переменная, которая содержит необходимое имя функциональности.

С помощью SQL-запроса Creatio позволяет подключить функциональность для группы пользователей и отключить функциональность для текущего пользователя и наоборот.

Чтобы **отключить функциональность для всех пользователей приложения**, выполните SQL-запрос к базе данных, в котором для колонки `[FeatureState]` таблицы `[AdminUnitFeatureState]` установите значение `0`.

Важно. Creatio предоставляет возможность отключения функциональности, которая была подключена через SQL-запрос к базе данных, только путем выполнения соответствующего SQL-запроса к базе данных. Отсутствует возможность отключения функциональности на странице [*Функциональность*] ([*Features*]).

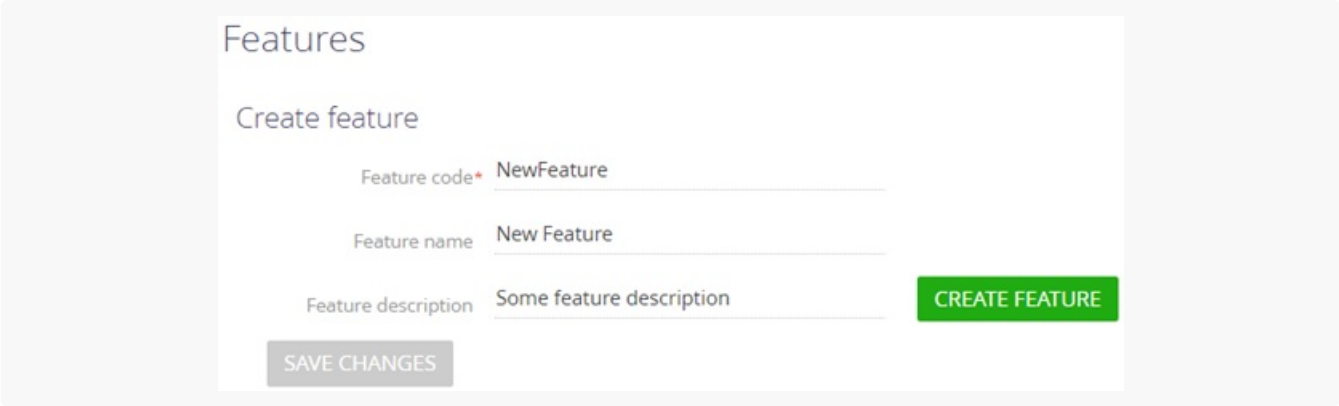
Реализовать пользовательскую функциональность

1. Добавьте **пользовательскую функциональность**.

   a. Перейдите на страницу [*Функциональность*] ([*Features*]), которая используется для точечного подключения функциональности.

