Creatio Academy

Front-end development

Controls

Version 8.0

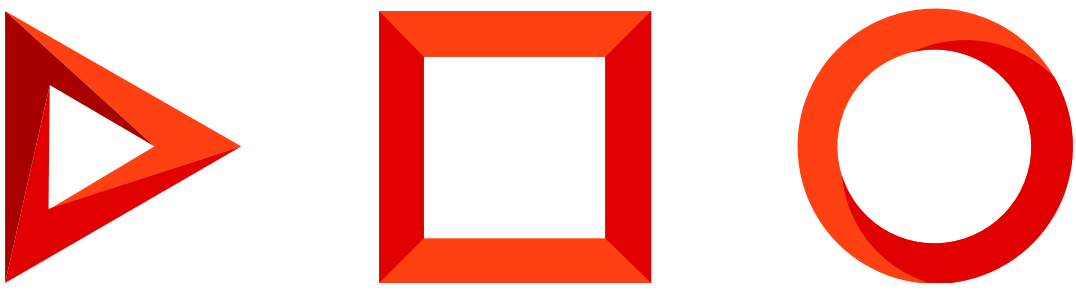


Table of Contents

Controls

Medium

Controls are objects that organize interaction between the user and Creatio. For example, buttons, details, fields, etc. Usually, navigation bars, dialog boxes, and toolbars display controls.

`Terrasoft.controls.Component` is the parent class for controls. `Terrasoft.BaseObject` class is the parent class for the `Component` class. The `Bindable` mixin of the `Component` class lets you bind control properties to the needed view model properties, methods, or attributes.

Declare events in the control to ensure it operates as intended. Control inherits the following **events** from the `Component` class:

- `added`. Triggered after the control is added to the container.
- `afterrender`. Triggered after the control is added and the HTML view is added to the DOM.
- `afterrerender`. Triggered after the control is rendered and the HTML view is updated in the DOM.
- `beforererender`. Triggered before the control is rendered and the HTML view is added to the DOM.
- `destroy`. Triggered before the control is deleted.
- `destroyed` Triggered after the control is deleted.
- `init`. Triggers after the control is initialized.

Learn more about the `Component` class events in the JS classes reference.

Control can subscribe to browser events and define its own events.

Implement the control in the `diff` array of modifications.

Array of modifications (`diff`)

```
/* diff array of modifications. */
diff: [{
    /* Run the operation that inserts the control into the page. */
    "operation": “insert”,
    /* The name of the parent element to insert the control. */
    "parentName": "CardContentContainer",
    /* The property of the parent element with which to execute the operation. */
    "propertyName": "items",
    /* The meta name of the button to add. */
    "name": "ExampleButton",
    /* Control values.
    Properties to pass to the element constructor. */
    "values": {
        /* Set the type of the added element to button. */
        "itemType": "Terrasoft.ViewItemType.BUTTON",
        /* The button title. */
```

```
        "caption": "ExampleButton",
        /* Bind the handler method of the button click. */
        "click": {"bindTo": "onExampleButtonClick"},
        /* The display style of the button. */
        "style": Terrasoft.controls.ButtonEnums.style.GREEN
    }
}]
```

The template (`template <tpl>`) defines the control appearance. Creatio generates the control view in the page view while rendering the control to the page view. Generation is based on the specified template.

The control element has no business logic. The module where you add the control implements the business logic.

The control has `styles` and `selectors` attributes that are defined in the `Component` parent class. These attributes let you customize styles flexibly.

Create custom control

Advanced

Case description

Create control which enables to enter only integer values in the specified range. Perform the checking of the entered value by pressing the Enter key and display the corresponding message if the number is outside the range. Use the `Terrasoft.controls.IntegerEdit` control as parent.

Source code

You can download the package with case implementation using the following link.

Case implementation algorithm

1. Create a client module

The procedure for creating a custom schema is covered in the “Create a client schema”.

Run the [*Add*] – [*Module*] menu command on the [*Schemas*] tab of the [*Configuration*] section.

