Creatio Academy

Record page

Record page

Version 8.0

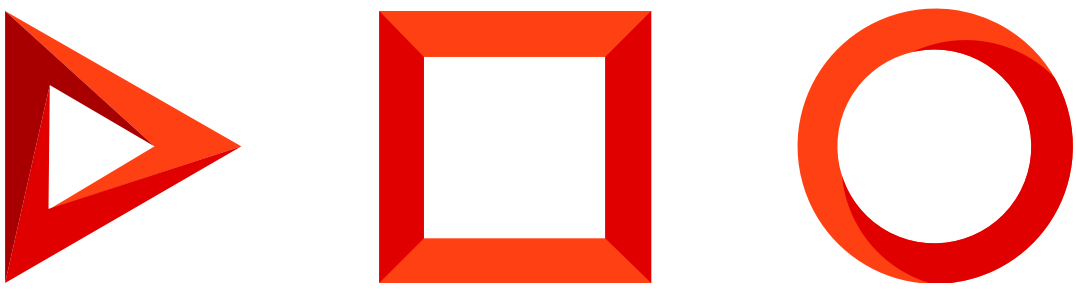


Table of Contents

Record page

Beginner

Record page is a UI element that stores the information about Creatio business objects organized as fields, tabs, details, and dashboards. The name of the record page corresponds to the name of the Creatio object. For example, account page, contact page, etc. The **purpose** of the record page is management of section list records. Each Creatio section includes one or more record pages.

Each record page is represented by a client module schema. For example, the `ContactPageV2` schema of the `Uiv2` package configures the contact page. The `BasePageV2` schema of the `NUI` package implements the functionality of a base record page. Record page schemas must inherit the `BasePageV2` schema.

The record page **types** are as follows:

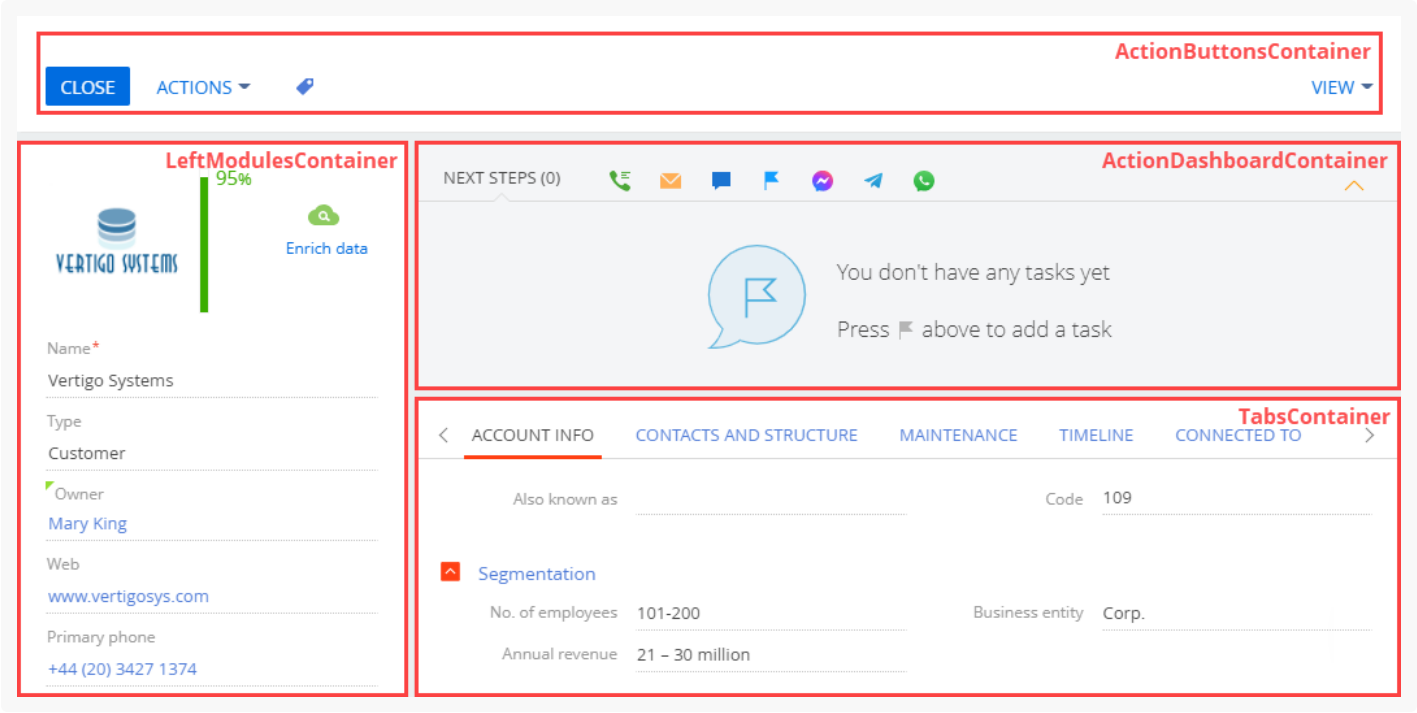
- The **page that adds records**. Creates a section list record.
- The **page that edits records**. Modifies an existing section list record.

Record page containers

Creatio stores the UI elements related to the record page in the corresponding containers. Configure the containers in the base schema of the record page or the schema of the replacing record page. The record page type does not affect the containers.

> **Note.** Creatio uses the meta names of HTML containers. The actual IDs of the corresponding HTML elements on the page are generated based on the meta names.

