Creatio Academy

Interface elements

Mini page

Version 8.0

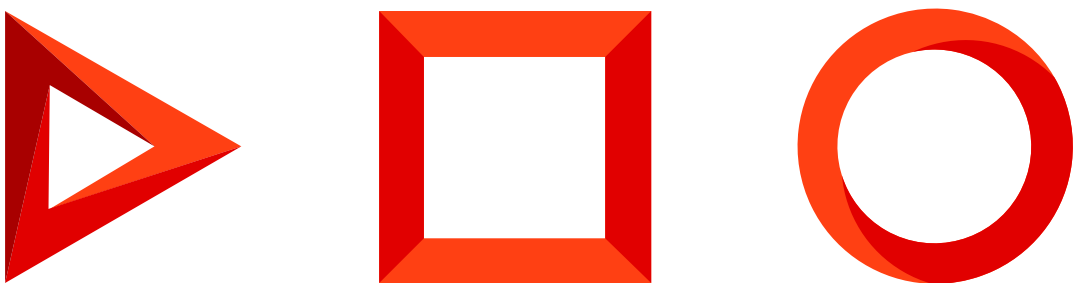


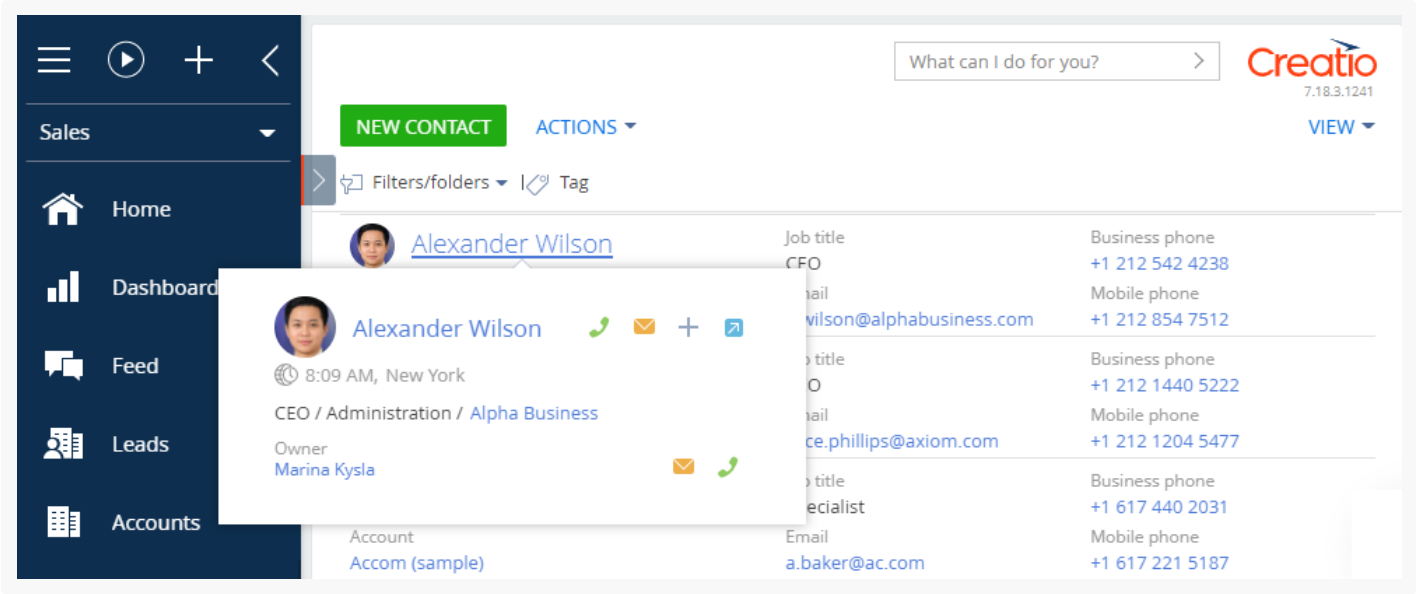
Table of Contents

Mini page

Beginner

A **mini page** is a short version of a record page with a limited number of fields. Use mini pages to view or edit the record data quickly without opening the full page. The **purpose** of a mini page is to streamline adding, editing, and viewing records. The set of fields is configured separately and therefore is different for each mini page type.

The contact mini page



You can create a mini page for any Creatio object.

Learn more about working with mini pages in user documentation: Work with mini pages.

View model schema of a mini page

Creato IDE uses view model schemas to implement mini pages.

The view model schema of a mini page **lets you set up**:

- the content of the mini page
- the position of the mini page's UI elements
- the behavior of the mini page's UI elements

For example, the `ContactMiniPage` schema configures the contact mini page and the `AccountMiniPage` schema configures the account mini page. The mini page schemas inherit from the `BaseMiniPage` schema of the `NUI` package.

The structure of the mini page's view model schema is the same as the general structure of the client view model schema.

The **required properties** of the mini page schema structure include:

- `entitySchemaName`, the name of the object schema to bind the mini page
- `diff`, the array of mini page's visual element modifications

The **optional properties** of the mini page schema structure include:

- `attributes`, schema attributes
- `methods`, schema methods
- `mixins`, schema mixins
- `messages`, schema messages

The optional properties **let you**:

- add custom management elements
- register messages
- set up the mini page business logic

You can use custom styles to modify the appearance of mini page visual elements.

Attention. Mini page business logic cannot be set up using business rules.

Mini page operations

Add a mini page to a section

1. Add the view model schema of a page to a custom package.
2. Select the `BaseMiniPage` schema as a parent object.
3. Add the needed mini page functionality to the schema source code. Make sure to specify the name of the object schema to bind the mini page in the `entitySchemaName` element and perform at least one modification in the `diff` array.
4. Modify the `[SysModuleEdit]` system database table via an SQL query.
5. Add the `[HasSectionCodeMiniPageAddMode]` system setting. Learn more about adding system settings in user documentation: Manage system settings.

Attention. The execution of an erroneous SQL query may corrupt existing data and render Creatio inoperable.

Add a mini page to a module

You might need to connect a mini page to a Creatio module to solve a business problem. **Modules** let you create links to specific Creatio objects. As such, a mini page connected to a module enables users to view the object data without opening the object section.

The following Creatio modules are connected to mini pages out-of-the-box:

- telephony in the communication panel
- email in the communication panel
- notification center in the communication panel
- the [*Feed*] section in the communication panel
- list in the dashboards section

To **add a mini page to a module**:

1. Create a module schema.
2. Create a module view and view model. Enable the `Terrasoft.MiniPageUtilities` utility class in the `mixins` view model property. The class lets you use the mini page call methods.
3. Add the module styles.
4. Create a container to display the module view.

