Creatio Academy

Mobile application customization

Business rules in mobile application

Version 7.18

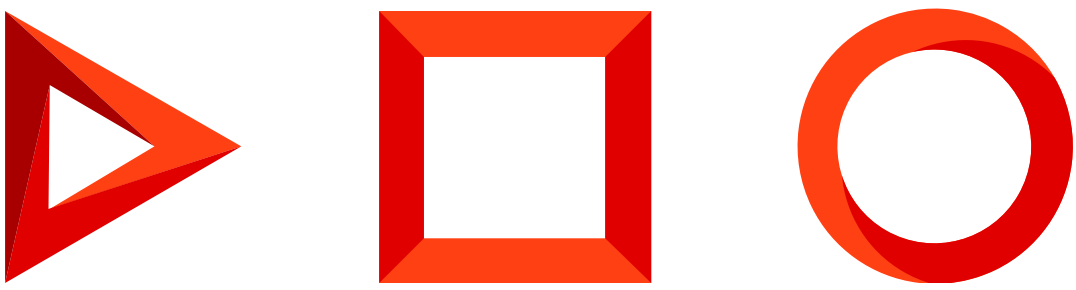

Table of Contents

Business rules in mobile application

Advanced

Business rules represent a Creatio mechanism that enables setting up the behavior of record edit page fields. You can use business rules to, e.g., set up visible or required fields, make fields enabled, etc.

Attention. Business rules work only on record edit and view pages.

Adding business rules to a page is performed via the `Terrasoft.sdk.Model.addBusinessRule(name, config)` method, where

- `name` – is the name of the model, bound to the edit page, e.g., “`Contact`”.
- `config` – is the object defining business rule properties. The list of properties depends on a specific business rule type.

In the mobile application you can add business rule that implements custom logic (custom business rule). The `Terrasoft.RuleTypes.Custom` method is provided for this type of business rules.

Filter values

Advanced

Example 1

Example. Filter values in a column by condition.

When selecting a value in the [*Product*] lookup column, only the products containing the `true` value in the [*Active*] column of the [*Product in invoice*] detail are available.

Example implementation

Filtration example

```
Terrasoft.sdk.Model.addBusinessRule("InvoiceProduct", {
    ruleType: Terrasoft.RuleTypes.Filtration,
    events: [Terrasoft.BusinessRuleEvents.Load],
    triggeredByColumns: ["Product"],
    filters: Ext.create("Terrasoft.Filter", {
        modelName: "Product",
        property: "Active",
```

```
        value: true
    })
});
```

Example 2

Example. Filter values in a column by the value in another column.

The [*Contact*] field on the record edit page of the [*Invoices*] section should be filtered based on the [*Account*] field value.

Example implementation

Filtration example

```
Terrasoft.sdk.Model.addBusinessRule("Invoice", {
    ruleType: Terrasoft.RuleTypes.Filtration,
    events: [Terrasoft.BusinessRuleEvents.Load, Terrasoft.BusinessRuleEvents.ValueChanged],
    triggeredByColumns: ["Account"],
    filteredColumn: "Contact",
    filters: Ext.create("Terrasoft.Filter", {
        property: "Account"
    })
});
```

Example 3

Example. Add and delete filtration by custom logic.

Example implementation

Filtration example

```
Terrasoft.sdk.Model.addBusinessRule("Activity", {
    name: "ActivityResultByAllowedResultFilterRule",
    position: 1,
    ruleType: Terrasoft.RuleTypes.Custom,
    triggeredByColumns: ["Result"],
    events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Load],
    executeFn: function(record, rule, column, customData, callbackConfig) {
```

```
        var allowedResult = record.get("AllowedResult");
        var filterName = "ActivityResultByAllowedResultFilter";
        if (!Ext.isEmpty(allowedResult)) {
            var allowedResultIds =  Ext.JSON.decode(allowedResult, true);
            var resultIdsAreCorrect = true;
            for (var i = 0, ln = allowedResultIds.length; i < ln; i++) {
                var item = allowedResultIds[i];
                if (!Terrasoft.util.isGuid(item)) {
                    resultIdsAreCorrect = false;
                    break;
                }
            }
            if (resultIdsAreCorrect) {
                var filter = Ext.create("Terrasoft.Filter", {
                    name: filterName,
                    property: "Id",
                    funcType: Terrasoft.FilterFunctions.In,
                    funcArgs: allowedResultIds
                });
                record.changeProperty("Result", {
                    addFilter: filter
                });
            } else {
                record.changeProperty("Result", {
                    removeFilter: filterName
                });
            }
        } else {
            record.changeProperty("Result", {
                removeFilter: filterName
            });
        }
        Ext.callback(callbackConfig.success, callbackConfig.scope, [true]);
    }
});
```

