

# Страница записи

Страница записи

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

<b>Страница записи</b>	<b>4</b>
Контейнеры страницы записи	4
Создать страницу записи	5
Настроить страницу записи	6
Добавить пользовательское действие на страницу записи	7
<b>Добавить действие на страницу записи</b>	<b>8</b>
1. Создать схему замещающей модели представления страницы заказа	8
2. Создать схему замещающей модели представления раздела	12
Результат выполнения примера	14
<b>Схема BasePageV2</b>	<b>15</b>
Сообщения	15
Атрибуты	17
Методы	19
<b>Схема BaseEntityPage</b>	<b>22</b>
Сообщения	22
Атрибуты	22
Миксины	23
Методы	23

# Страница записи



**Страница записи** — элемент интерфейса, который хранит информацию о бизнес-объектах приложения в виде полей, вкладок, деталей и дашбордов. Имя страницы записи соответствует имени объекта системы (например, страница контрагента, страница контакта и т. д.). **Назначение** страницы записи — работа с записями [реестра раздела](#). Каждый раздел приложения содержит одну или несколько страниц записей.

Каждая страница записи представлена схемой [клиентского модуля](#). Например, страница контакта сконфигурирована в схеме `ContactPageV2` пакета `UIv2`. Функциональность базовой страницы записи реализована в схеме `BasePageV2` пакета `NUI`. Все схемы страниц записи должны наследовать схему `BasePageV2`.

**Виды** страницы записи:

- **Страница добавления записи** — используется для создания записи реестра раздела.
- **Страница редактирования записи** — используется для редактирования существующей записи реестра раздела.

## Контейнеры страницы записи

Элементы пользовательского интерфейса приложения, которые относятся к странице записи, размещены в соответствующих контейнерах. Контейнеры конфигурируются в базовой схеме страницы записи или схеме замещающей страницы записи. Не зависят от вида страницы записи.

**На заметку.** В приложении используются мета-имена html-контейнеров. На основании мета-имен приложение формирует фактические идентификаторы соответствующих html-элементов страницы записи.

Основные **контейнеры** страницы записи представлены на рисунке ниже.

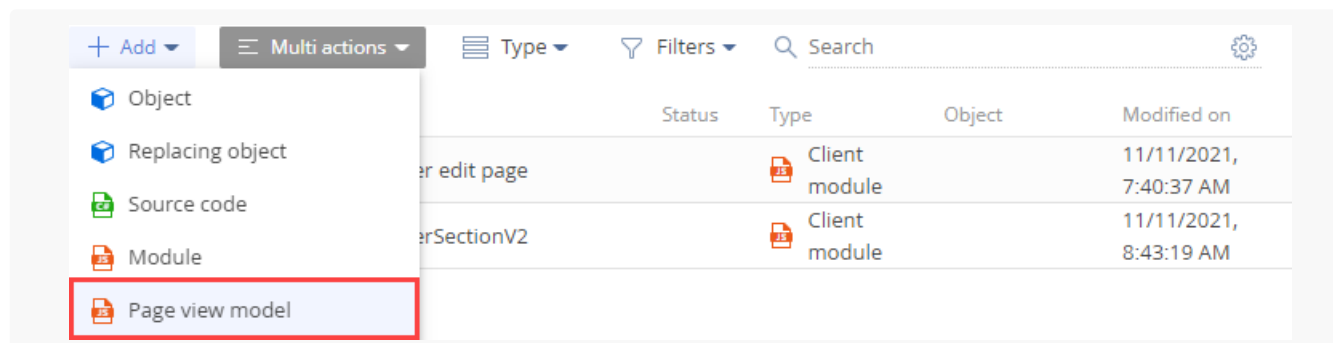
The screenshot displays a record page for 'Vertigo Systems' with the following components:

- ActionButtonsContainer:** Located at the top right, containing a 'CLOSE' button, an 'ACTIONS' dropdown, and a 'VIEW' dropdown.
- LeftModulesContainer:** Located on the left side, showing a progress bar at 95%, an 'Enrich data' button, and a form with fields for Name (Vertigo Systems), Type (Customer), Owner (Mary King), Web (www.vertigosys.com), and Primary phone (+44 (20) 3427 1374).
- ActionDashboardContainer:** Located at the top right, below the buttons, showing 'NEXT STEPS (0)' and a message: 'You don't have any tasks yet. Press [icon] above to add a task'.
- TabsContainer:** Located on the right side, containing tabs for ACCOUNT INFO, CONTACTS AND STRUCTURE, MAINTENANCE, TIMELINE, and CONNECTED TO. The 'ACCOUNT INFO' tab is active, showing fields for 'Also known as', 'Code' (109), 'Segmentation', 'No. of employees' (101-200), 'Business entity' (Corp.), and 'Annual revenue' (21 - 30 million).