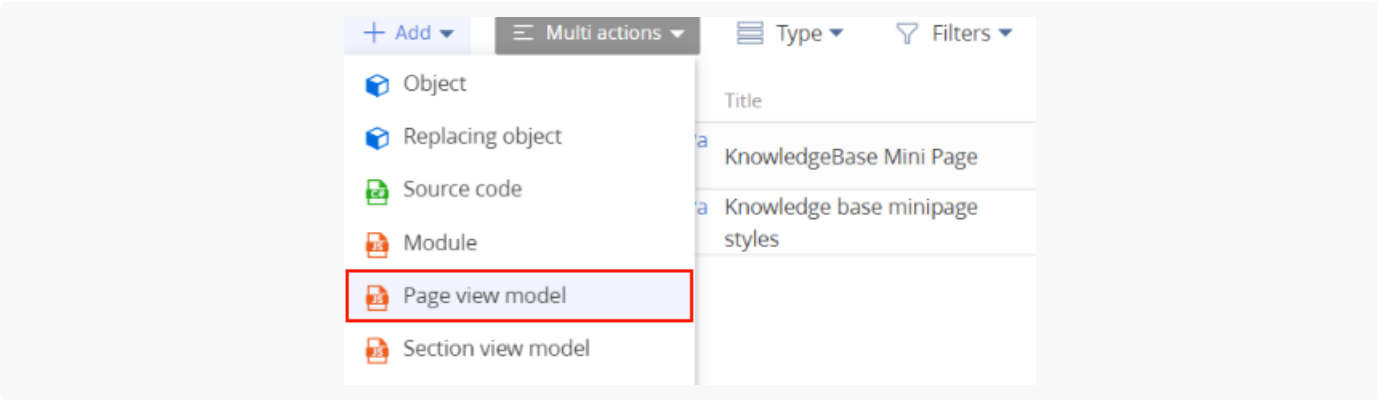
Create a custom mini page

Advanced

Example. Create a custom mini page for the [*Knowledge base*] section. The mini page must contain the base set of [*Name*] and [*Tags*] fields and let you download attachments.

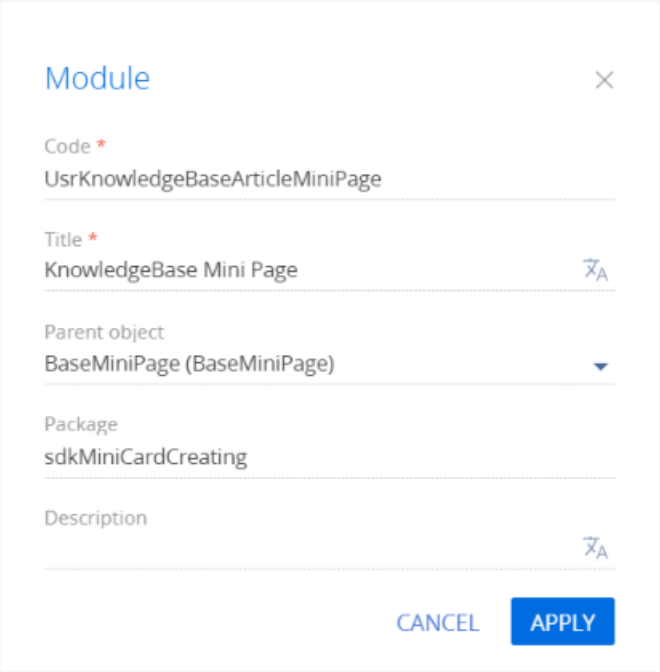
1. Create a view model schema of the mini page

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



3. Fill out the schema properties in the Schema Designer:
   - Set [*Code*] to "`UsrKnowledgeBaseArticleMiniPage`."
   - Set [*Title*] to "`KnowledgeBase Mini Page`."

- Set [*Parent object*] to "`BaseMiniPage`."



Click [*Apply*] to apply the properties.

2. Display the fields of the primary object

Add the needed source code in the Schema Designer.

1. Specify the `KnowledgeBase` schema as the object schema.
2. Add the needed modifications to the `diff` array of view model modifications.

   The **view model elements** of the base mini page are as follows:

   - `MiniPage` is the page field.
   - `HeaderContainer` is the page heading. By default, the mini page places it in the first row of the field.

   In this example, the `diff` modification array contains two new objects that configure the [*Name*] and [*Keywords*] fields.

View the source code of the view model schema below.

`UsrKnowledgeBaseArticleMiniPage.js`

```
define("UsrKnowledgeBaseArticleMiniPage", [], function() {
    return {
        entitySchemaName: "KnowledgeBase",
        attributes: {
            "MiniPageModes": {
                "value": [this.Terrasoft.ConfigurationEnums.CardOperation.VIEW]
```

```
            }
        },
        diff: /**SCHEMA_DIFF*/[
            {
                "operation": "insert",
                "name": "Name",
                "parentName": "HeaderContainer",
                "propertyName": "items",
                "index": 0,
                "values": {
                    "labelConfig": {
                        "visible": false
                    },
                    "isMiniPageModelItem": true
                }
            },
            {
                "operation": "insert",
                "name": "Keywords",
                "parentName": "MiniPage",
                "propertyName": "items",
                "values": {
                    "labelConfig": {
                        "visible": false
                    },
                    "isMiniPageModelItem": true,
                    "layout": {
                        "column": 0,
                        "row": 1,
                        "colSpan": 24
                    }
                }
            }
        ]/**SCHEMA_DIFF*/
    };
});
```

3. Add a functional button to the mini page

As per the example conditions, the mini page must let you download the files bound to the knowledge base article.

To implement the management of additional data, display the data as a drop-down list of a preconfigured button.

Modify the source code of the view model in the Schema Designer.

To **add a button** that selects the files of the knowledge base article:

1. Add the `FilesButton` element, which is the button's description, to the `diff` array.

2. Add the `Article` virtual column that connects the main and additional records to the `attributes` property.
3. Add the `MiniPageModes` attribute that represents the array of the needed mini page operations to the `attributes` property.
4. Add the button image to schema resources. For example, use the following image: . Learn more about adding images to resources in a separate article: Add a field with an image.
5. Add methods that manage the drop-down list of the file selection button to the `methods` property:

   - `init()` is an overridden base method.
   - `onEntityInitialized()` is an overridden base method.
   - `setArticleInfo()` sets the value of the `Article` attribute.
   - `getFiles(callback, scope)` retrieves the data about the files of the current knowledge base article.
   - `initFilesMenu(files)` populates the collection of the file selection button's drop-down list.
   - `fillFilesExtendedMenuData()` initiates the upload of files and the addition of them to the file selection button's drop-down list.
   - `downloadFile()` initiates the download of the selected file.

View the view model schema's source code that adds the functional button below.