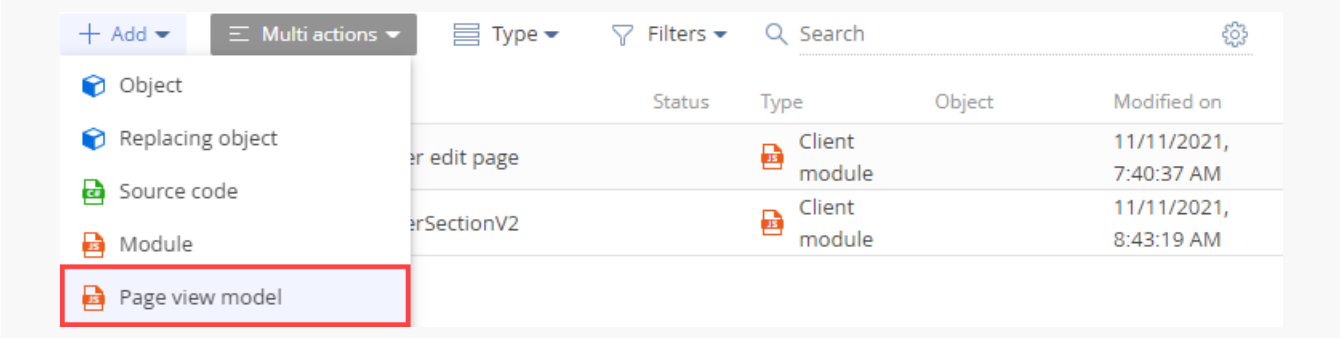
View the main record page **containers** in the figure below.



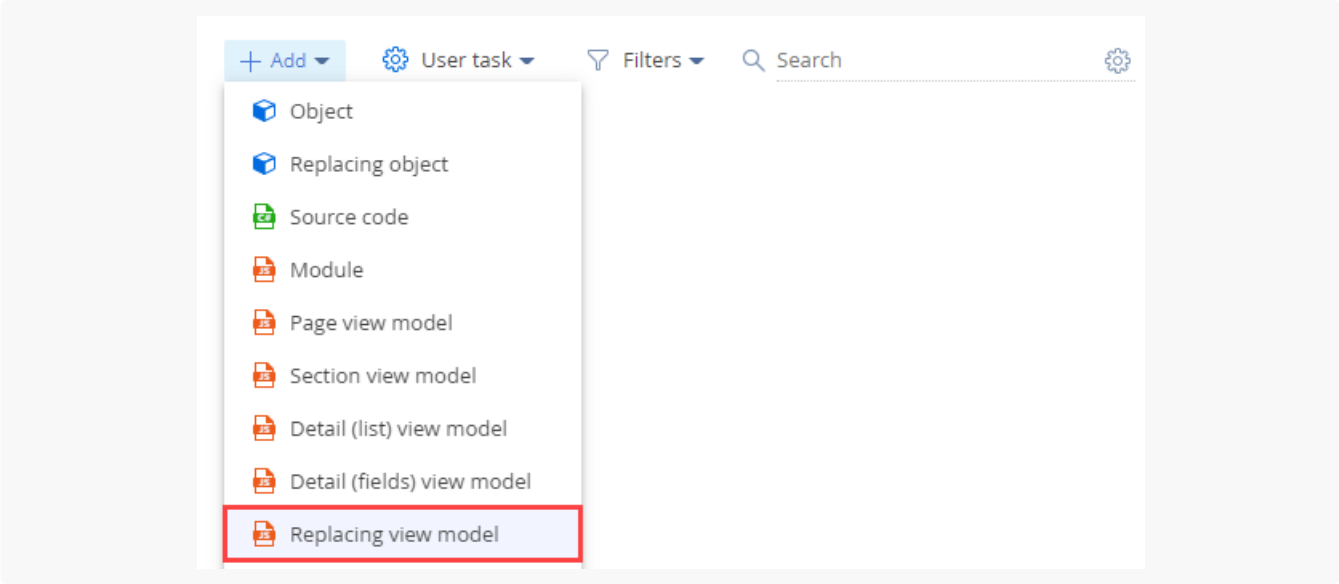
- `ActionButtonsContainer` stores the action buttons of the record page.
- `LeftModulesContainer` stores the main data input and edit fields.
- `ActionDashboardContainer` stores the action panel and workflow bar.
- `TabsContainer` stores the tabs that group input and edit fields by a category. For example, current employment.

Create a record page

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] on the section list toolbar and select the **type of the view model schema**.
   - Select [*Page view model*] to **create a new record page**.



   - Select [*Replacing view model*] to **replace an existing record page**.



3. Fill out the following **schema properties**.

   - [*Code*] is the schema name. Required. The code must contain a prefix specified in the [*Prefix for object name*] (the [*SchemaNamePrefix*] code) system setting. The default prefix is `Usr`.
   - [*Title*] is the localizable schema title. Required.
   - [*Parent object*] is the record page schema whose functionality to inherit.
     - Select "`BasePageV2`" to **create a new record page**.
     - Select the schema of the needed record page to **replace an existing record page**. For example, select the `AccountPageV2` schema of the `Uiv2` package to display a custom account page in Creatio.

4. Implement the record page logic.
5. Click [*Save*] on the Module Designer's toolbar.

Set up the record page

Use **controls** to set up the record page.

Creatio lets you set up the record page in the following **ways**:

- Add standard page controls.
- Edit standard page controls.
- Add custom page controls.

The main page **controls** are as follows:

- Action dashboard. Learn more about setting up the action dashboard in a separate article: Action dashboard.
- Field. Learn more about setting up page fields in a separate article: Field.
- Button. Learn more about setting up buttons in a separate article: Button.

Add a custom action to a record page

Creatio lets you add a custom action to the [*Actions*] drop-down menu of the record page. The `BasePageV2` schema of the base record page implements the menu.

To **add a custom action to the record page**:

1. Create a record page or replacing record page. To do this, take steps 1-3 in the instruction to create a record page.
2. Override the protected `getActions()` virtual method that returns the page action list. The page action list is an instance of the `Terrasoft.BaseViewModelCollection` class. Each action is a view model.
3. Pass the `getButtonMenuItem()` method to the `addItem()` method as a parameter. The `addItem()` method adds a custom action to the collection.
4. Pass the configuration object to the `getButtonMenuItem()` callback method as a parameter. The `getButtonMenuItem()` method creates an instance of the action view model.
5. Implement the action configuration object that lets you set the properties of the action view model explicitly or use the base binding mechanism.