Specify following properties of the schema:

- [Name] – “UsrLimitedIntegerEdit”
- [Title] – "UsrLimitedIntegerEdit"

Add the following source code to the schema:

```
// Declaration of the module.
define("UsrLimitedIntegerEdit", [], function () {
});
```

2. Create a class of the control

Modify the source code according to the example below.

```
define("UsrLimitedIntegerEdit", [], function () {
    //  Declaration of the class of the control.
    Ext.define("Terrasoft.controls.UsrLimitedIntegerEdit", {
        // Base class.
        extend: "Terrasoft.controls.IntegerEdit",
        // Alias (abbreviated name of the class)..
        alternateClassName: "Terrasoft.UsrLimitedIntegerEdit",
        // The smallest allowed value.
        minLimit: -1000,
        // The highest value allowed.
        maxLimit: 1000,
        // A method for checking for an occurrence in the range of valid values.
        isOutOfLimits: function (numericValue) {
            if (numericValue < this.minLimit || numericValue > this.maxLimit) {
                return true;
            }
            return false;
        },
        // Override the method of the event handler for pressing the Enter key.
        onEnterKeyPressed: function () {
            // Call the basic functionality.
            this.callParent(arguments);
            // Get the entered value.
            var value = this.getTypedValue();
            // Reduction to a numberic type.
            var numericValue = this.parseNumber(value);
            // Check for occurrence in the range of acceptable values.
            var outOfLimits = this.isOutOfLimits(numericValue);
            if (outOfLimits) {
                // Form the warning message.
                var msg = "Value " + numericValue + " is out of limits [" + this.minLimit + ", "
                // Modify the configuration object to display a warning message.
                this.validationInfo.isValid = false;
                this.validationInfo.invalidMessage = msg;
            }
            else{
                // Modify the configuration object to hide the warning message.
                this.validationInfo.isValid = true;
                this.validationInfo.invalidMessage ="";
            }
            // Call the logic for displaying the warning message.
```

```
                this.setMarkOut();
            },
        });
});
```

Note.

You can use the logic of the `onEnterKeyPressed()` method in the in the `onBlur()` event handler.

Save the schema.

Ecxept the `extend` and `alternateClassName` standard properties, the `minLimit` and `maxLimit` properties that specify the range of allowed values are added to the class. Default values are used for these properties.

The required control logic is implemented in the `onEnterKeyPressed` override method After calling the base logic in which the generation of the value change events is performed, the entered value is checked for validity. If the number is not valid, the corresponding warning message is displayed in the input field. The `isOutOfLimits` method is provided to check the occurrence of the entered value in the range of allowed values.

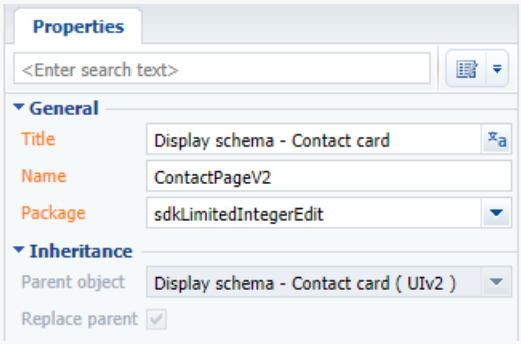
Attention.

With this implementation, despite the output of the corresponding warning, the entered value is still stored and transferred to the schema view model in which the component will be used.

3. Add the control to the Creatio interface

To add the created control to the Creatio, create the replacing schema, for example, the contact record page. Create a replacing client module and specify the [*Display schema – Contact card*] (`ContactPageV2`) schema as parent object (Fig. 1). Creating a replacing page is covered in the "Create a client schema" article.