UsrKnowledgeBaseArticleMiniPage.js

```
define("UsrKnowledgeBaseArticleMiniPage",
["terrasoft", "KnowledgeBaseFile", "ConfigurationConstants"],
    function(Terrasoft, KnowledgeBaseFile, ConfigurationConstants) {
        return {
            entitySchemaName: "KnowledgeBase",
            attributes: {
                "MiniPageModes": {
                    "value": [this.Terrasoft.ConfigurationEnums.CardOperation.VIEW]
                },
                "Article": {
                    "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                    "referenceSchemaName": "KnowledgeBase"
                }
            },
            methods: {
                /* Initialize the collection of the file selection button's dropdown list.*/
                init: function() {
                    this.callParent(arguments);
                    this.initExtendedMenuButtonCollections("File", ["Article"], this.close);
                },
                /* Initialize the value of the attribute that binds the main and additional recc
    Populate the collection of the file selection button's drop-down list.*/
                onEntityInitialized: function() {
                    this.callParent(arguments);
                    this.setArticleInfo();
```

```
                    this.fillFilesExtendedMenuData();
                },
                /* Initiate the upload of files and the addition of them to the file selection b
                fillFilesExtendedMenuData: function() {
                    this.getFiles(this.initFilesMenu, this);
                },
                /* Set the value of the attribute that binds the main and additional records.*/
                setArticleInfo: function() {
                    this.set("Article", {
                        value: this.get(this.primaryColumnName),
                        displayValue: this.get(this.primaryDisplayColumnName)
                    });
                },
                /* Retrieve the data about the files of the current knowledge base article.*/
                getFiles: function(callback, scope) {
                    var esq = this.Ext.create("Terrasoft.EntitySchemaQuery", {
                        rootSchema: KnowledgeBaseFile
                    });
                    esq.addColumn("Name");
                    var articleFilter = >this.Terrasoft.createColumnFilterWithParameter(
                        this.Terrasoft.ComparisonType.EQUAL, "KnowledgeBase", this.get(this.prim
                    var typeFilter = this.Terrasoft.createColumnFilterWithParameter(
                        this.Terrasoft.ComparisonType.EQUAL, "Type", ConfigurationConstants.File
                    esq.filters.addItem(articleFilter);
                    esq.filters.addItem(typeFilter);
                    esq.getEntityCollection(function(response) {
                        if (!response.success) {
                            return;
                        }
                        callback.call(scope, response.collection);
                    }, this);
                },
                /* Populate the collection of the file selection button's drop-down list.*/
                initFilesMenu: function(files) {
                    if (files.isEmpty()) {
                        return;
                    }
                    var data = [];
                    files.each(function(file) {
                        data.push({
                            caption: file.get("Name"),
                            tag: file.get("Id")
                        });
                    }, this);
                    var recipientInfo = this.fillExtendedMenuItems("File", ["Article"]);
                    this.fillExtendedMenuData(data, recipientInfo, this.downloadFile);
                },
                /* Initiate the download of the selected file.*/
```

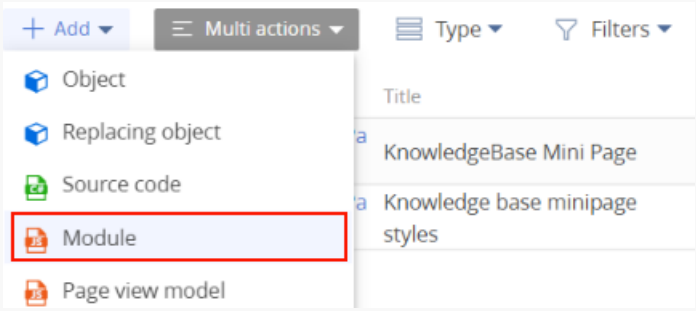
```
            downloadFile: function(id) {
                var element = document.createElement("a");
                element.href = "../rest/FileService/GetFile/" + KnowledgeBaseFile.uId + "/"
                document.body.appendChild(element);
                element.click();
                document.body.removeChild(element);
            }
        },
        diff: /**SCHEMA_DIFF*/[
            {
                "operation": "insert",
                "name": "Name",
                "parentName": "HeaderContainer",
                "propertyName": "items",
                "index": 0,
                "values": {
                    "labelConfig": {
                        "visible": true
                    },
                    "isMiniPageModelItem": true
                }
            },
            {
                "operation": "insert",
                "name": "Keywords",
                "parentName": "MiniPage",
                "propertyName": "items",
                "values": {
                    "labelConfig": {
                        "visible": true
                    },
                    "isMiniPageModelItem": true,
                    "layout": {
                        "column": 0,
                        "row": 1,
                        "colSpan": 24
                    }
                }
            },
            {
                "operation": "insert",
                "parentName": "HeaderContainer",
                "propertyName": "items",
                "name": "FilesButton",
                "values": {
                    "itemType": Terrasoft.ViewItemType.BUTTON,
                    /* Set up the button image.*/
                    "imageConfig": {
                        /* Add the image to the mini page resources before you run the code.
```

```
                        "bindTo": "Resources.Images.FilesImage"
                    },
                    /* Set up the drop-down list.*/
                    "extendedMenu": {
                        /* The name of the drop-down list element.*/
                        "Name": "File",
                        /* The name of the mini page attribute that binds the main and addit
                        "PropertyName": "Article",
                        /* Set up the button click processor.*/
                        "Click": {
                            "bindTo": "fillFilesExtendedMenuData"
                        }
                    }
                },
                "index": 1
            }
        ]/**SCHEMA_DIFF*/
    };
});
```

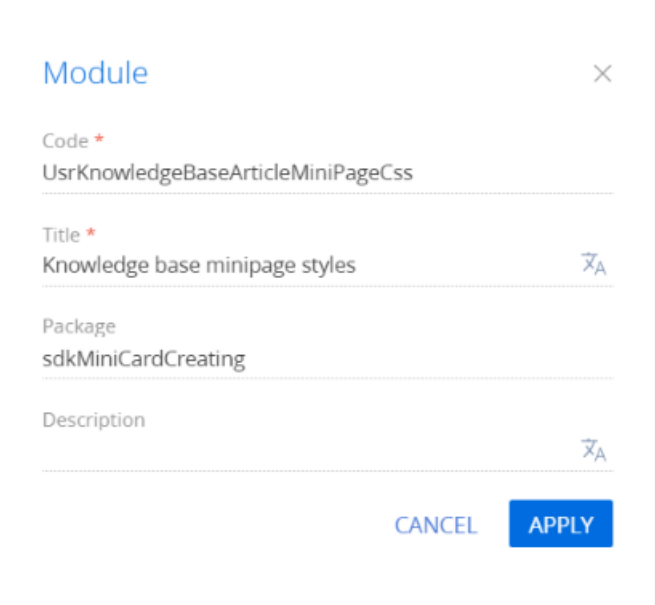
4. Apply styles to the mini page

To add styles to the view model, create an individual style module and enable it in the view model schema.

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



3. Fill out the schema properties in the Schema Designer:
   - Set [*Code*] to "`UsrKnowledgeBaseArticleMiniPageCss`."
   - Set [*Title*] to "`Knowledge base minipage styles`."



Click [*Apply*] to apply the properties.

4. Specify the needed styles in the context menu of the LESS node.

UsrKnowledgeBaseArticleMiniPageCss.js

```
div[data-item-marker="UsrKnowledgeBaseArticleMiniPageContainer"] > div {
width: 250px;
}
```

5. Add the module loading to the source code of the view model schema in the Schema Designer.

   View the full source code of the mini page below:

UsrKnowledgeBaseArticleMiniPage.js

```
define("UsrKnowledgeBaseArticleMiniPage",
["terrasoft", "KnowledgeBaseFile", "ConfigurationConstants", "css!UsrKnowledgeBaseArticleMini
    function(Terrasoft, KnowledgeBaseFile, ConfigurationConstants) {
        return {
            entitySchemaName: "KnowledgeBase",
            attributes: {
                "MiniPageModes": {
                    "value": [this.Terrasoft.ConfigurationEnums.CardOperation.VIEW]
                },
                "Article": {
                    "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                    "referenceSchemaName": "KnowledgeBase"
                }
            },
            methods: {
```

```
                init: function() {
                    this.callParent(arguments);
                    this.initExtendedMenuButtonCollections("File", ["Article"], this.close);
                },
                onEntityInitialized: function() {
                    this.callParent(arguments);
                    this.setArticleInfo();
                    this.fillFilesExtendedMenuData();
                },
                fillFilesExtendedMenuData: function() {
                    this.getFiles(this.initFilesMenu, this);
                },
                setArticleInfo: function() {
                    this.set("Article", {
                        value: this.get(this.primaryColumnName),
                        displayValue: this.get(this.primaryDisplayColumnName)
                    });
                },
                getFiles: function(callback, scope) {
                    var esq = this.Ext.create("Terrasoft.EntitySchemaQuery", {
                        rootSchema: KnowledgeBaseFile
                    });
                    esq.addColumn("Name");
                    var articleFilter = this.Terrasoft.createColumnFilterWithParameter(
                        this.Terrasoft.ComparisonType.EQUAL, "KnowledgeBase", this.get(this.p
                    var typeFilter = this.Terrasoft.createColumnFilterWithParameter(
                        this.Terrasoft.ComparisonType.EQUAL, "Type", ConfigurationConstants.F
                    esq.filters.addItem(articleFilter);
                    esq.filters.addItem(typeFilter);
                    esq.getEntityCollection(function(response) {
                        if (!response.success) {
                            return;
                        }
                        callback.call(scope, response.collection);
                    }, this);
                },
                initFilesMenu: function(files) {
                    if (files.isEmpty()) {
                        return;
                    }
                    var data = [];
                    files.each(function(file) {
                        data.push({
                            caption: file.get("Name"),
                            tag: file.get("Id")
                        });
                    }, this);
                    var recipientInfo = this.fillExtendedMenuItems("File", ["Article"]);
```

```
                this.fillExtendedMenuData(data, recipientInfo, this.downloadFile);
            },
            downloadFile: function(id) {
                var element = document.createElement("a");
                element.href = "../rest/FileService/GetFile/" + KnowledgeBaseFile.uId + "
                document.body.appendChild(element);
                element.click();
                document.body.removeChild(element);
            }
        },
        diff: /**SCHEMA_DIFF*/[
            {
                "operation": "insert",
                "name": "Name",
                "parentName": "HeaderContainer",
                "propertyName": "items",
                "index": 0,
                "values": {
                    "labelConfig": {
                        "visible": true
                    },
                    "isMiniPageModelItem": true
                }
            },
            {
                "operation": "insert",
                "name": "Keywords",
                "parentName": "MiniPage",
                "propertyName": "items",
                "values": {
                    "labelConfig": {
                        "visible": true
                    },
                    "isMiniPageModelItem": true,
                    "layout": {
                        "column": 0,
                        "row": 1,
                        "colSpan": 24
                    }
                }
            },
            {
                "operation": "insert",
                "parentName": "HeaderContainer",
                "propertyName": "items",
                "name": "FilesButton",
                "values": {
                    "itemType": Terrasoft.ViewItemType.BUTTON,
                    "imageConfig": {
```

```
                            "bindTo": "Resources.Images.FilesImage"
                        },
                        "extendedMenu": {
                            "Name": "File",
                            "PropertyName": "Article",
                            "Click": {
                                "bindTo": "fillFilesExtendedMenuData"
                            }
                        }
                    },
                    "index": 1
                }
            ]/**SCHEMA_DIFF*/
        };
    });
```

5. Register the mini page in the database

You must register new mini pages in the database. Run the following SQL query to modify the database:

Query to create a mini page

```
DECLARE
    -- The name of the mini page view schema.
    @ClientUnitSchemaName NVARCHAR(100) = 'UsrKnowledgeBaseArticleMiniPage',
    -- The name of the object schema to bind the mini page.
    @EntitySchemaName NVARCHAR(100) = 'KnowledgeBase'

UPDATE SysModuleEdit
SET MiniPageSchemaUId = (
    SELECT TOP 1 UId
    FROM SysSchema
    WHERE Name = @ClientUnitSchemaName
)
WHERE SysModuleEntityId = (
    SELECT TOP 1 Id
    FROM SysModuleEntity
    WHERE SysEntitySchemaUId = (
        SELECT TOP 1 UId
        FROM SysSchema
        WHERE Name = @EntitySchemaName
            AND ExtendParent = 0
    )
);
```

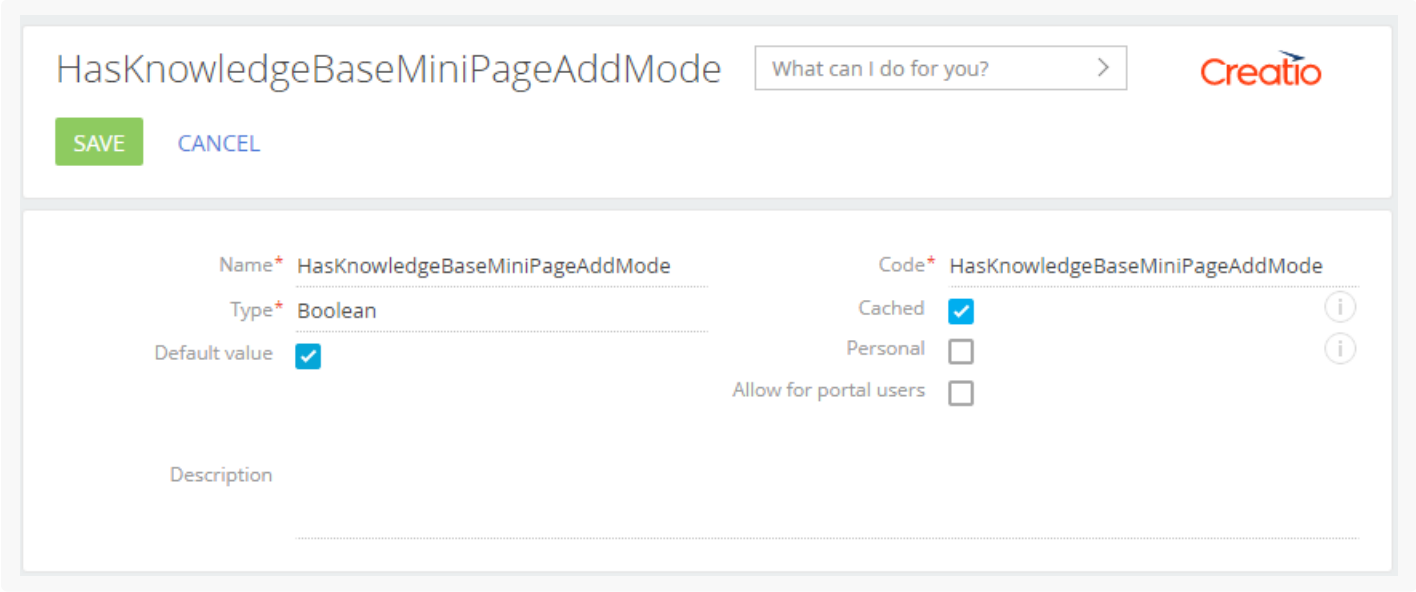
As a result of the query, Creatio will add a unique mini page identifier to the `[MiniPageSchemaUId]` field of the `[SysModuleEdit]` table record that corresponds to the [*Knowledge base*] section.

	CardSchemaUId	ActionKindCaption	ActionKindName	PageCaption	MiniPageSchemaUId
50	9DBD0611-FA52-4A90-9542-E5FD997B4AFD	New article	KnowledgeBase	Knowledge base article	9C072BC6-93C7-488A-87A6-A41B79CC67...