Attention. If you replace the base record page, call the `this.callParent(arguments)` method in the `getActions()` method of the replacing view model schema. `this.callParent(arguments)` returns the action collection of the parent record page.

View the template to add a custom action to a record page below.

Template to add a custom action to a record page

```
/**
* Return the action collection of the record page.
* @protected
* @virtual
* @return {Terrasoft.BaseViewModelCollection} Return the action collection of the record page.
*/
getActions: function() {
 /* The action list is an instance of Terrasoft.BaseViewModelCollection. */
 var actionMenuItems = this.Ext.create("Terrasoft.BaseViewModelCollection");
 /* Add the action to the collection. Pass the method that instantiates the action model instanc
 actionMenuItems.addItem(this.getButtonMenuItem({
  /* The configuration object that sets up the action. */
  ...
 }));
 /* Return the new action collection. */
 return actionMenuItems;
}
```

Add an action to the record page

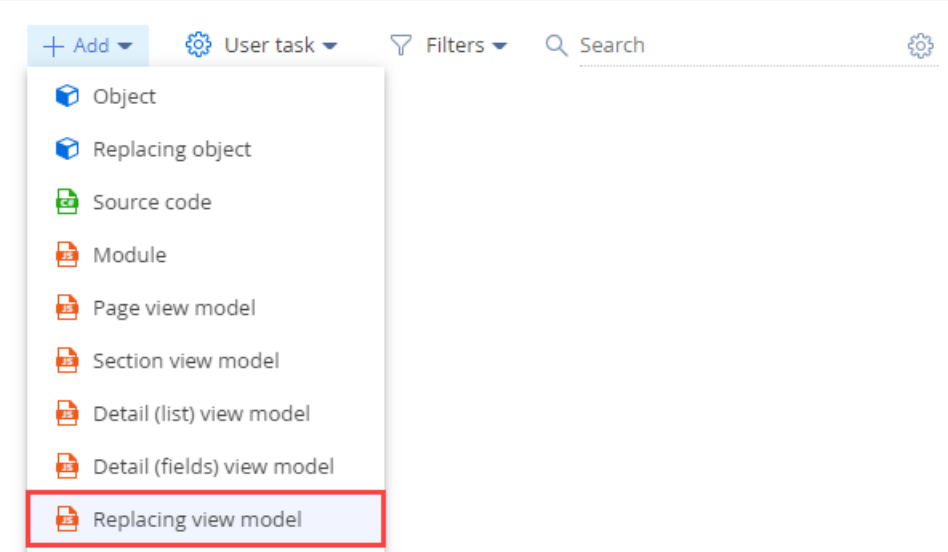
Medium

The example is relevant to Sales Creatio products.

Example. Add a [*Show execution date*] action to the order page. The action must bring up a notification box that contains the scheduled order execution date. The action must be active for orders at the [*In progress*] stage.

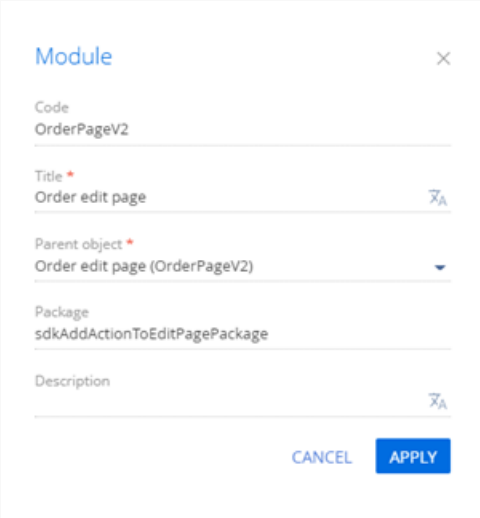
1. Create a replacing view model schema of the order page

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the following **schema properties**.

   - Set [*Code*] to "`OrderPageV2`."
   - Set [*Title*] to "`Order edit page`."
   - Set [*Parent object*] to "`OrderPageV2`."

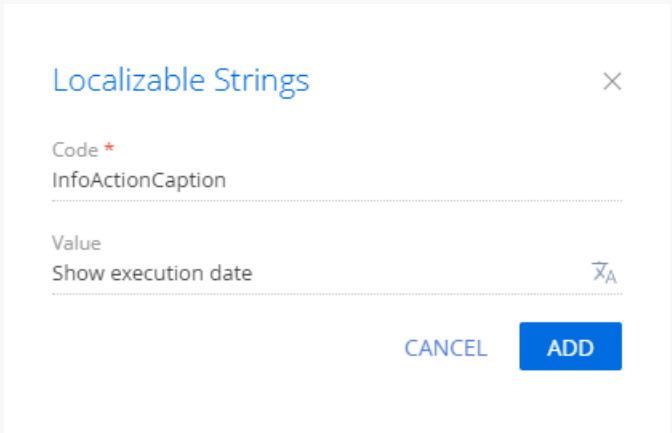


4. Add a **localizable string** that contains the menu item caption.

   a. Click the + button in the context menu of the [*Localizable strings*] node.

   b. Fill out the **localizable string properties**:

   - Set [*Code*] to "`InfoActionCaption`."
   - Set [*Value*] to "`Show execution date`."



   e. Click [*Add*] to add a localizable string.

5. Implement the **menu item logic**.

   To do this, implement the following **methods** in the `methods` property:

   - `isRunning()` – checks whether the order is at the [*In progress*] stage and the menu item is available.
   - `showOrderInfo()` – the action handler method. Displays the scheduled order execution date in the notification box. The record page action applies to a specific object opened on the page. To access the field values of the record page object in the action handler method, use the `get()` (retrieve the value) and `set()` (assign the value) view model methods.
   - `getActions()` – an overridden base method. Returns the action collection of the replacing page.