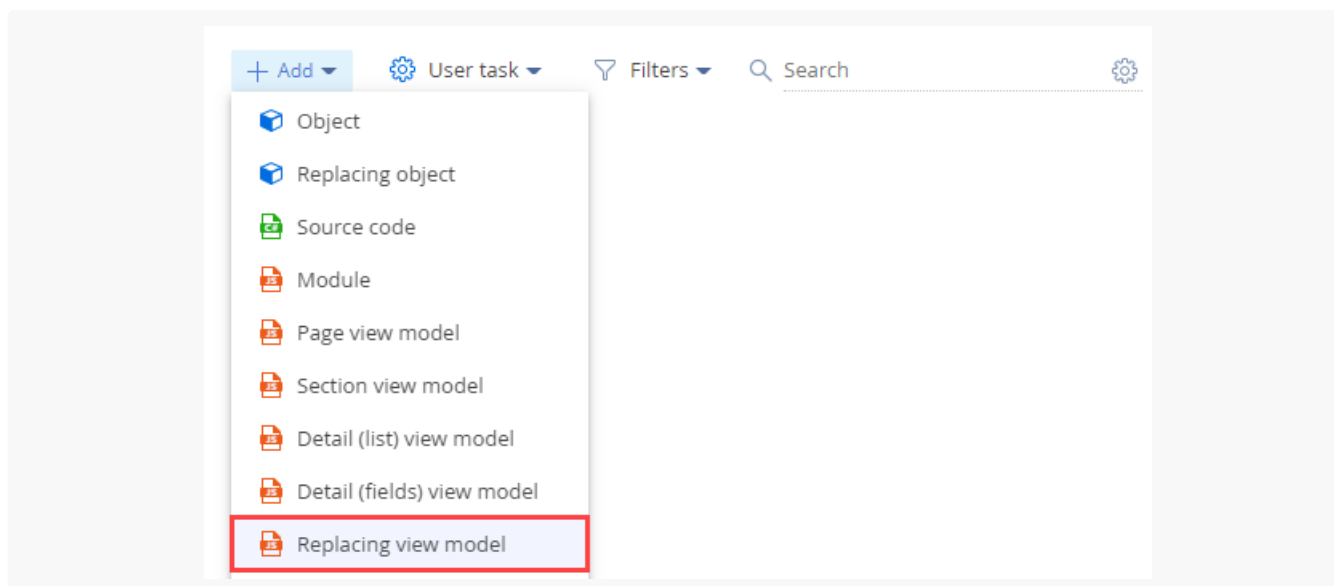
- Контейнер кнопок действий ( `ActionButtonsContainer` ) — содержит кнопки действий страницы записи.
- Контейнер левой части страницы записи ( `LeftModulesContainer` ) — содержит основные поля ввода и редактирования данных.
- Контейнер панели действий ( `ActionDashboardContainer` ) — содержит панель действий и полосу стадий.
- Контейнер вкладок ( `TabsContainer` ) — содержит вкладки с полями ввода и редактирования, которые сгруппированы по признаку, например, месту работы.

## Создать страницу записи

1. [Перейдите в раздел \[ Конфигурация \]](#) ( `[ Configuration ]` ) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ **Добавить** ] —> и выберите **вид схемы модели представления**.
  - Для **создания новой страницы записи** выберите [ *Модель представления страницы* ] ( `[ Page view model ]` ).



- Для **замещения существующей страницы записи** выберите [ *Замещающая модель представления* ] ([ *Replacing view model* ]).



### 3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — название схемы (обязательное свойство). Должно содержать префикс (по умолчанию `usr`), указанный в системной настройке [ *Префикс названия объекта* ] (код [ *SchemaNamePrefix* ]).
- [ *Заголовок* ] ([ *Title* ]) — локализуемый заголовок схемы (обязательное свойство).
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите схему страницы записи, функциональность которой необходимо наследовать.
  - Для **создания новой страницы записи** выберите "BasePageV2".
  - Для **замещения существующей страницы записи** выберите схему страницы записи, которую планируется заместить. Например, чтобы в приложении отображалась пользовательская страница контрагента, выберите схему `AccountPageV2` пакета `UIv2`.

### 4. Реализуйте логику страницы записи.

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

## Настроить страницу записи

Настройка страницы записи выполняется с помощью **элементов управления**.

**Действия** для настройки страницы записи, которые позволяет выполнять приложение:

- Добавлять стандартные элементы управления страницы.
- Изменять стандартные элементы управления страницы.
- Добавлять пользовательские элементы управления страницы.

Основные **элементы управления** страницы:

- Панель действий. Настройка панели действий страницы записи описана в статье [Панель действий](#).
- Поле. Настройка полей страницы записи описана в статье [Поле](#).
- Кнопка. Настройка кнопок страницы записи описана в статье [Кнопка](#).

## Добавить пользовательское действие на страницу записи

Creatio предоставляет возможность добавления пользовательского действия в выпадающее меню [ Действия ] ([ Actions ]) страницы записи, которое реализовано в схеме `BasePageV2` базовой страницы записи.

Чтобы **добавить пользовательское действие на страницу записи**:

1. Создайте страницу записи или замещающую страницу записи. Для создания страницы записи выполните шаги 1-3 [инструкции по созданию страницы записи](#).
2. Переопределите защищенный виртуальный метод `getActions()`, который возвращает перечень действий страницы. Перечень действий страницы является экземпляром класса `Terrasoft.BaseViewModelCollection`. Каждое действие — это модель представления.
3. В метод `addItem()` в качестве параметра передайте callback-метод `getButtonItem()`. Метод `addItem()` добавляет в коллекцию пользовательское действие.
4. В callback-метод `getButtonItem()` в качестве параметра передайте конфигурационный объект. Метод `getButtonItem()` создает экземпляр модели представления действия.
5. Реализуйте конфигурационный объект действия, который позволяет явно задать свойства модели представления действий или использовать базовый механизм привязки.