6. Add the system setting

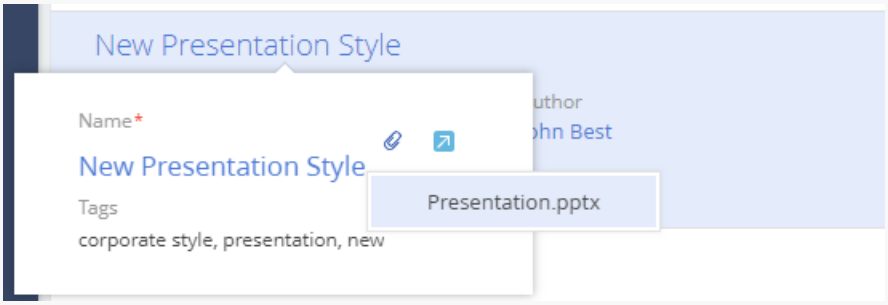
Add the system setting that has the following properties to the [*System settings*] section of the System Designer:

- Set [*Name*] to "`HasKnowledgeBaseMiniPageAddMode`."
- Set [*Code*] to "`HasKnowledgeBaseMiniPageAddMode`."
- Set [*Type*] to "`Boolean`."
- Set [*Default value*] to selected checkbox.



Outcome of the example

After you save the schema and refresh the Creatio web page, hover over the article name in the [*Knowledge base*] section to bring up a custom mini page that displays the files bound to the record. You will be able to download the files.



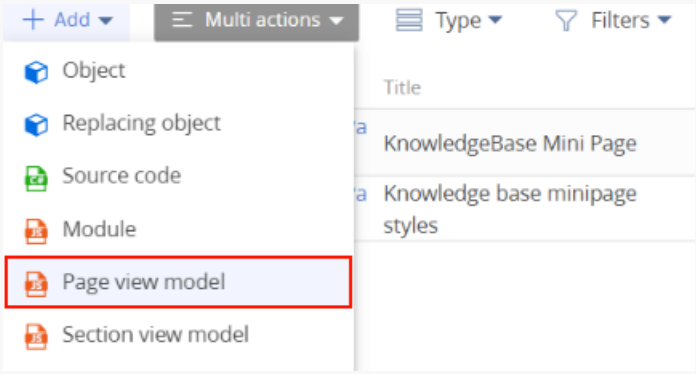
Create a mini page that adds records

Advanced

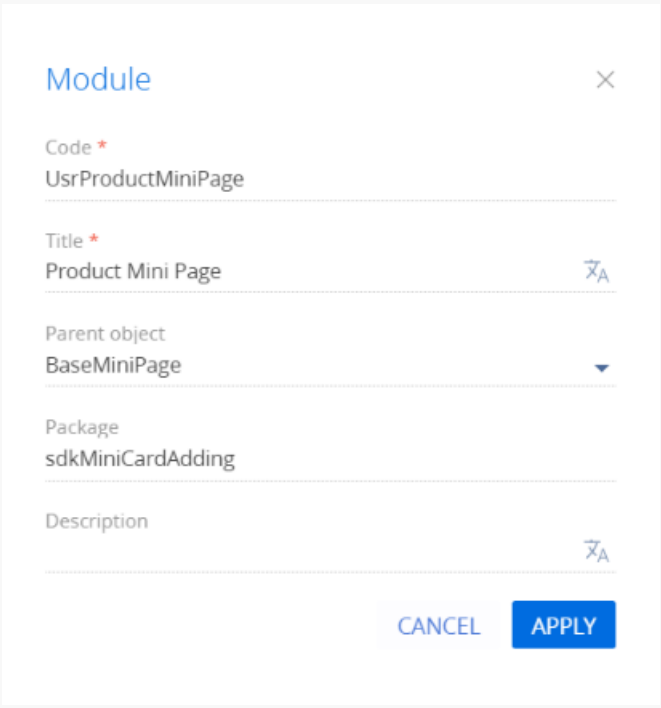
Example. Create a custom mini page that adds a new record to the [*Products*] section. The mini page must contain the base set of [*Name*] and [*Code*] fields.

1. Create a view model schema of the mini page

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



3. Fill out the schema properties in the Schema Designer:
   - Set [*Code*] to "`UsrProductMiniPage`."
   - Set [*Title*] to "`Product Mini Page`."
   - Set [*Parent object*] to "`BaseMiniPage`."



Click [*Apply*] to apply the properties.

2. Display the fields of the primary object

Add the needed source code in the Schema Designer.

1. Specify the `Product` schema as the object schema.
2. Declare the `MiniPageModes` attribute. Assign the array that has the collection of the needed mini page operations to the attribute.

> **Note.** If you also need to display the mini page on the section page (see Create a custom mini page), add `this.Terrasoft.ConfigurationEnums.CardOperation.VIEW` to the array assigned to the `MiniPageModes` attribute

3. Add the needed modifications to the `diff` array of view model modifications.

   The **view model elements** of the base mini page are as follows:

   - `MiniPage` is the page field.
   - `HeaderContainer` is the page heading. By default, the mini page places it in the first row of the field.

   In this example, the `diff` modification array contains two new objects that configure the [*Name*] and [*Code*] fields.

View the source code of the view model schema below.

`UsrProductMiniPage.js` file that displays the object fields

```
define("UsrProductMiniPage", ["UsrProductMiniPageResources"],
    function(resources) {
        return {
            entitySchemaName: "Product",
            details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
            attributes: {
                "MiniPageModes": {
                    "value": [this.Terrasoft.ConfigurationEnums.CardOperation.ADD]
                }
            },
            diff: /**SCHEMA_DIFF*/[
                {
                    "operation": "insert",
                    "parentName": "MiniPage",
                    "propertyName": "items",
                    "name": "Name",
                    "values": {
                        "isMiniPageModelItem": true,
                        "layout": {
                            "column": 0,
                            "row": 1,
                            "colSpan": 24
                        },
                        "controlConfig": {
                            "focused": true
                        }
                    }
                },
                {
                    "operation": "insert",
                    "parentName": "MiniPage",
                    "propertyName": "items",
                    "name": "Code",
                    "values": {
                        "isMiniPageModelItem": true,
                        "layout": {
                            "column": 0,
                            "row": 2,
                            "colSpan": 24
                        }
                    }
                }
            ]/**SCHEMA_DIFF*/
        };
    });
```

3. Register the mini page in the database

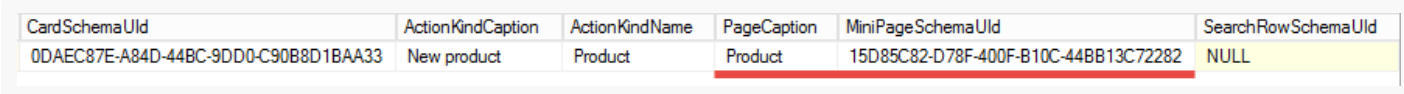
Register new mini pages in the database. Run the following SQL query to do that.