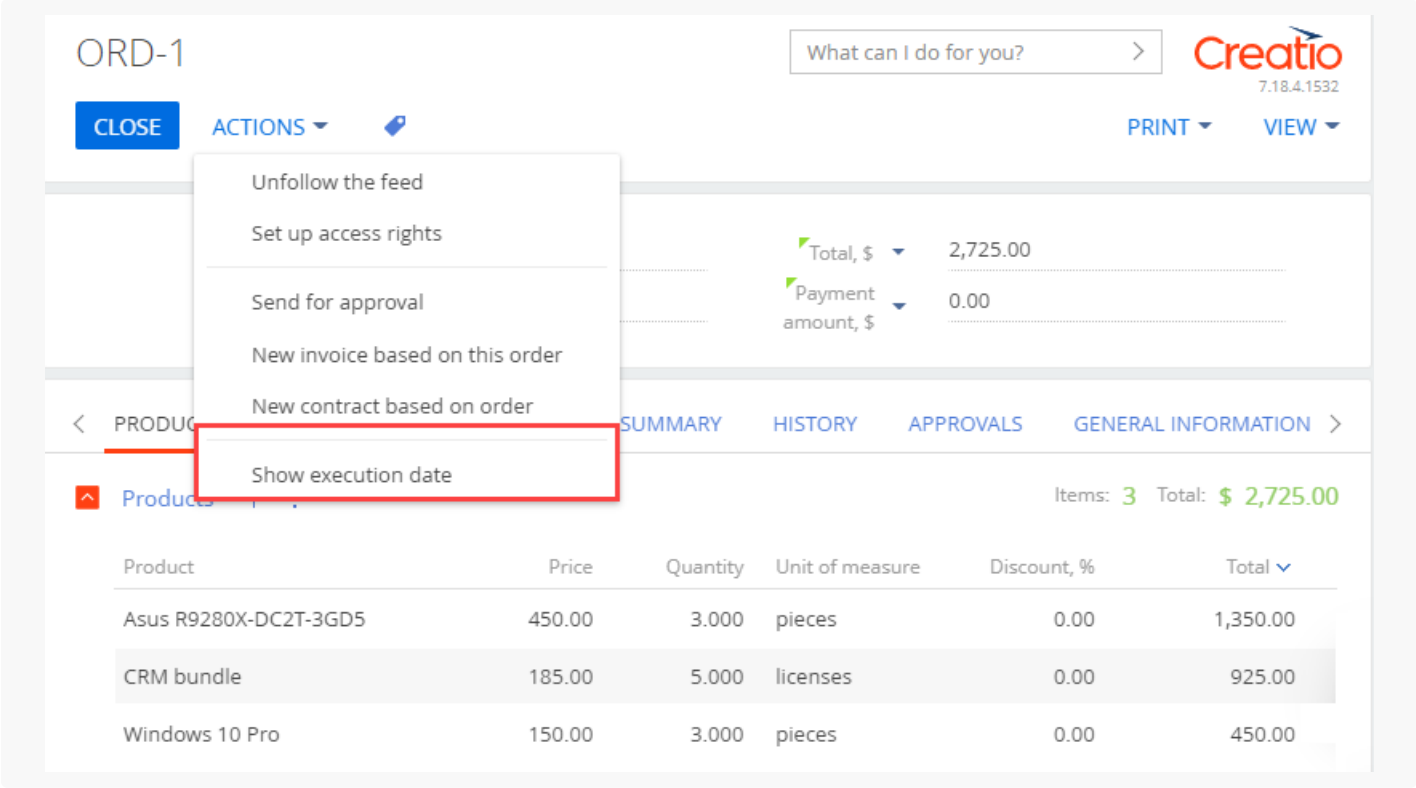
   View the source code of the order page's replacing view model below.