**Важно.** При замещении базовой страницы записи в методе `getActions()` схемы замещающей модели представления вызовите метод `this.callParent(arguments)`, который возвращает коллекцию действий родительской страницы записи.

Шаблон добавления пользовательского действия на страницу записи приведен ниже.

### Шаблон добавления пользовательского действия на страницу записи

```
/**
 * Возвращает коллекцию действий страницы записи.
 * @protected
 * @virtual
 * @return {Terrasoft.BaseViewModelCollection} Возвращает коллекцию действий страницы записи.
 */
getActions: function() {
    /* Список действий — экземпляр Terrasoft.BaseViewModelCollection. */
    var actionMenuItems = this.Ext.create("Terrasoft.BaseViewModelCollection");
    /* Добавление действия в коллекцию. В качестве callback-метода передается метод, который инстанс
    actionMenuItems.addItem(this.getButtonItem({
        /* Конфигурационный объект настройки действия. */
        ...
```

```

    }));
    /* Возвращает новую коллекцию действий. */
    return actionMenuItems;
}

```

# Добавить действие на страницу записи

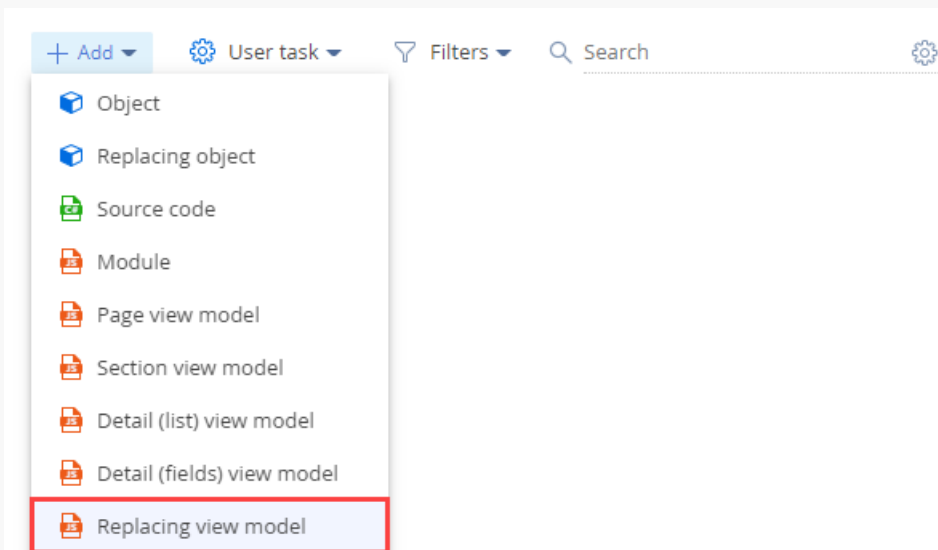
 Средний

Пример реализован для продуктов линейки Sales Creatio.

**Пример.** На страницу заказа добавить действие [ Показать дату выполнения ] ([ Show execution date ]), которое в информационном окне отображает планируемую дату выполнения заказа. Действие активно для заказов, которые находятся на стадии [ Исполнение ] ([ In progress ]).

## 1. Создать схему замещающей модели представления страницы заказа

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "OrderPageV2".
- [ Заголовок ] ([ Title ]) — "Страница редактирования заказа" ("Order edit page").



- [ Родительский объект ] ([ Parent object ]) — выберите "OrderPageV2".

The screenshot shows a 'Module' configuration window. The fields are filled as follows:

- Code:** OrderPageV2
- Title:** Order edit page
- Parent object:** Order edit page (OrderPageV2)
- Package:** sdkAddActionToEditPagePackage
- Description:** (empty)

Buttons: CANCEL, APPLY

4. Добавьте **локализуемую строку** с текстом пункта меню, который планируется добавить.

- В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
  - [ Код ] ([ Code ]) — "InfoActionCaption".
  - [ Значение ] ([ Value ]) — "Показать дату выполнения" ("Show execution date").

The screenshot shows a 'Localizable Strings' window. The fields are filled as follows:

- Code:** InfoActionCaption
- Value:** Show execution date

Buttons: CANCEL, ADD

- Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).

5. Реализуйте **логику работы пункта меню**.

Для этого в свойстве `methods` реализуйте **методы**:

- `isRunning()` — проверяет находится ли заказ на стадии [ Исполнение ] ([ In progress ]) и определяет доступность добавленного пункта.
- `showOrderInfo()` — метод-обработчик действия. В информационном окне отображает планируемую дату завершения заказа. Действие страницы записи применяется к конкретному объекту, который открыт на странице. Для доступа к значениям полей объекта страницы записи в методе-

обработчике действия необходимо использовать методы модели представления `get()` (получить значение) и `set()` (установить значение).

- `getActions()` — переопределенный базовый метод. Возвращает коллекцию действий замещающей страницы.

Исходный код схемы замещающей модели представления страницы заказа представлен ниже.