Query to create a mini page

```
DECLARE
    -- The name of the mini page view schema.
@ClientUnitSchemaName NVARCHAR(100) = 'UsrProductMiniPage',
    -- The name of the object schema to bind the mini page.
@EntitySchemaName NVARCHAR(100) = 'Product'

UPDATE SysModuleEdit
SET MiniPageSchemaUId = (
    SELECT TOP 1 UId
    FROM SysSchema
    WHERE Name = @ClientUnitSchemaName
)
WHERE SysModuleEntityId = (
    SELECT TOP 1 Id
    FROM SysModuleEntity
    WHERE SysEntitySchemaUId = (
        SELECT TOP 1 UId
        FROM SysSchema
        WHERE Name = @EntitySchemaName
            AND ExtendParent = 0
    )
);
```

As a result of the query, Creatio will add the unique mini page identifier to the `[MiniPageSchemaUId]` field of the `[SysModuleEdit]` table record that corresponds to the [*Products*] section.

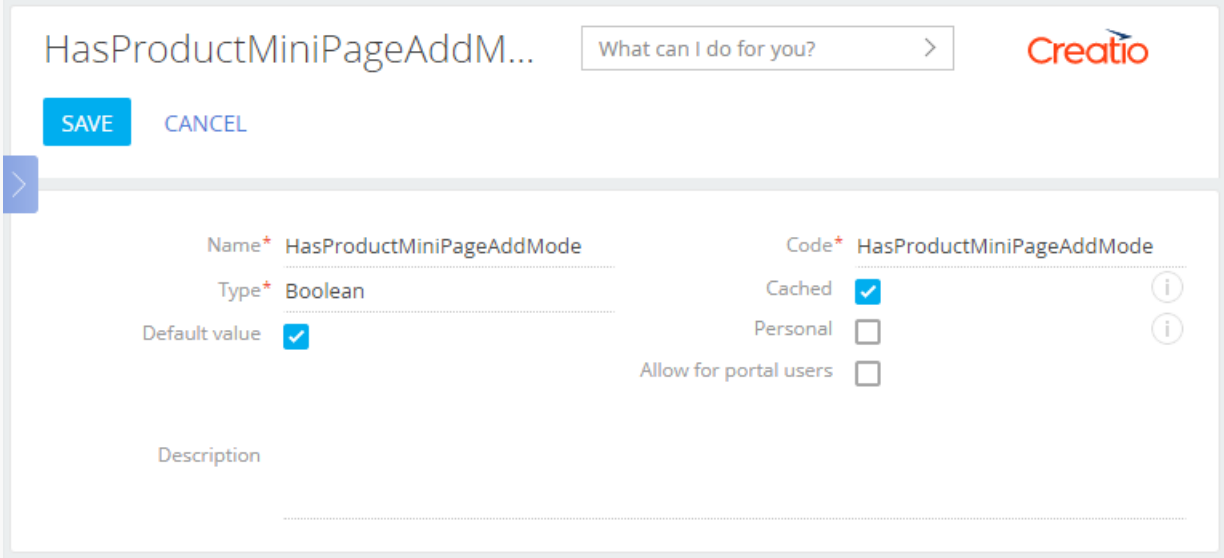


CardSchemaUId	ActionKindCaption	ActionKindName	PageCaption	MiniPageSchemaUId	SearchRowSchemaUId
0DAEC87E-A84D-44BC-9DD0-C90B8D1BAA33	New product	Product	Product	15D85C82-D78F-400F-B10C-44BB13C72282	NULL

4. Add the system setting

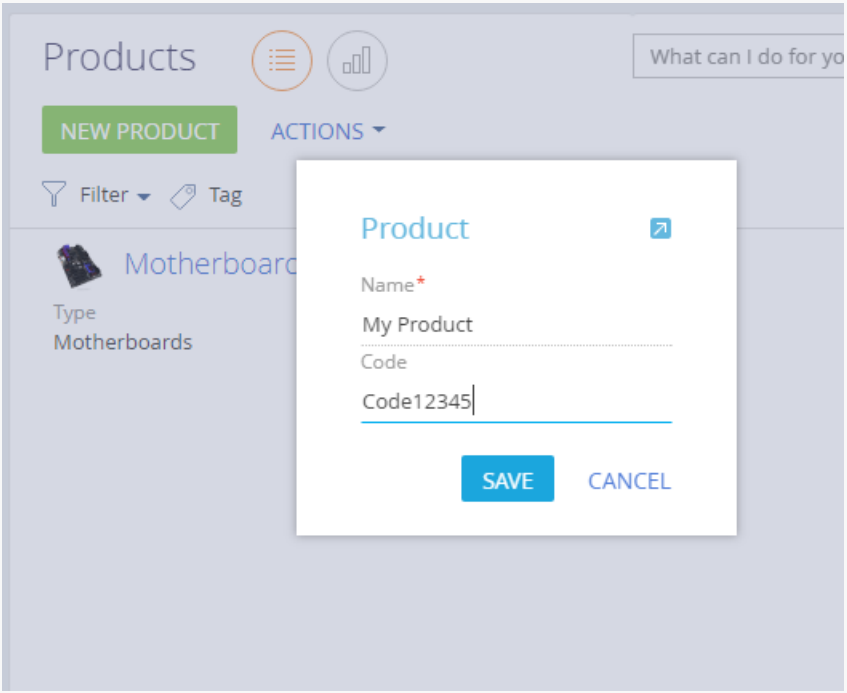
Add the system setting that has the following properties to the [*System settings*] section of the System Designer:

- Set [*Name*] to "`HasProductMiniPageAddMode`."
- Set [*Code*] to "`HasProductMiniPageAddMode`."
- Set [*Type*] to "`Boolean`."
- Set [*Default value*] to selected checkbox.

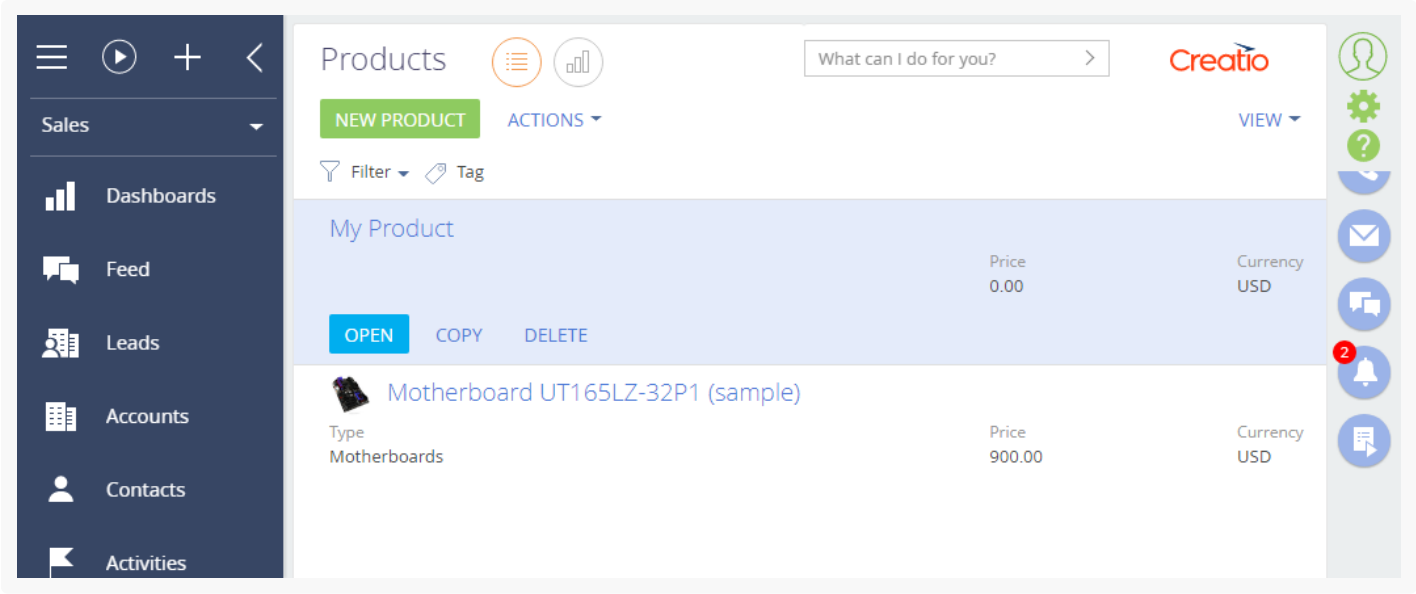


Outcome of the example

As a result, Creatio will display the mini page with two fields when you add a new product.