OrderPageV2

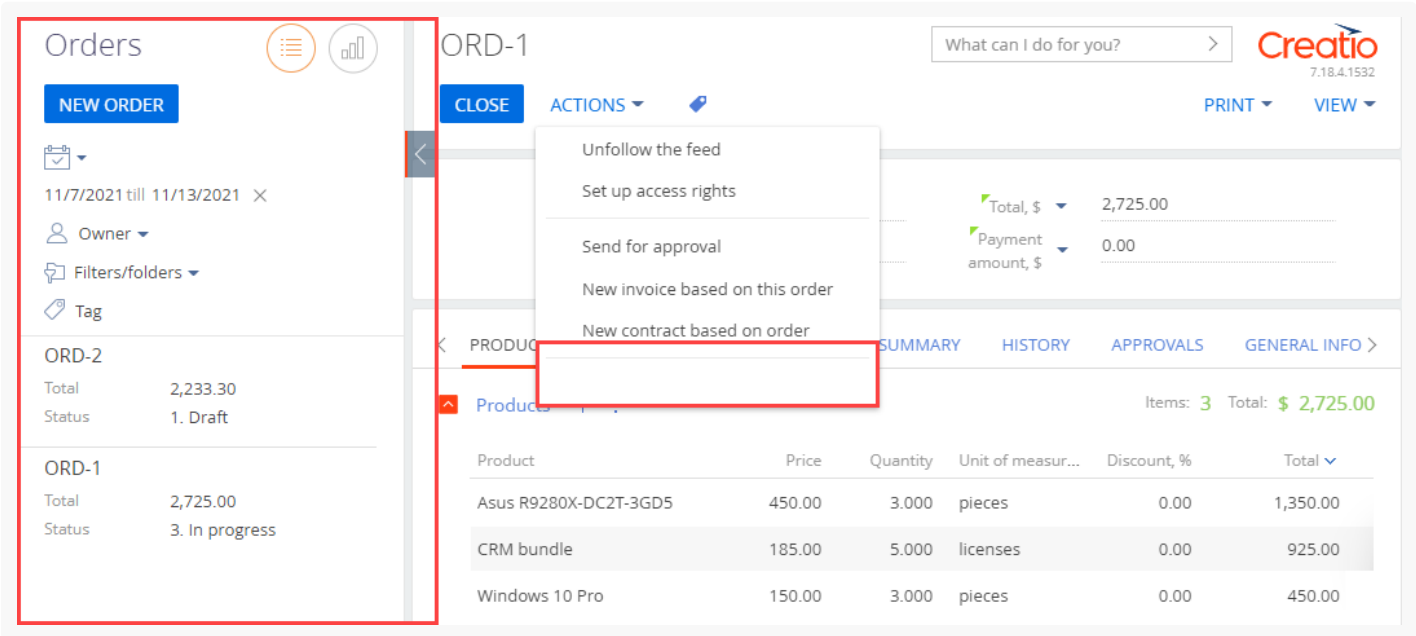
```
define("OrderPageV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstants)
    return {
        /* The name of the record page object's schema. */
        entitySchemaName: "Order",
        /* The methods of the record page's view model. */
        methods: {
            /* Check the order stage to determine the menu item availability. */
            isRunning: function() {
                /* Return true if the order status is [In progress], return false otherwise.
                if (this.get("Status")) {
                    return this.get("Status").value === OrderConfigurationConstants.Order.Ord
                }
                return false;
            },
            /* The action handler method. Displays the order execution date in the notificati
            showOrderInfo: function() {
                /* Retrieve the order execution date. */
                var dueDate = this.get("DueDate");
                /* Display the notification box. */
                this.showInformationDialog(dueDate);
            },
            /* Override the base virtual method that returns the action collection of the rec
            getActions: function() {
                /* Call the parent method implementation to retrieve the collection of the ba
                var actionMenuItems = this.callParent(arguments);
                /* Add a separator line. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Add a menu item to the action list of the record page. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    /* Bind the menu item caption to the localizable schema string. */
                    "Caption": {bindTo: "Resources.Strings.InfoActionCaption"},
                    /* Bind the action handler method. */
                    "Tag": "showOrderInfo",
                    /* Bind the property of the menu item availability to the value returned
                    "Enabled": {bindTo: "isRunning"}
                }));
                return actionMenuItems;
            }
        }
    };
});
```

6. Click [*Save*] on the Designer's toolbar.

As a result, Creatio will add the [*Show execution date*] action to the page of the order at the [*In progress*] stage.



Creatio does not display the custom page action in the vertical list view.

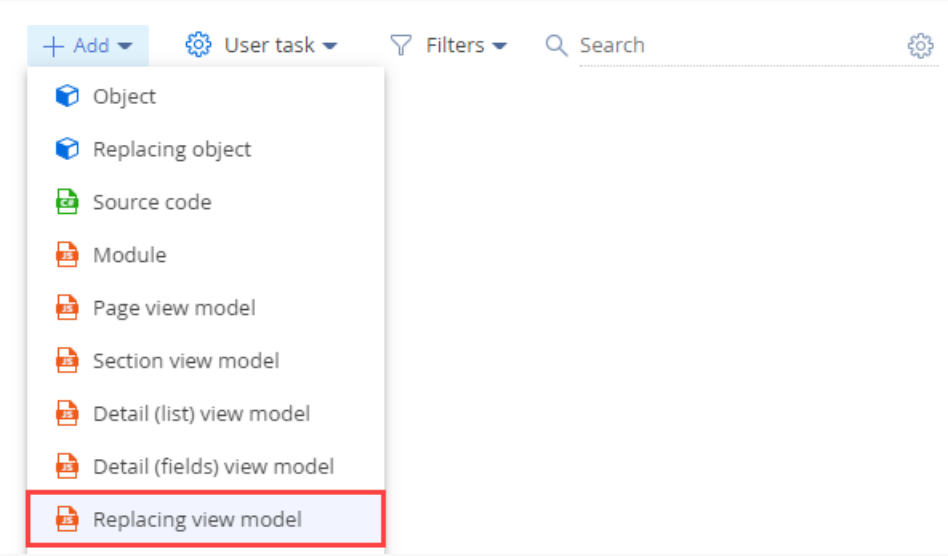


To ensure the page action is displayed correctly, add the following to the replacing view model schema of the section:

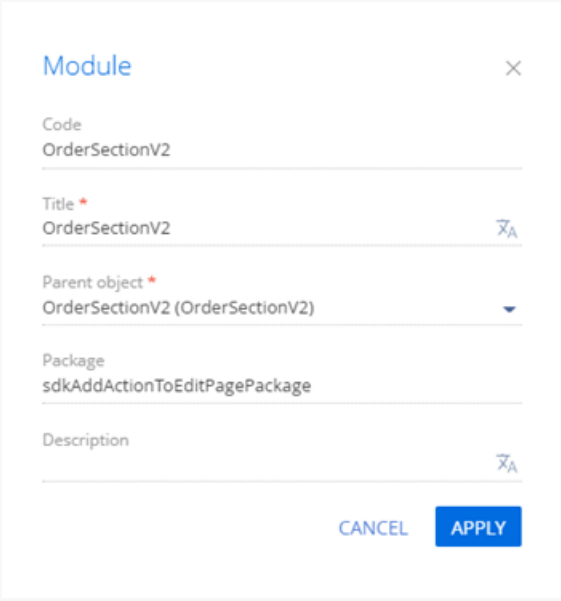
- The localizable string that contains the menu item caption.
- The method that determines the menu item availability.

2. Create a replacing view model schema of the section

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the following **schema properties**.
   - Set [*Code*] to "`OrderSectionV2`."
   - Set [*Title*] to "`Order section`."
   - Set [*Parent object*] to "`OrderSectionV2`."



4. Add a **localizable string** that contains the menu item caption. To do this, take step 4 of the procedure to create a replacing view model schema of the order page.
5. Implement the **menu item logic**. To do this, implement the `isRunning()` **method** in the `methods` property. The method checks if the order is at [*In progress*] stage and determines the menu item availability.