#### OrderPageV2

```
define("OrderPageV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstants)
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Order",
        /* Методы модели представления страницы записи. */
        methods: {
            /* Проверяет стадию заказа для определения доступности пункта меню. */
            isRunning: function() {
                /* Метод возвращает true, если статус заказа [Исполнение], иначе возвращает false */
                if (this.get("Status")) {
                    return this.get("Status").value === OrderConfigurationConstants.Order.OrderStatus.Complete;
                }
                return false;
            },
            /* Метод-обработчик действия. В информационном окне отображает дату выполнения заказа */
            showOrderInfo: function() {
                /* Получает дату выполнения заказа. */
                var dueDate = this.get("DueDate");
                /* Отображает информационное окно. */
                this.showInformationDialog(dueDate);
            },
            /* Переопределяет базовый виртуальный метод, который возвращает коллекцию действий */
            getActions: function() {
                /* Вызывается родительская реализация метода для получения коллекции проинициализированных действий */
                var actionMenuItems = this.callParent(arguments);
                /* Добавляет линию-разделитель. */
                actionMenuItems.addItem(this.getButtonItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Добавляет пункт меню в список действий страницы записи. */
                actionMenuItems.addItem(this.getButtonItem({
                    /* Привязывает заголовок пункта меню к локализуемой строке схемы. */
                    "Caption": {bindTo: "Resources.Strings.InfoActionCaption"},
                    /* Привязывает метод-обработчик действия. */
                    "Tag": "showOrderInfo",
                    /* Привязывает свойство доступности пункта меню к значению, которое возвращает метод isRunning */
                    "Enabled": {bindTo: "isRunning"}
                }));
            }
        }
    });
```

```

        return actionMenuItems;
    }
}
};
});

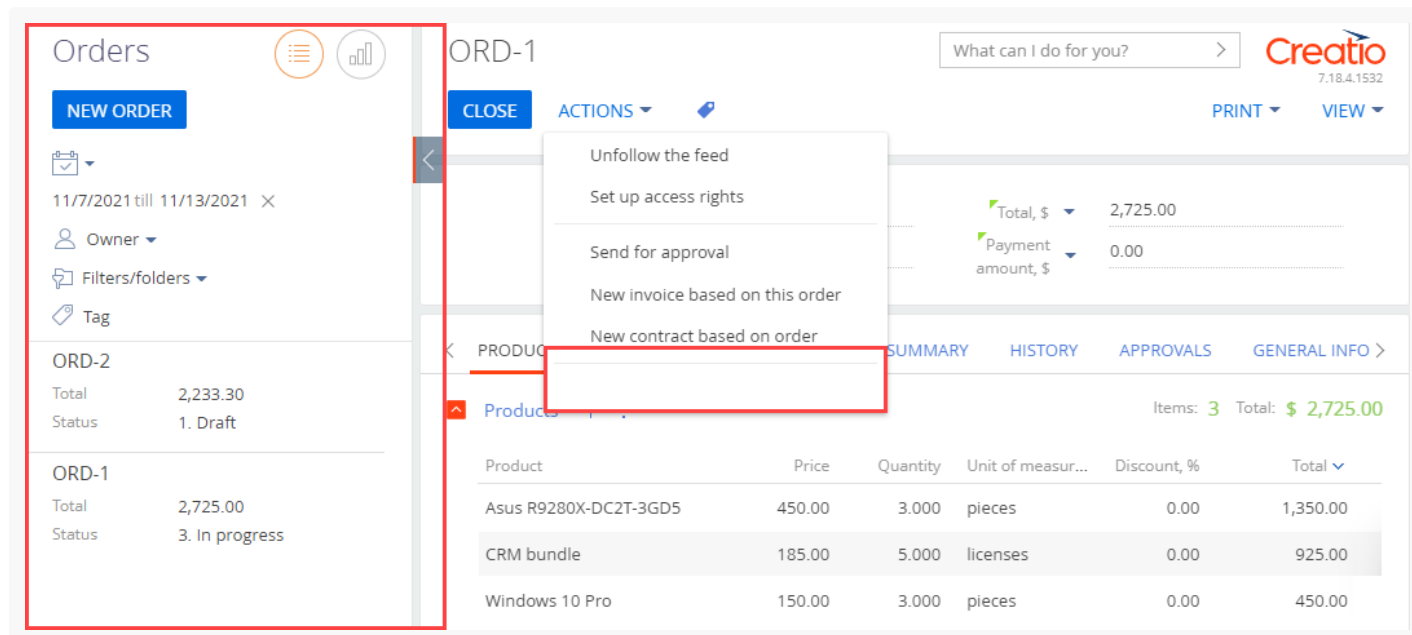
```

6. На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

В результате на страницу заказа, который находится на стадии [ Исполнение ] ([ In progress ]), добавлено действие [ Показать дату выполнения ] ([ Show execution date ]).

The screenshot displays the Creatio CRM interface for an order (ORD-1). The top navigation bar includes a search box, the Creatio logo, and version information (7.18.4.1532). The main header shows the order ID 'ORD-1' and a 'CLOSE' button. The 'ACTIONS' menu is open, showing several options: 'Unfollow the feed', 'Set up access rights', 'Send for approval', 'New invoice based on this order', 'New contract based on order', and 'Show execution date'. The 'Show execution date' option is highlighted with a red rectangle. The right sidebar shows financial details: 'Total, \$' 2,725.00 and 'Payment amount, \$' 0.00. Below this, there are tabs for 'SUMMARY', 'HISTORY', 'APPROVALS', and 'GENERAL INFORMATION'. The 'SUMMARY' tab is active, showing a table of products with columns: Product, Price, Quantity, Unit of measure, Discount, %, and Total. The table lists three items: 'Asus R9280X-DC2T-3GD5' (Price: 450.00, Quantity: 3.000, Total: 1,350.00), 'CRM bundle' (Price: 185.00, Quantity: 5.000, Total: 925.00), and 'Windows 10 Pro' (Price: 150.00, Quantity: 3.000, Total: 450.00). The total for all items is \$ 2,725.00.