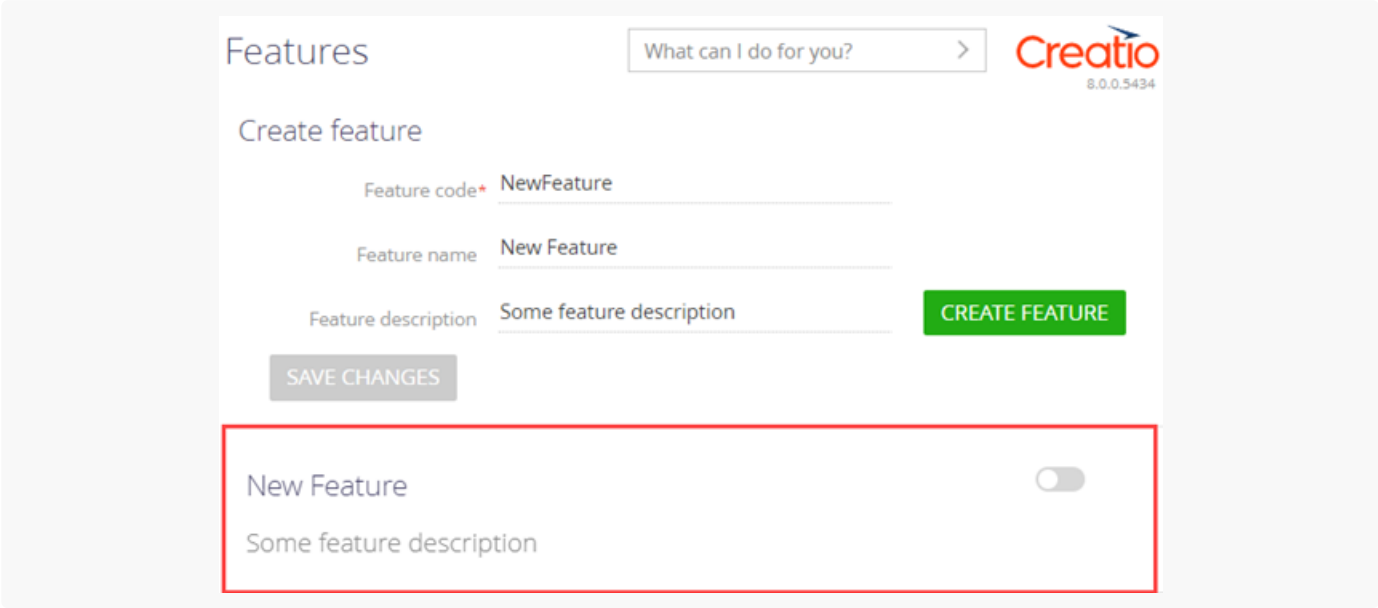
   b. Заполните **свойства добавляемой функциональности**:

   - [*Код функциональности*] ([*Feature code*]) — код пользовательской функциональности, которую планируется добавить (обязательное свойство).
   - [*Имя функциональности*] ([*Feature name*]) — имя пользовательской функциональности, которую планируется добавить.
   - [*Описание функциональности*] ([*Feature description*]) — описание пользовательской функциональности, которую планируется добавить.



f. Нажмите кнопку [*Добавить функциональность*] ([*Create feature*]).

В результате пользовательская функциональность добавлена в реестр страницы [*Функциональность*] ([*Features*]).



2. Реализуйте **пользовательскую функциональность в исходном коде**. Пользовательская функциональность определяется в блоке условного оператора, который проверяет состояние подключения функциональности (колонка `[FeatureState]` таблицы `[AdminUnitFeatureState]`).

   Варианты реализации пользовательской функциональности:

   - На front-end стороне. Для этого воспользуйтесь инструкцией, которая приведена в пункте Реализовать пользовательскую функциональность (front-end).
   - На back-end стороне. Для этого воспользуйтесь инструкцией, которая приведена в пункте Реализовать пользовательскую функциональность (back-end).

Реализовать пользовательскую функциональность (front-end)

1. Создайте схему модели представления. Для этого воспользуйтесь инструкцией, которая приведена в статье Клиентский модуль.
2. В дизайнере модуля добавьте исходный код. Для этого используйте шаблон, который приведен

ниже.

Шаблон реализации пользовательской функциональности

```
/* Метод, в котором определяется дополнительная функциональность. */
someMethod: function() {
    /* Проверка подключения функциональности. */
    if (Terrasoft.Features.getIsEnabled("КодФункциональности")) {
        /* Реализация дополнительной функциональности. */
        ...
    }
    /* Реализация метода. */
    ...
}
```

Для удобства использования в базовой схеме `BaseSchemaViewModel` модели представления определен метод `getIsFeatureEnabled`. Поэтому можно заменить метод `Terrasoft.Features.getIsEnabled` на `this.getIsFeatureEnabled("КодФункциональности")`.

3. На панели инструментов дизайнера модуля нажмите [*Сохранить*] ([*Save*]).

Чтобы пользовательская функциональность отобразилась в клиентском коде и загружалась браузером на front-end стороне, обновите страницу.

Реализовать пользовательскую функциональность (back-end)

1. Создайте схему исходного кода. Для этого воспользуйтесь инструкцией, которая приведена в статье Исходный код.
2. В дизайнере исходного кода добавьте исходный код.