   View the source code of the section page's replacing view model schema below.

   `OrderSectionV2`

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* The name of the section page schema. */
        entitySchemaName: "Order",
        /* The methods of the section page's view model. */
        methods: {
            /* Check the order stage to determine the menu item availability. */
            isRunning: function(activeRowId) {
                activeRowId = this.get("ActiveRow");
                /* Retrieve the data collection of the section list's list view. */
                var gridData = this.get("GridData");
                /* Retrieve the order model by the specified primary column value. */
                var selectedOrder = gridData.get(activeRowId);
                /* Retrieve the properties of the order status model. */
                var selectedOrderStatus = selectedOrder.get("Status");
                /* Return true if the order status is [In progress], return false otherwise.
                return selectedOrderStatus.value === OrderConfigurationConstants.Order.OrderS
            }
        }
    };
});
```

6. Click [*Save*] on the Designer's toolbar.

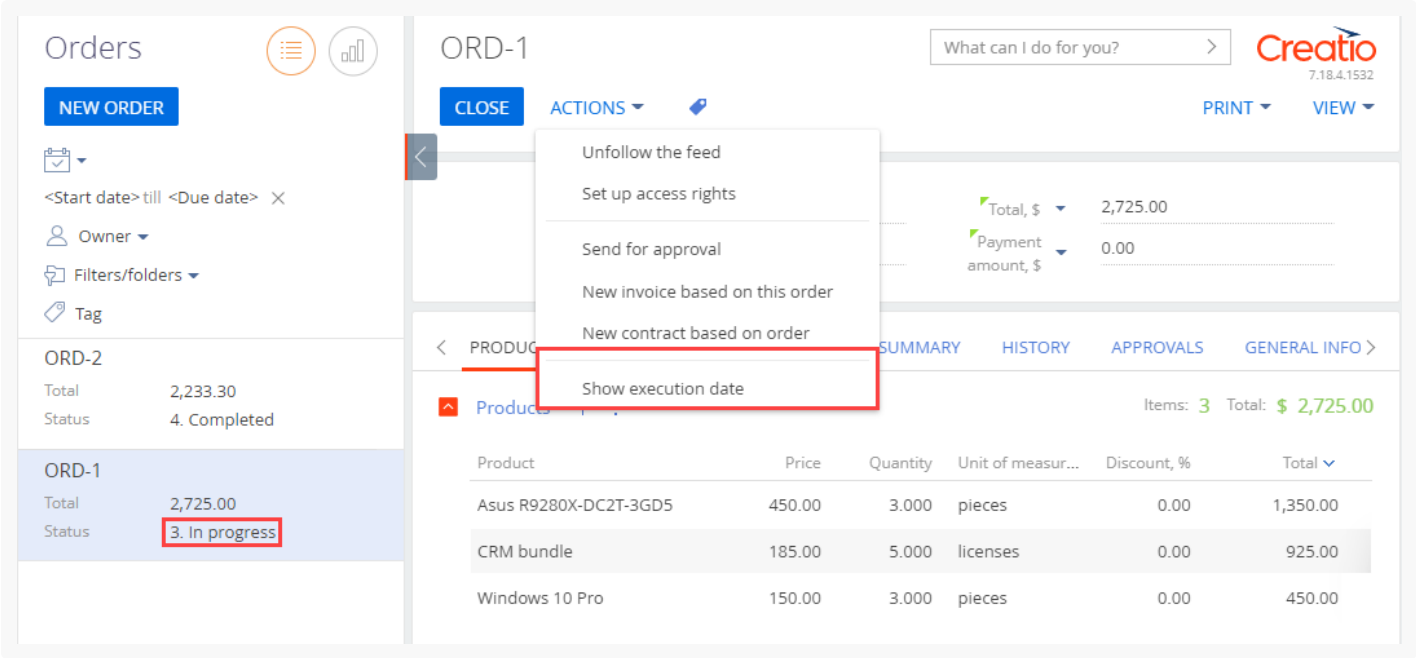
Outcome of the example

To **view the outcome of the example**:

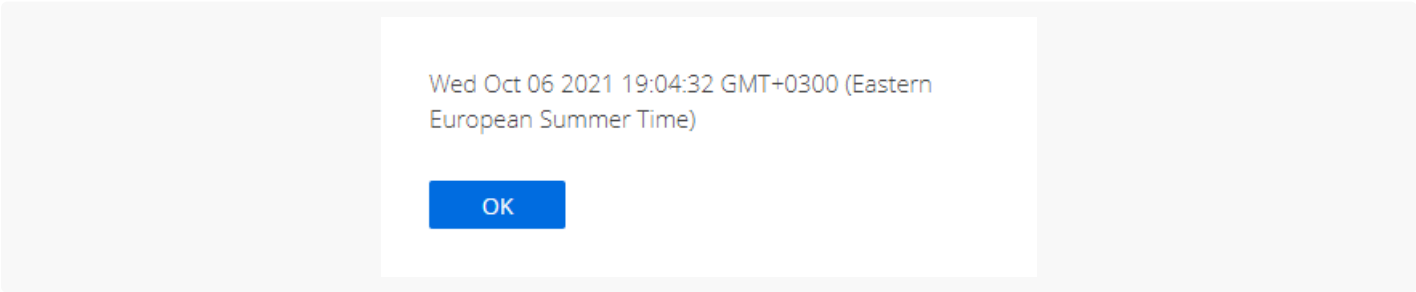
1. Clear the browser cache.
2. Refresh the [*Orders*] section page.

As a result, Creatio will add the [*Show execution date*] action to the order page.

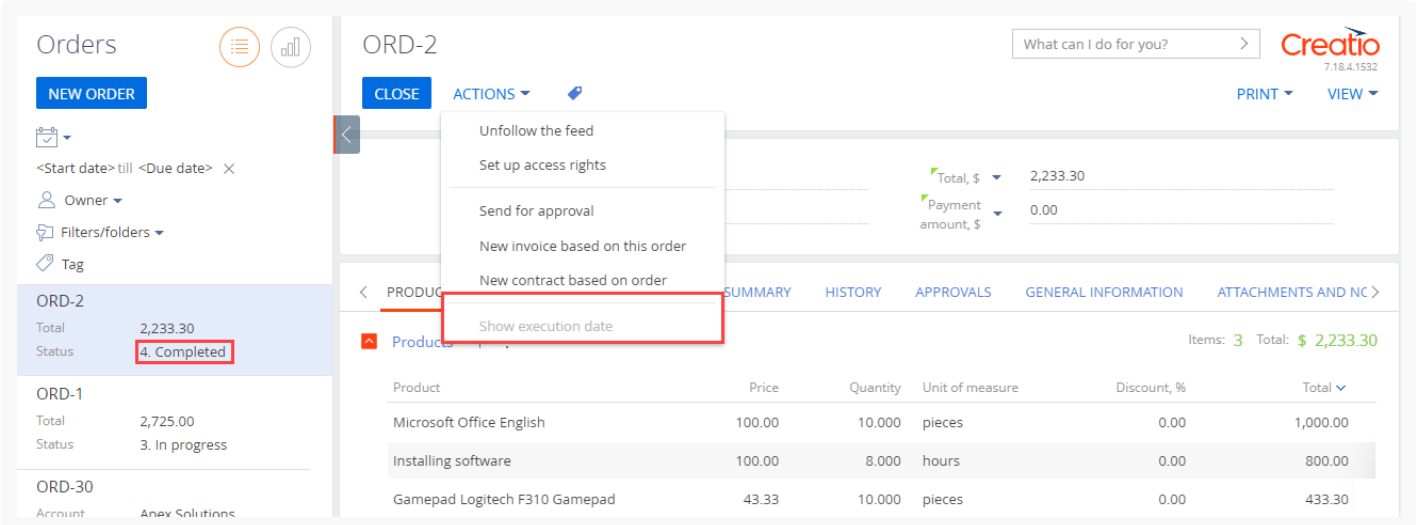
If the order is at the [*In progress*] stage, the [*Show execution date*] action will be active.



Run the [*Show execution date*] action to bring up the notification box that displays the scheduled order execution date.



If the order is not at the [*In progress*] stage, the [*Show execution date*] action will be inactive.



BaseEntityPage schema JS

BaseEntityPage schema

Medium

`BaseEntityPage` is the base schema of the record page. Implemented in the `NUI` package. This is a view model schema. Learn more about the schema properties in a separate article: Client schema structure. Record page schemas must inherit the `BaseEntityPage` schema.

Messages

Messages of the base record page

Name	Mode	Direction	Description
`UpdateDetail`	Address	Publish	Communicates the change details of the record page.
`CardModule Response`	Address	Bidirectional	Communicates the record page change.
`CardRendered`	Broadcasting	Bidirectional	Communicates the record page processing.
`Entity Initialized`	Broadcasting	Bidirectional	Communicates the object initialization and sends the object information.
`ShowProcess Page`	Address	Publish	Displays the process page.

The `Terrasoft.core.enums.MessageMode` enumeration represents the message modes, and the `Terrasoft.core.enums.MessageDirectionType` enumeration represents the message directions. Learn more about the `MessageMode` enumeration in the JS class library. Learn more about the `MessageDirectionType` enumeration in the JS class library.

Attributes

`IsEntityInitialized` BOOLEAN

Initializes the entity.

`DefaultValues` ARRAY

The array of default values for the object.

`IsModelItemsEnabled` BOOLEAN

The availability flag for model elements.

`DetailsConfig` `CUSTOM_OBJECT`

The configuration details.

The `Terrasoft.core.enums.DataValueType` enumeration represents the attribute types. Learn more about the `DataValueType` enumeration in the JS class library.

Mixins

`EntityResponseValidationMixin` `Terrasoft.EntityResponseValidationMixin`

Checks the server response. If `false`, generates an error message and launches a dialog.

Methods

`init(callback, scope)`

Initializes the record page.

Parameters

`{Function} callback`	The callback function.