Select a field by condition

Advanced

Example. Highlight the field with the result of the activity, if its status is “`Completed`”, the [*Result*] field is not filled and the [*ProcessElementId*] column has a value.

Example implementation

Highlight the field by condition

```
// Rule for the activity edit page.
Terrasoft.sdk.Model.addBusinessRule("Activity", {
    // The name of the business rule.
    name: "ActivityResultRequiredByStatusFinishedAndProcessElementId",
    // Business rule type: custom.
    ruleType: Terrasoft.RuleTypes.Custom,
    //The rule is initiated by the Status and Result columns.
    triggeredByColumns: ["Status", "Result"],
    // The rule will work before saving the data and after changing the data.
    events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Save],
    // Handler function.
    executeFn: function(record, rule, column, customData, callbackConfig) {
        // A flag of the validity of the property and the rule.
        var isValid = true;
        // The value of the ProcessElementId column.
        var processElementId = record.get("ProcessElementId");
        // If the value is not empty.
        if (processElementId && processElementId !== Terrasoft.GUID_EMPTY) {
            // Set the validity flag.
            isValid = !(record.get("Status.Id") === Terrasoft.Configuration.ActivityStatus.Finis
                Ext.isEmpty(record.get("Result")));
        }
        // Change the properties of the Result column.
        record.changeProperty("Result", {
            // Set the column correctness indicator.
            isValid: {
                value: isValid,
                message: Terrasoft.LS["Sys.RequirementRule.message"]
            }
        });
        // Asynchronous return of values.
        Ext.callback(callbackConfig.success, callbackConfig.scope, [isValid]);
    }
});
```

Reset negative values to 0

Advanced

Example. Implement the logic for dropping negative values to 0.

Example implementation

Reset negative values to 0

```
Terrasoft.sdk.Model.addBusinessRule("Opportunity", {
    name: "OpportunityAmountValidatorRule",
    ruleType: Terrasoft.RuleTypes.Custom,
    triggeredByColumns: ["Amount"],
    events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Save],
    executeFn: function(model, rule, column, customData, callbackConfig) {
        var revenue = model.get("Amount");
        if ((revenue < 0) || Ext.isEmpty(revenue)) {
            model.set("Amount", 0, true);
        }
        Ext.callback(callbackConfig.success, callbackConfig.scope);
    }
});
```

Generate the title of an activity

Advanced

Example. Implement generating the activity header for the FieldForce solution.

Example implementation

Generating the activity header

```
Terrasoft.sdk.Model.addBusinessRule("Activity", {
    name: "FieldForceActivityTitleRule",
    ruleType: Terrasoft.RuleTypes.Custom,
    triggeredByColumns: ["Account", "Type"],
    events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Load],
    executeFn: function(record, rule, column, customData, callbackConfig, event) {
        if (event === Terrasoft.BusinessRuleEvents.ValueChanged || record.phantom) {
            var type = record.get("Type");
            var typeId = type ? type.get("Id") : null;
            if (typeId !== Terrasoft.Configuration.ActivityTypes.Visit) {
                Ext.callback(callbackConfig.success, callbackConfig.scope, [true]);
                return;
            }
            var account = record.get("Account");
            var accountName = (account) ? account.getPrimaryDisplayColumnValue() : "";
            var title = Ext.String.format("{0}: {1}", Terrasoft.LocalizableStrings.FieldForceTit
            record.set("Title", title, true);
```

```
        }
        Ext.callback(callbackConfig.success, callbackConfig.scope, [true]);
    }
});
```

config object property C#

Advanced

The base business rule

The base business rule is an abstract class, i.e., all business rules should be its inheritors.