After you save the mini page, the corresponding record will appear in the section list.



Attention. Creatio will display the record in the section list only after you refresh the browser page. If you want Creatio to display the record immediately after you save the mini page, add the corresponding functionality to the mini page and section page schemas via the message mechanism. Learn more in a separate article: Module message exchange.

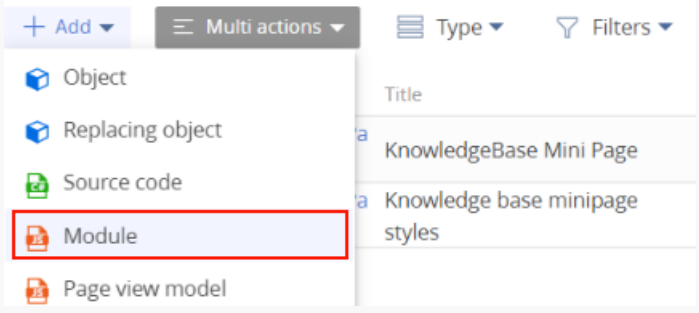
Add a mini page to a module

Advanced

Example. Display the current user in the top right of Creatio next to the profile icon. Hover over the link to the current user to open the mini page.

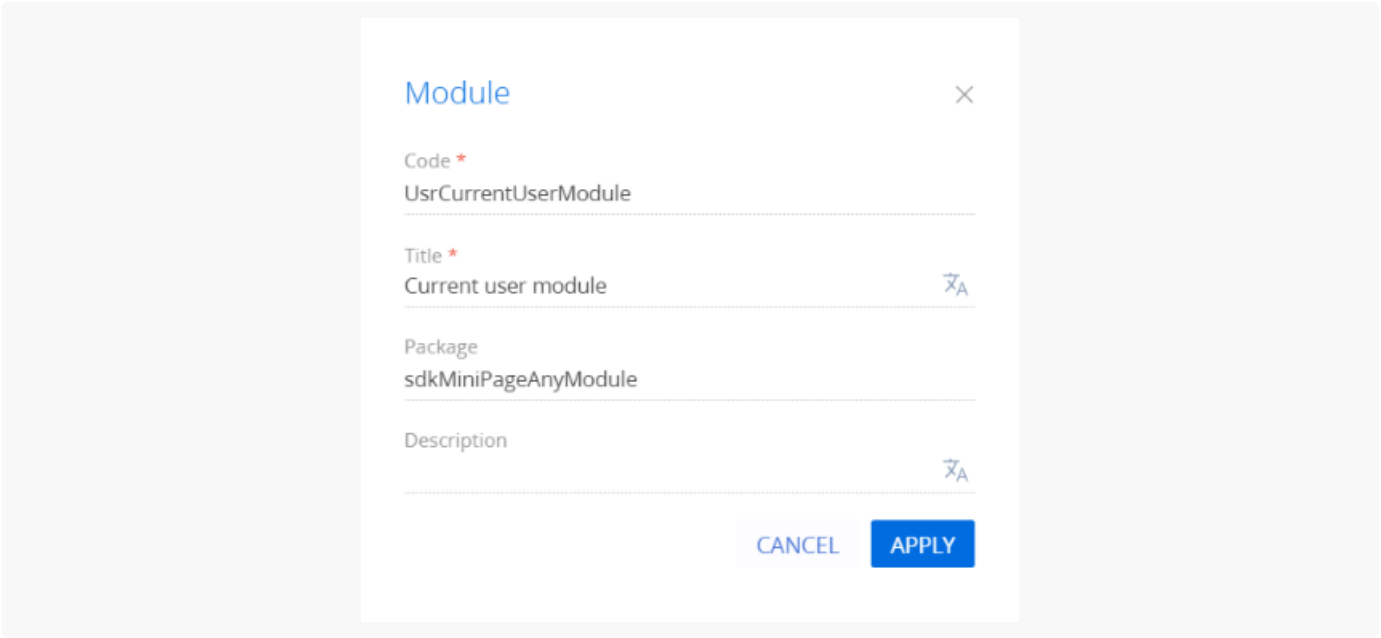
1. Create a module schema

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



3. Fill out the schema properties in the Schema Designer:

- Set [*Code*] to "`UsrCurrentUserModule`."
- Set [*Title*] to "`Current user module`."



Click [*Apply*] to apply the properties.

2. Create a view and view model of the module

Add the source code to the `UsrCurrentUserModule` module in the Schema Designer.

1. Implement the class inherited from `Terrasoft.BaseViewModel` to create a view model.
2. Enable the `Terrasoft.MiniPageUtilities` utility class in the `mixins` property of the view model. The class lets you use the mini page call methods.
3. Implement the class inherited from `Terrasoft.BaseModule` to create a view.
4. Override the following methods of the `Terrasoft.BaseModule` class in the new class:
   - `init()` initializes the view model of the module.
   - `render()` binds the view model to the container view display passed in the `renderTo` parameter.
   - `getViewModel()` creates the view model.
   - `getView()` retrieves the view to display it. The view must contain the full name of the current user and a hyperlink to the contact page. Define the hover event handler when creating the hyperlink.
5. Define the `viewModel` property that stores the link to the retrieved view model.

View the source code of the module below.