`{Object} scope`	The method execution context.

`saveDetailsInChain(next)`

Saves details to a chain.

Parameters

`{Function} next`	The callback function.

`getDetailId(detailName)`

Returns the detail ID.

Parameters

`{String} detailName`	The detail name.

`loadDetail(config)`

Loads the detail. If the detail is loaded, displays it once more.

Parameters

`{Object} config`	The detail configuration.

`saveCheckCanEditRight(callback, scope)`

Checks whether the user has edit permission.

Parameters

`{Function} callback`	The callback function.
`{Object} scope`	The method execution context.

`getLookupDisplayValueQuery(config)`

Generates a search query for the display value.

Parameters

`{Object} config`	The detail configuration.

`loadLookupDisplayValue(name, value, callback, scope)`

Populates the lookup fields.

Parameters

`{String} name`	The object schema name.
`{String} value`	The object value.
`{Function} callback`	The callback function.
`{Object} scope`	The method execution context.

`canAutoCleanDependentColumns()`

Returns `true` if a business rule can clear the object column.

BasePageV2 schema JS

Medium

`BasePageV2` is a base schema of a mini page. Implemented in the `NUI` package. This is a view model schema. Learn more about the schema properties in a separate article: Client schema.

Messages

Messages of a base mini page

Name	Mode	Direction	Description
`UpdatePageHeaderCaption`	Address	Publishing	Updates the page header.
`GridRowChanged`	Address	Subscription	Retrieves the ID of the selected list record when the record changes.
`UpdateCardProperty`	Address	Subscription	Changes the model parameter value.
`UpdateCardHeader`	Address	Subscription	Updates the add page header.
`CloseCard`	Address	Publishing	Closes the add page.
`OpenCard`	Address	Subscription	Opens the add page.
`OpenCardInChain`	Address	Publishing	Opens an add page chain.
`GetCardState`	Address	Subscription	Returns the add page status.
`IsCardChanged`	Address	Publishing	Informs of the add page change.
`GetActiveViewName`	Address	Publishing	Retrieves the name of the active view.
`GetMiniPageMasterEntityInfo`	Address	Subscription	Retrieves data about the master entity of the mini add page.
`GetPageTips`	Address	Subscription	Retrieves page tooltips.
`GetColumnInfo`	Address	Subscription	Retrieves column data.
`GetEntityColumnChanges`	Broadcasting	Publishing	Sends entity column data when the column changes.
`ReloadSectionRow`	Address	Publishing	Reloads the section row depending on the primary column value.

`ValidateCard`	Address	Subscription	Launches the add page validation.
`ReInitializeActionsDashboard`	Address	Publishing	Relaunches the action dashboard validation.
`ReInitializeActionsDashboard`	Address	Subscription	Updates the action dashboard configuration.
`ReloadDashboardItems`	Broadcasting	Publishing	Reloads the dashboard elements.
`ReloadDashboardItemsPTP`	Address	Publishing	Reloads the dashboard elements on the current page.
`CanChangeHistoryState`	Broadcasting	Subscription	Allows or prohibits changing the current history state.
`IsEntityChanged`	Address	Subscription	Returns the changed entity.
`IsDcmFilterColumnChanged`	Address	Subscription	Returns `true` if the filtered columns changed.
`UpdateParentLookupDisplayValue`	Broadcasting	Bidirectional	Updates the parent lookup record in accordance with the configuration.
`UpdateParentLookupDisplayValue`	Broadcasting	Bidirectional	States that data must be reloaded on the following launch.

The `Terrasoft.core.enums.MessageMode` enumeration represents the message modes, and the `Terrasoft.core.enums.MessageDirectionType` enumeration represents the message directions. Learn more about the `MessageMode` enumeration in the JS class library. Learn more about the `MessageDirectionType` enumeration in the JS class library.