   Класс `Terrasoft.Configuration.FeatureUtilities` предоставляет набор расширяющих методов класса `UserConnection`, которые позволяют использовать функциональность `Feature toggle` в схемах исходного кода на back-end стороне приложения. Для этого используйте шаблон, который приведен ниже.

   Шаблон реализации пользовательской функциональности

```
...
/* Пространство имен, в котором определена возможность переключения дополнительной функционал
using Terrasoft.Configuration;
...
/* Метод, в котором будет определяться дополнительная функциональность. */
public void AnyMethod() {
    /* Проверка, подключена ли функциональность. */
    if (UserConnection.GetIsFeatureEnabled("КодФункциональности")) {
        /* Реализация дополнительной функциональности. */
    }
```

```
    /* Реализация метода. */
    ...
}
```

3. Установите состояние функциональности, вызвав метод `SetFeatureState`. Для этого используйте шаблон, который приведен ниже.

Шаблон вызова метода `SetFeatureState`

```
UserConnection.SetFeatureState("КодФункциональности", FeatureState);
```

Класс FeatureUtilities C#

Средний

Пространство имен `Terrasoft.Configuration`.

Класс `Terrasoft.Configuration.FeatureUtilities` предоставляет набор расширяющих методов класса `UserConnection`, которые позволяют использовать функциональность `Feature Toggle` в схемах исходного кода на back-end стороне приложения. Также в классе `FeatureUtilities` объявлено перечисление состояний функциональностей (колонка `[FeatureState]` таблицы `[AdminUnitFeatureState]`).

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

Методы

`static int GetFeatureState(this UserConnection source, string code)`

Возвращает состояние функциональности.

Параметры

`source`	Пользовательское подключение.
`code`	Код функциональности.

`static int GetFeatureState(this UserConnection source, string code, Guid sysAdminUnitId)`

Возвращает состояние функциональности.

Параметры

source	Пользовательское подключение.
code	Код функциональности.
sysAdminUnitId	Уникальный идентификатор записи.

`static bool GetIsFeatureEnabledForAnyUser(this UserConnection source, string code)`

Признак, который возвращает состояние функциональности для любого пользователя.

Параметры

source	Пользовательское подключение.
code	Код функциональности.

`static bool GetIsFeatureEnabledForAllUsers(this UserConnection source, string code)`

Признак, который возвращает состояние функциональности для всех пользователей.

Параметры

source	Пользовательское подключение.
code	Код функциональности.

`static Dictionary<string, int> GetFeatureStates(this UserConnection source)`

Возвращает состояния всех функциональностей.

Параметры

source	Пользовательское подключение.

`static List<FeatureInfo> GetFeaturesInfo(this UserConnection source)`

Возвращает информацию относительно всех функциональностей и их состояний.

Параметры

source	Пользовательское подключение.

```
static void SetFeatureState(this UserConnection source, string code, int state, bool forAllUsers
```

Устанавливает состояние функциональности.

Параметры

source	Пользовательское подключение.
code	Код функциональности.
state	Новое состояние функциональности.
forAllUsers	Признак, который устанавливает функциональность для всех пользователей.

```
static void SafeSetFeatureState(this UserConnection source, string code, int state, bool forAllU
```

Установить состояние функциональности или создать функциональность (если она не существует).

Параметры

source	Пользовательское подключение.
code	Код функциональности.
state	Новое состояние функциональности.
forAllUsers	Признак, который устанавливает функциональность для всех пользователей.

```
static void CreateFeature(this UserConnection source, string code, string name, string descripti
```

Создает функциональность.

Параметры

`source`	Пользовательское подключение.
`code`	Код функциональности.
`name`	Имя функциональности.
`description`	Описание функциональности.

```
static bool GetIsFeatureEnabled(this UserConnection source, string code)
```

Признак, который проверяет подключение функциональности.

Параметры

`source`	Пользовательское подключение.
`code`	Код функциональности.

```
static bool GetIsFeatureEnabled(this UserConnection source, string code, Guid sysAdminUnitId)
```

Признак, который проверяет подключение функциональности.