`UsrCurrentUserModule.js`

```
/* Define the module. */
```

```
define("UsrCurrentUserModule", ["MiniPageUtilities"], function() {
    /* Define the CurrentUserViewModel class. */
    Ext.define("Terrasoft.configuration.CurrentUserViewModel", {
        /* The parent class name. */
        extend: "Terrasoft.BaseViewModel",
        /* The shortened class name. */
        alternateClassName: "Terrasoft.CurrentUserViewModel",
        /* The mixins used. */
        mixins: {
            MiniPageUtilitiesMixin: "Terrasoft.MiniPageUtilities"
        }
    });
    /* Define the UsrCurrentUserModule class. */
    Ext.define("Terrasoft.configuration.UsrCurrentUserModule", {
        /* The shortened class name. */
        alternateClassName: "Terrasoft.UsrCurrentUserModule",
        /* The parent class name. */
        extend: "Terrasoft.BaseModule",
        /* The Ext object. */
        Ext: null,
        /* The sandbox object. */
        sandbox: null,
        /* The Terrasoft object. */
        Terrasoft: null,
        /* The view model. */
        viewModel: null,
        /* Create the module views. */
        getView: function() {
            /* Retrieve the current user contact. */
            var currentUser = Terrasoft.SysValue.CURRENT_USER_CONTACT;
            /* The view that represents the Terrasoft.Hyperlink class instance. */
            return Ext.create("Terrasoft.Hyperlink", {
                /* Populate the link anchor text using the contact name. */
                "caption": currentUser.displayValue,
                /* The link hover event handler. */
                "linkMouseOver": {"bindTo": "linkMouseOver"},
                /* The property that contains the additional object parameters. */
                "tag": {
                    /* The ID of the current user. */
                    "recordId": currentUser.value,
                    /* The object schema name. */
                    "referenceSchemaName": "Contact"
                }
            });
        },
        /* Create the view model of the module. */
        getViewModel: function() {
            return Ext.create("Terrasoft.CurrentUserViewModel");
```

```
        },
        /* Initialize the module. */
        init: function() {
            this.viewModel = this.getViewModel();
        },
        /* Display the module view. */
        render: function(renderTo) {
            /* Retrieve the view object. */
            var view = this.getView();
            /* Bind the view to the view model. */
            view.bind(this.viewModel);
            /* Display the view in the renderTo element. */
            view.render(renderTo);
        }
    });
    return Terrasoft.UsrCurrentUserModule;
});
```

3. Add the module styles

Add styles to the module to customize the appearance of the hyperlink.

To **add the module styles**:

1. Select the [*LESS*] node in the Schema Designer.
2. Add the following source code.

Module styles

```
.current-user-class a {
    font-weight: bold;
    font-size: 2.0em;
    margin: 6px 20px;
}

.current-user-class a:hover {
    text-decoration: none;
}
```

Save the module.

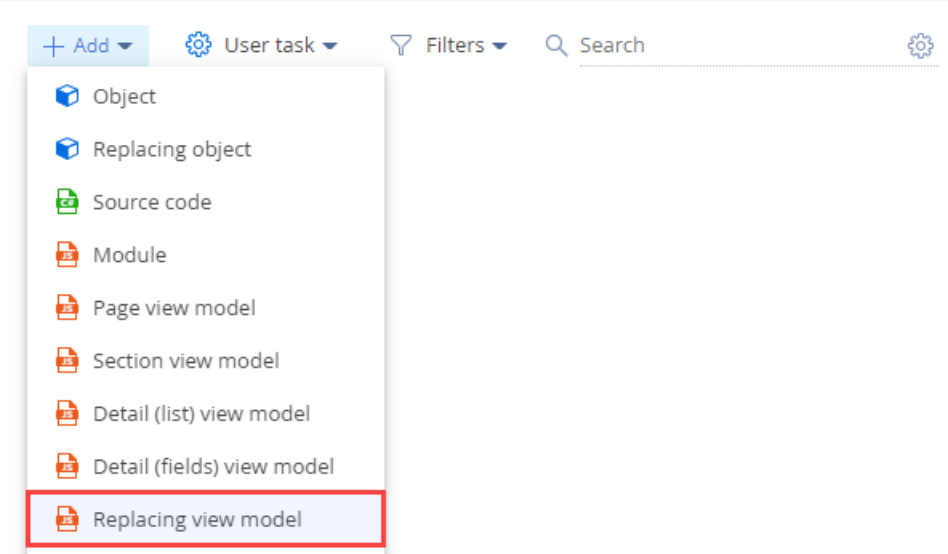
4. Create the view display container

To display a link to the user profile in the top right of Creatio, place the container and upload the view of the module to the container.

To do this, create the schema of the replacing view model that expands the functionality of the `MainHeaderSchema`

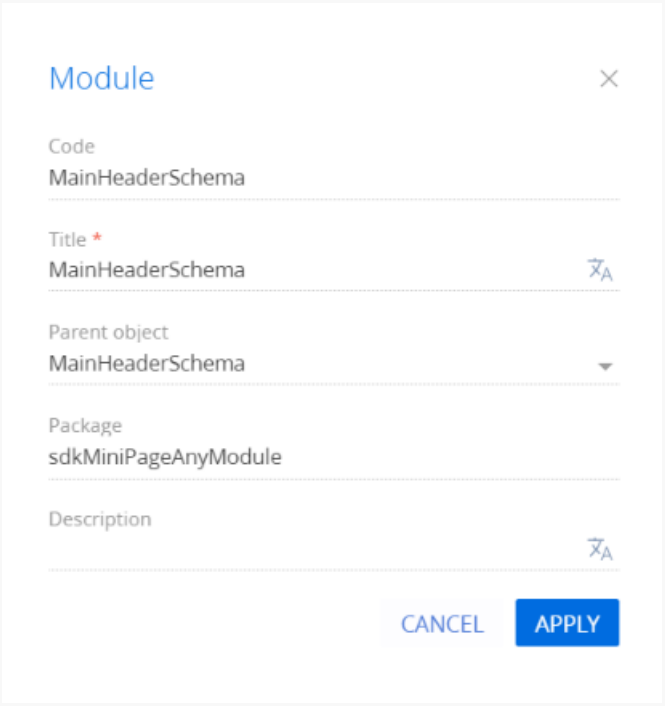
schema.

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. Select the `MainHeaderSchema` parent object in the Module Designer.

   After you confirm the parent object, Creatio will populate the other properties.



   Click [*Apply*] to apply the properties.

4. Add the source code in the Module Designer.

   Use the `diff` property to display the view in the source code of the replacing view model schema. Set the

`RightHeaderContainer` element as a parent element of the container to display the container in the top right of the page. Then, override the `onRender()` method and implement the upload of the created module in the method.

View the source code of the replacing view model schema below.

MainHeaderSchema.js

```
/* Define the module. */
define("MainHeaderSchema", [], function() {
    return {
        methods: {
            /* Execute the actions after displaying the view. */
            onRender: function() {
                /* Call the parent method. */
                this.callParent(arguments);
                /* Load the module of the current user. */
                this.loadCurrentUserModule();
            },
            /* Load the module of the current user. */
            loadCurrentUserModule: function() {
                /* Retrieve the container to upload the module. */
                var currentUserContainer = this.Ext.getCmp("current-user-container");
                /* Check if the container exists. */
                if (currentUserContainer && currentUserContainer.rendered) {
                    /* Upload the module to the container. */
                    this.sandbox.loadModule("UsrCurrentUserModule", {
                        /* The container name. */
                        renderTo: "current-user-container"
                    });
                }
            }
        },
        diff: [
            {
                /* The element insert operation. */
                "operation": "insert",
                /* The element name. */
                "name": "CurrentUserContainer",
                /* The parent container name. */
                "parentName": "RightHeaderContainer",
                /* The property name. */
                "propertyName": "items",
                /* The element values. */
                "values": {
                    /* The container ID. */
                    "id": "current-user-container",
                    /* The element type. */
```

```
                    "itemType": Terrasoft.ViewItemType.CONTAINER,
                    /* The container classes. */
                    "wrapClass": ["current-user-class"],
                    /* The container items. */
                    "items": []
                }
            }
        ]
    };
});
```

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

Outcome of the example

After you refresh the web page, Creatio will display the full name of the current user and a hyperlink to their contact page. Hover over the link to bring up the mini page that contains the data of the current user.