Attributes

`IsLeftModulesContainerVisible` `BOOLEAN`

The visibility flag of `LeftModulesContainer`.

`IsActionDashboardContainerVisible` `BOOLEAN`

The visibility flag of `ActionDashboardContainer`.

`HasActiveDcm` `BOOLEAN`

The visibility flag of `DcmActionsDashboardContainer`.

`ActionsDashboardAttributes` `CUSTOM_OBJECT`

Custom attributes of `DcmActionsDashboardContainer`.

`IsPageHeaderVisible` `BOOLEAN`

The visibility flag of the page title.

`ActiveTabName` `TEXT`

Saves the name of the active tab.

`GridDataViewName` `TEXT`

The name of the `GridData` view.

`AnalyticsDataViewName` `TEXT`

The name of the `AnalyticsData` view.

`IsCardOpenedAttribute` `STRING`

Attribute of the add page body when the add page is displayed or hidden.

`IsMainHeaderVisibleAttribute` `STRING`

Attribute of the add page body when the main header is displayed or hidden.

`PageHeaderColumnNames` `CUSTOM_OBJECT`

The array of page header column names.

`IsNotAvailable` `BOOLEAN`

Flag that marks the page as not available.

`CanCustomize` `BOOLEAN`

Flag that marks the page as customizable.

`Operation` ENUM

Operations of the add page.

`EntityReloadScheduled` BOOLEAN

Flag that forces entity reload on the following launch.

The `Terrasoft.core.enums.DataValueType` enumeration represents the attribute data types. Learn more about the `DataValueType` enumeration in the JS class library.

Methods

`onDiscardChangesClick(callback, scope)`

Handles the [*Discard*] button click event.

Parameters

`{String} [callback]`	The callback function.
`{Terrasoft.BaseViewModel} [scope]`	The method execution scope.

`addChangeDataViewOptions(viewOptions)`

Adds switchpoint views to the drop-down list of the [*View*] button.

Parameters

`{Terrasoft.BaseViewModelCollection} viewOptions`	Items in the drop-down list of the [*View*] button.

`addSectionDesignerViewOptions(viewOptions)`

Adds the [*Open section wizard*] item to the drop-down list of the [*View*] button.

Parameters

`{Terrasoft.BaseViewModelCollection} viewOptions`	Items in the drop-down list of the [*View*] button.

`getReportFilters()`

Returns the query filter collection.

`initPageHeaderColumnNames()`

Initializes the column names of the page header.

`getParameters(parameters)`

Returns the values of `ViewModel` parameters.

Parameters

`{Array} parameters`	Parameter names.

`setParameters(parameters)`

Sets the `ViewModel` parameters.

Parameters

`{Object} parameters`	Parameter values.

`getLookupModuleId()`

Returns the ID of the lookup page module.

`onReloadCard(defaultValues)`

`ReloadCard` message handler. Reloads the data of the add page entity.

Parameters

`{Object[]} defaultValues`	Array of default values.

`onGetColumnInfo(columnName)`

Returns the column information.

Parameters

`{String} columnName`	The name of the column.

`getTabsContainerVisible()`

Returns the visibility status of container tabs.

`getPrintMenuItemVisible(reportId)`

Returns the visibility status of the [*Print*] button's drop-down list items (i. e., reports).

Parameters

`{String} reportId`	Report ID.

`getDataViews()`

Retrieves the section view.

`runProcess(tag)`

Runs a business process using the start button for global business processes.

Parameters

`{Object} tag`	ID of the business process schema.

`runProcessWithParameters(config)`

Runs a business process with parameters.

Parameters

`{Object} config`	The configuration object.

`onCanBeDestroyed(cacheKey)`

Checks for unsaved data. Edit the cached `config.result` if the data was changed but not saved.

Parameters

`{String} cacheKey`	The key of the cached configuration object.

`onValidateCard()`

Displays an error message if the add page is invalid.