В режиме вертикального представления реестра пользовательское действие страницы не отображается.

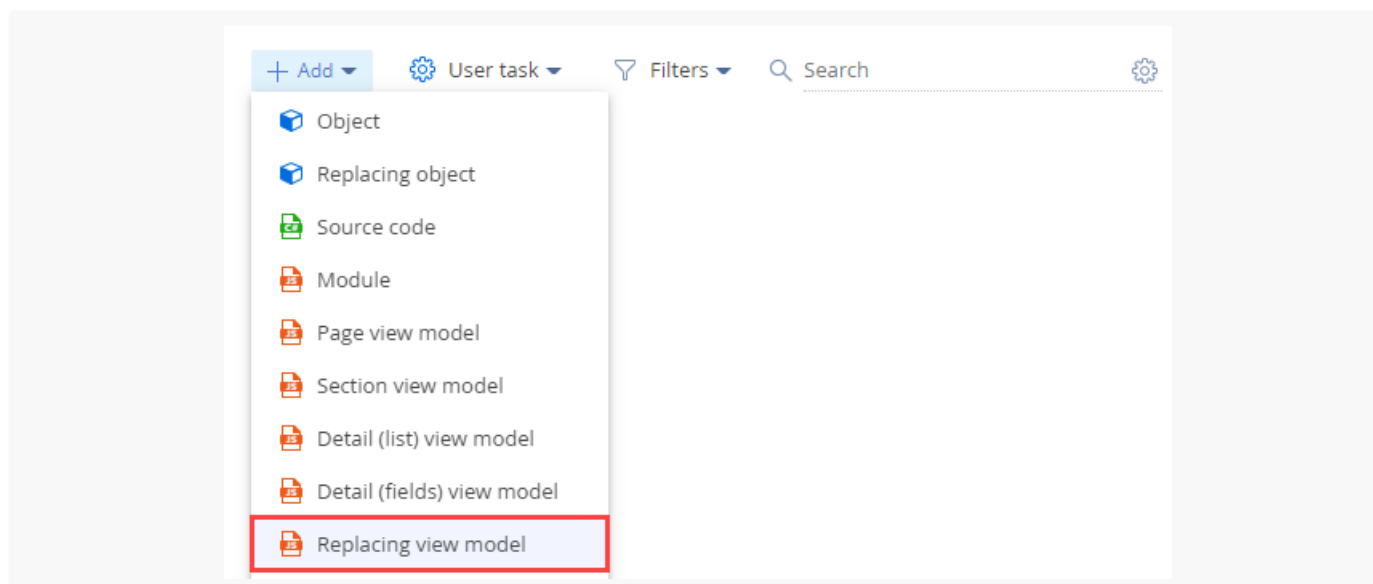


Для корректного отображения действия страницы в схему замещающей модели представления раздела необходимо добавить:

- Локализуемую строку, которая содержит текст пункта меню.
- Метод, который определяет доступность пункта меню.

## 2. Создать схему замещающей модели представления раздела

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Замещающая модель представления* ] ([ *Add* ] —> [ *Replacing view model* ]).



### 3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "OrderSectionV2".
- [ *Заголовок* ] ([ *Title* ]) — "Раздел заказа" ("Order section").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "OrderSectionV2".

4. Добавьте **локализуемую строку** с текстом пункта меню, который планируется добавить. Для этого выполните шаг 4 [алгоритма создания схемы замещающей модели представления страницы заказа](#).
5. Реализуйте **логику работы пункта меню**. Для этого в свойстве `methods` реализуйте **метод** `isRunning()`, который проверяет находится ли заказ на стадии [ *Исполнение* ] ([ *In progress* ]) и определяет доступность добавленного пункта.

Исходный код схемы замещающей модели представления страницы раздела представлен ниже.

OrderSectionV2

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* Название схемы страницы раздела. */
        entitySchemaName: "Order",
        /* Методы модели представления страницы раздела. */
        methods: {
            /* Проверяет стадию заказа для определения доступности пункта меню. */
            isRunning: function(activeRowId) {
                activeRowId = this.get("ActiveRow");
                /* Получает коллекцию данных списочного представления реестра раздела. */
                var gridData = this.get("GridData");
                /* Получает модель выбранного заказа по заданному значению первичной колонки.
                var selectedOrder = gridData.get(activeRowId);
                /* Получает свойства модели – статуса выбранного заказа. */
                var selectedOrderStatus = selectedOrder.get("Status");
```