Fig. 1. Properties of the replacing edit page



Add the following source code to the schema:

```
// Declaration of the module. Be sure to specify the dependency
// of the module in which the class of the control is declared.
define("ContactPageV2", ["UsrLimitedIntegerEdit"],
    function () {
        return {
            attributes: {
                // Attribute associated with the value in the control.
                "ScoresAttribute": {
                    // Attribute data type is integer.
                    "dataValueType": this.Terrasoft.DataValueType.INTEGER,
                    // Attribute type is a virtual column.
                    "type": this.Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                    // The default value.
                    "value": 0
                }
            },
            diff: /**SCHEMA_DIFF*/[
                {
                    // The type of operation is the addition.
                    "operation": "insert",
                    // The name of the container to which the control is added.
                    "parentName": "ProfileContainer",
                    // The name of the property in the container to which you want to add
                    // instance of the control.
                    "propertyName": "items",
                    // The name of the control.
                    "name": "Scores",
                    // Header.
                    "caption": "Scores",
                    // Values   passed to the properties of the control.
                    "values": {
                        // The type of the control is the component.
                        "itemType": Terrasoft.ViewItemType.COMPONENT,
                        // The name of the class.
                        "className": "Terrasoft.UsrLimitedIntegerEdit",
                        // The value property of the component is associated with the ScoresAttr
                        "value": { "bindTo": "ScoresAttribute" },
                        // Values   for the minLimit property.
                        "minLimit": -300,
                        // Values   for the maxLimit property.
                        "maxLimit": 300,
                        // The location of the component in the container.
                        "layout": {
                            "column": 0,
                            "row": 6,
                            "colSpan": 24,
                            "rowSpan": 1
                        }
```

```
                    }
                }
            ]/**SCHEMA_DIFF*/
        };
    });
```

Save the schema.

The added `ScoresAttribute` attribute contains the value connected to the value entered in input field of the control. You can use an integer column of the object connected to the view model of the record edit page instead of the attribute.

The configuration object determining the values of the properties of control entity is added to the `diff` array. The value of the “value” property is connected to the `ScoresAttribute` attribute. The values that specify a valid input range are assigned to the `minLimit` and `maxLimit` properties.

Attention.

If the `minLimit` and `maxLimit` properties are not explicitly specified in the configuration object, the default range (-1000, 1000) will be applied.

As a result, the integer field will be added to the contact record page (Fig. 2). The warning message will be displayed in the field if the invalid message will be entered (Fig. 3).

Fig. 2. Case result

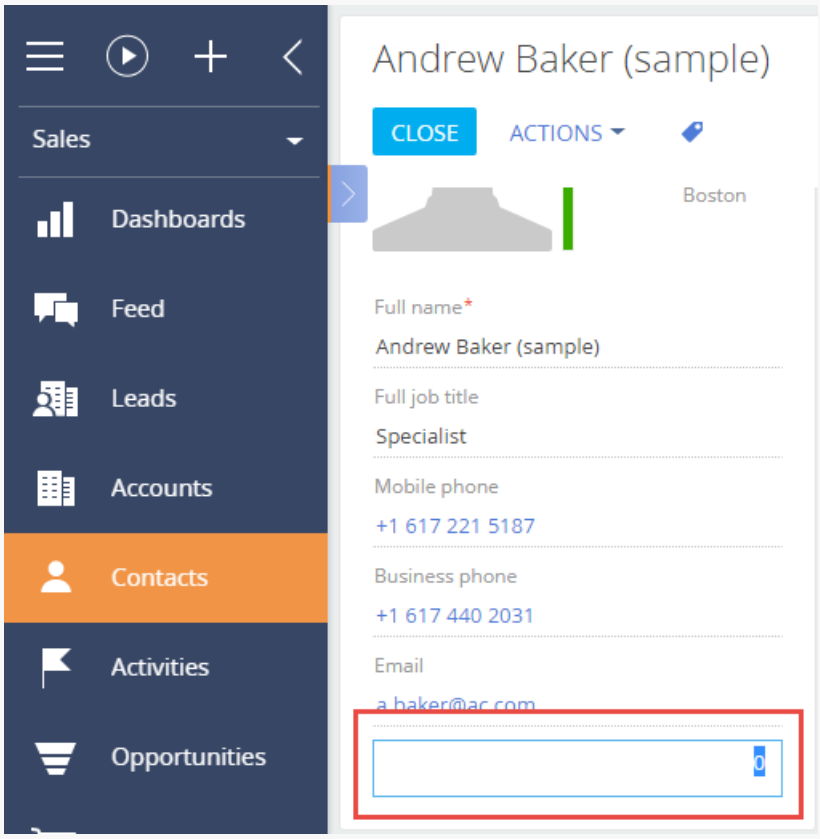


Fig. 3. Displaying of warning message