Параметры

`source`	Пользовательское подключение.
`code`	Код функциональности.
`sysAdminUnitId`	Уникальный идентификатор записи.

Перечисление `FeatureState`

Перечисление `FeatureState` определяет состояния функциональностей (колонка `[FeatureState]` таблицы `[AdminUnitFeatureState]`).

`Disabled`	`0`	Функциональность отключена.
`Enabled`	`1`	Функциональность подключена.
`Established`	`2`	Функциональность установлена.

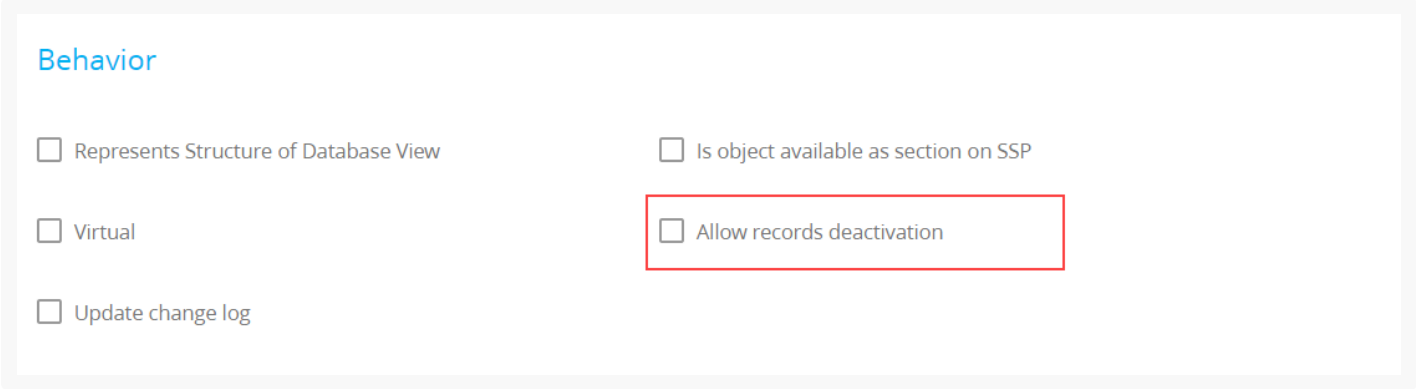
Деактивация записей объектов

Сложный

Creatio предоставляет возможность деактивировать записи объектов системы для исключения их из бизнес-логики. Это может понадобиться, например, если данные устарели и больше не используются.

Использование в дизайнере объекта

Функциональность включается специальным свойством [*Разрешить деактивацию записей*] ([*Allow record deactivation*]) в дизайнере объектов и становится доступной после публикации объекта.



Функциональность деактивации записей объектов доступна для всех объектов, но автоматическая фильтрация записей работает только в выпадающих списках, на странице выбора из справочника и в быстрых фильтрах. На страницах с содержимым справочников, в расширенных фильтрах и разделах автоматический фильтр не применяется.

Использование в программном коде

За включение или отключение фильтрации по неактивным записям отвечает параметр `UseRecordDeactivation` класса `EntitySchemaQuery`. По умолчанию этот параметр имеет значение `false`. Если ему присвоить значение `true`, то в запрос на выборку данных из объекта, где включена деактивация записей, будет добавлен фильтр, исключающий неактивные записи.

Пример использования в клиентском коде

```
var esq = Ext.create("Terrasoft.EntitySchemaQuery", {
   rootSchemaName: "MyCustomLookup",
   useRecordDeactivation: true
});
```

Пример использования в серверном коде

```
var esq = new EntitySchemaQuery(userConnection.EntitySchemaManager, "ContactType") {
    UseRecordDeactivation = true
};
```

```
esq.PrimaryQueryColumn.IsAlwaysSelect = true;
```

Результирующий SQL-запрос в данном случае будет иметь следующий вид:

SQL-запрос

```
SELECT
  [ContactType].[Id] [Id]
FROM [dbo].[ContactType] [ContactType] WITH(NOLOCK)
WHERE
  [ContactType].[RecordInactive] = 0
```