```

        /* Метод возвращает true, если статус заказа [Исполнение], иначе возвращает false
        return selectedOrderStatus.value === OrderConfigurationConstants.Order.OrdersStatus.InProgress;
    }
}
};
});

```

6. На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Очистите кэш браузера.
2. Обновите страницу раздела [ Заказы ] ([ Orders ]).

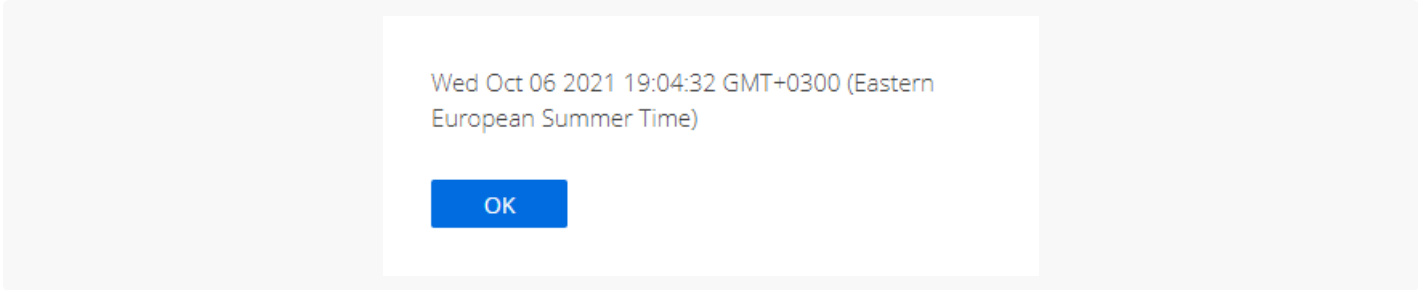
В результате выполнения примера на страницу заказа добавлено действие [ Показать дату выполнения ] ([ Show execution date ]).

Если заказ находится на стадии [ Исполнение ] ([ In progress ]), то действие [ Показать дату выполнения ] ([ Show execution date ]) активно.

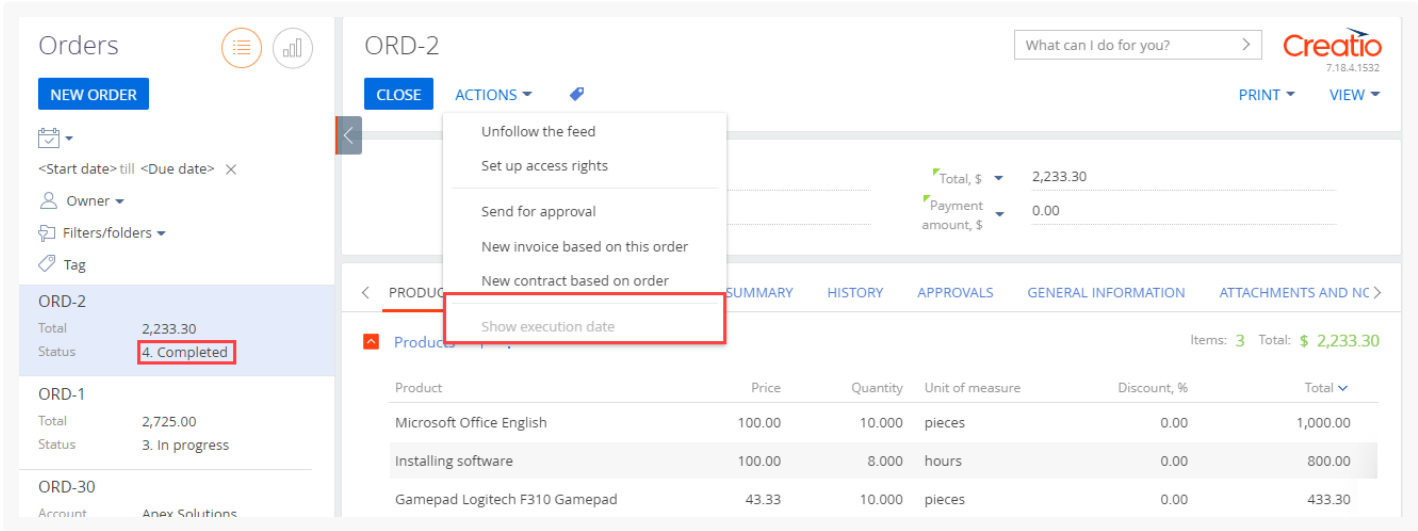
The screenshot shows the Creatio interface for the 'Orders' section. On the left, a list of orders is displayed. The order 'ORD-1' is highlighted, showing a total of 2,725.00 and a status of '3. In progress'. The main panel displays the details for 'ORD-1', including a summary of the order with a total of 2,725.00. A context menu is open over the 'ORD-1' row, showing various actions. The 'Show execution date' action is highlighted with a red box.

Product	Price	Quantity	Unit of measur...	Discount, %	Total
Asus R9280X-DC2T-3GD5	450.00	3.000	pieces	0.00	1,350.00
CRM bundle	185.00	5.000	licenses	0.00	925.00
Windows 10 Pro	150.00	3.000	pieces	0.00	450.00

В результате выбора действия [ Показать дату выполнения ] ([ Show execution date ]), в информационном окне отображается планируемая дата выполнения заказа.



Если заказ не находится на стадии [ *Исполнение* ] ([ *In progress* ]), то действие [ *Показать дату выполнения* ] ([ *Show execution date* ]) неактивно.



# Схема BasePageV2 JS

 Средний

BasePageV2 — базовая схема карточки. Реализована в пакете `nui`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#).

## Сообщения

Сообщения базовой карточки

Название	Режим	Направление	Описание
UpdatePageHeaderCaption	Адресное	Публикация	Обновляет заголовок страницы.
GridRowChanged	Адресное	Подписка	Получает идентификатор выбранной в реестре записи при ее изменении.
UpdateCardProperty	Адресное	Подписка	Изменяет значение параметра модели.

UpdateCard Header	Адресное	Подписка	Обновляет заголовок карточки.
CloseCard	Адресное	Публикация	Закрывает карточку.
OpenCard	Адресное	Подписка	Открывает карточку.
OpenCardInChain	Адресное	Публикация	Открывает цепочку карточек.
GetCardState	Адресное	Подписка	Возвращает состояние карточки.
IsCardChanged	Адресное	Публикация	Сообщает об изменении карточки.
GetActiveView Name	Адресное	Публикация	Получает имя активного представления.
GetMiniPage MasterEntity Info	Адресное	Подписка	Возвращает информацию об основной сущности мини-карточки.
GetPageTips	Адресное	Подписка	Возвращает подсказки страницы.
GetColumnInfo	Адресное	Подписка	Возвращает информацию колонки.
GetEntityColumn Changes	Широковещательное	Публикация	Отправляет информацию колонки сущности при ее изменении.
ReloadSection Row	Адресное	Публикация	Перезагружает строку раздела в соответствии со значением основного столбца.
ValidateCard	Адресное	Подписка	Запускает проверку валидности карточки.
ReInitialize Actions Dashboard	Адресное	Публикация	Запускает повторную инициализацию панели действий.
ReInitialize Actions	Адресное	Подписка	Обновляет конфиг панели действий.



Dashboard			
ReloadDashboardItems	Широковещательное	Публикация	Перезагружает элементы дашбордов.
ReloadDashboardItemsPTP	Адресное	Публикация	Перезагружает элементы дашбордов для текущей страницы.
CanChangeHistoryState	Широковещательное	Подписка	Разрешает или запрещает изменение текущего состояния истории.
IsEntityChanged	Адресное	Подписка	Возвращает измененную сущность.
IsDcmFilterColumnChanged	Адресное	Подписка	Возвращает <code>true</code> , если изменены отфильтрованные колонки.
UpdateParentLookupDisplayValue	Широковещательное	Двунаправленное	Обновляет значение родительской записи справочника по конфигу.
UpdateParentLookupDisplayValue	Широковещательное	Двунаправленное	Указывает на необходимость перезагрузки данных при следующем запуске.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

## Атрибуты

`IsLeftModulesContainerVisible` `BOOLEAN`

Признак видимости контейнера `LeftModulesContainer`.

`IsActionDashboardContainerVisible` `BOOLEAN`

Признак видимости контейнера `ActionDashboardContainer`.

`HasActiveDcm` `BOOLEAN`

Признак видимости контейнера `DcmActionsDashboardContainer`.

---

`ActionsDashboardAttributes` `CUSTOM_OBJECT`

Пользовательские атрибуты контейнера `DcmActionsDashboardContainer`.

---

`IsPageHeaderVisible` `BOOLEAN`

Флаг видимости заголовка страницы.

---

`ActiveTabName` `TEXT`

Сохранить имя активной вкладки.

---

`GridDataViewName` `TEXT`

Имя представления `GridData`.

---

`AnalyticsDataViewName` `TEXT`

Имя представления `AnalyticsData`.

---

`IsCardOpenedAttribute` `STRING`

Атрибут тела карточки, когда карточки отображена или скрыта.

---

`IsMainHeaderVisibleAttribute` `STRING`

Атрибут тела карточки, когда основной заголовок отображен или скрыт.

---

`PageHeaderColumnNames` `CUSTOM_OBJECT`

Массив имен колонок заголовка страницы.

---

`IsNotAvailable` `BOOLEAN`

Признак недоступности страницы.

---

`CanCustomize` `BOOLEAN`

Признак возможности кастомизации страницы.

---

Operation ENUM

Операции карточки.

EntityReloadScheduled BOOLEAN

Признак необходимости перезагрузки сущности при следующем запуске.

Типы данных атрибутов представлены перечислением `Terrasoft.core.enums.DataValueType`.

Перечисление `DataValueType` описано в [Библиотеке JS классов](#).

## Методы

`onDiscardChangesClick(callback, scope)`

Обрабатывает нажатие кнопки [ Отменить ] ([ Discard ]).

### Параметры

<code>{String} [callback]</code>	Функция обратного вызова.
<code>{Terrasoft.BaseViewModel} [scope]</code>	Контекст выполнения метода.

`addChangeDataViewOptions(viewOptions)`

Добавляет представления точек переключения в выпадающий список кнопки [ Вид ] ([ View ]).

### Параметры

<code>{Terrasoft.BaseViewModelCollection} viewOptions</code>	Пункты выпадающего списка кнопки [ Вид ] ([ View ]).
--	--

`addSectionDesignerViewOptions(viewOptions)`

Добавляет пункт [ Открыть мастер раздела ] ([ Open section wizard ]) в выпадающий список кнопки [ Вид ] ([ View ]).

### Параметры

<code>{Terrasoft.BaseViewModelCollection} viewOptions</code>	Пункты выпадающего списка кнопки [ Вид ] ([ View ]).
--	--

`getReportFilters()`

Возвращает коллекцию фильтров для запроса.

`initPageHeaderColumnNames()`

Инициализировать имена колонок заголовка страницы.

`getParameters(parameters)`

Возвращает значения параметров `ViewModel`.

#### Параметры

`{Array} parameters`

Имена параметров.

`setParameters(parameters)`

Задаёт параметры `ViewModel`.

#### Параметры

`{Object} parameters`

Значения параметров.

`getLookupModuleId()`

Возвращает идентификатор модуля страницы справочника.

`onReloadCard(defaultValues)`

Обработчик сообщения `ReloadCard`. Перезагружает данные сущности карточки.

#### Параметры

`{Object[]} defaultValues`

Массив значений по умолчанию.

`onGetColumnInfo(columnName)`

Возвращает информацию колонки.

#### Параметры

`{String} columnName`

Имя колонки.

`getTabsContainerVisible()`

Возвращает статус видимости вкладок контейнера.

`getPrintMenuItemVisible(reportId)`

Возвращает статус видимости пунктов выпадающего списка (т. е. отчетов) кнопки [ *Печать* ] ([ *Print* ]).

#### Параметры

<code>{String} reportId</code>	Идентификатор отчета.
--------------------------------	-----------------------

`getDataViews()`

Получает представление раздела.

`runProcess(tag)`

Запустить бизнес-процесс с помощью кнопки запуска глобальных бизнес-процессов.

#### Параметры

<code>{Object} tag</code>	Идентификатор схемы бизнес-процесса.
---------------------------	--------------------------------------

`runProcessWithParameters(config)`

Запустить бизнес-процесс с параметрами.

#### Параметры

<code>{Object} config</code>	Конфигурационный объект.
------------------------------	--------------------------

`onCanBeDestroyed(cacheKey)`

Проверяет наличие несохраненных данных. Измените `config.result` из кэша, если данные изменены, но не сохранены.

#### Параметры

<code>{String} cacheKey</code>	Ключ конфигурационного объекта в кэше.
--------------------------------	--

onValidateCard()

Отображается сообщение о некорректности, если карточка невалидна.

# Схема BaseEntityPage JS

## Оснoвы

`BaseEntityPage` — базовая схема страницы записи. Реализована в пакете `NUI`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Все схемы страниц записей должны наследовать схему `BaseEntityPage`.

## Сообщения

Сообщения базовой страницы записи

Название	Режим	Направление	Описание
<code>Update Detail</code>	Адресное	Публикация	Сообщает детали изменения карточки.
<code>CardModule Response</code>	Адресное	Двунаправленное	Сообщает об изменении страницы записи.
<code>Card Rendered</code>	Широковещательное	Двунаправленное	Сообщает об обработке страницы записи.
<code>Entity Initialized</code>	Широковещательное	Двунаправленное	Сообщает о выполнении инициализации объекта и отправляет информацию об объекте.
<code>ShowProcess Page</code>	Адресное	Публикация	Отображает страницу процесса.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

## Атрибуты

`IsEntityInitialized` `BOOLEAN`

Используется для подготовки сущности.

DefaultValues `ARRAY`

Массив значений по умолчанию для объекта.

IsModelItemsEnabled `BOOLEAN`

Признак доступности элементов модели.

DetailsConfig `CUSTOM_OBJECT`

Детали конфигурации.

Типы данных атрибутов представлены перечислением `Terrasoft.core.enums.DataValueType`.

Перечисление `DataValueType` описано в [Библиотеке JS классов](#).

## Миксины

EntityResponseValidationMixin `Terrasoft.EntityResponseValidationMixin`

Проверяет ответ сервера. Если `false`, то генерирует сообщение об ошибке и запускает диалог.

## Методы

`init(callback, scope)`

Инициализирует страницу записи.

### Параметры

<code>{Function} callback</code>	Функция обратного вызова.
<code>{Object} scope</code>	Контекст выполнения метода.

`saveDetailsInChain(next)`

Сохраняет детали в цепочку.

### Параметры

<code>{Function} next</code>	Функция обратного вызова.
------------------------------	---------------------------

`getDetailId(detailName)`

Возвращает идентификатор детали.

### Параметры

{String} detailName	Имя детали.
---------------------	-------------

loadDetail(config)

Загружает деталь. Если деталь загружена, то повторно ее отображает.

### Параметры

{Object} config	Конфигурация детали.
-----------------	----------------------

saveCheckCanEditRight(callback, scope)

Проверяет наличие прав пользователя на редактирование.

### Параметры

{Function} callback	Функция обратного вызова.
{Object} scope	Контекст выполнения метода.

getLookupDisplayValueQuery(config)

Формирует поисковый запрос для отображаемого значения.

### Параметры

{Object} config	Конфигурация детали.
-----------------	----------------------

loadLookupDisplayValue(name, value, callback, scope)

Заполняет справочные поля.

### Параметры



{String} name	Имя схемы объекта.
{String} value	Значение объекта.
{Function} callback	Функция обратного вызова.
{Object} scope	Контекст выполнения метода.

---

`canAutoCleanDependentColumns()`

Возвращает `true` , если бизнес-правило может очищать колонку объекта.