The properties of the `config` configuration object that can be used by the inheritors of the business rule.

Configuration object properties

`ruleType`

The type of rule. The value must be included into the `Terrasoft.RuleTypes` enumeration.

`triggeredByColumns`

The column array that triggers the rule.

`message`

A text message displayed under the control element connected with the column in case business rule is not executed. It is necessary for rules that inform a user of warnings.

`name`

A unique name of a business rule. It is necessary if you need to delete a rule by the `Terrasoft.sdk` methods.

`position`

A position of a business rule that defines its order priority in the current queue.<

`events`

An event array, defining the time of running business rules. It should contain values included into the `Terrasoft.BusinessRuleEvents` enumeration.<

Possible values (`Terrasoft.BusinessRuleEvents`)

`Save`	the rule is executed before saving the data
`ValueChanged`	the rule is executed when the data is modified (while editing)
`Load`	the rule is executed when the edit page is opened

The [*Is required*] business rule (`Terrasoft.RuleTypes.Requirement`) C#

Defines whether an edit page field is required.

Configuration object properties

`ruleType`

Should contain the `Terrasoft.RuleTypes.Requirement` value for this rule.

`requireType`

Verification type. The value must be included into the `Terrasoft.RequirementTypes` enumeration. The rule can verify one or all the columns from `triggeredByColumns`.

`triggeredByColumns`

The column array that triggers the rule. If the verification type equals `Terrasoft.RequirementTypes.Simple`, one column in the array should be specified.

Possible values (`Terrasoft.RequirementTypes`)

`Simple`	value verification in one column
`OneOf`	one of the columns specified in the `triggeredByColumns` should be populated

Use case

```
Terrasoft.sdk.Model.addBusinessRule("Contact", {
    ruleType: Terrasoft.RuleTypes.Requirement,
    requireType : Terrasoft.RequirementTypes.OneOf,
    events: [Terrasoft.BusinessRuleEvents.Save],
    triggeredByColumns: ["HomeNumber", "BusinessNumber"],
    columnNames: ["HomeNumber", "BusinessNumber"]
});
```

The [*Visibility*] business rule (`Terrasoft.RuleTypes.Visibility`) C#

You can hide and display fields per condition using this rule.

Configuration object properties

`ruleType`

Should contain the `Terrasoft.RuleTypes.Visibility` value for this rule.

`triggeredByColumns`

The column array that triggers the rule.

`events`

An event array, defining the time of running business rules. It should contain values included into the `Terrasoft.BusinessRuleEvents` enumeration.

`conditionalColumns`

Condition array of business rule execution. Usually, these are specific column values.

`dependentColumnNames`

Column name array that the business rule is applied to.

Use case

```
Terrasoft.sdk.Model.addBusinessRule("Account", {
    ruleType: Terrasoft.RuleTypes.Visibility,
    conditionalColumns: [
        {name: "Type", value: Terrasoft.Configuration.Consts.AccountTypePharmacy}
    ],
    triggeredByColumns: ["Type"],
    dependentColumnNames: ["IsRx", "IsOTC"]
});
```

The fields connected with the `IsRx` and `IsOTC` columns are displayed if the `Type` column contains the value defined by the `Terrasoft.Configuration.Consts.AccountTypePharmacy` invariable.

```
Terrasoft.Configuration.Consts = {
    AccountTypePharmacy: "d12dc11d-8c74-46b7-9198-5a4385428f9a"
};
```

You can use the 'd12dc11d-8c74-46b7-9198-5a4385428f9a' value instead of the invariable.

The [*Enabled/Disabled*] business rule (Terrasoft.RuleTypes.Activation) C#

This business rule enables and disables fields for entering values per condition.

Configuration object properties

`ruleType`

Should contain the `Terrasoft.RuleTypes.Activation` value for this rule.<

`triggeredByColumns`

The column array that triggers the rule.

`events`

An event array, defining the time of running business rules. It should contain values included into the `Terrasoft.BusinessRuleEvents` enumeration.

`conditionalColumns`

Condition array of business rule execution. Usually, these are specific column values.

`dependentColumnNames`

Column name array that the business rule is applied to.

Whether a field connected with the `Stock` column is enabled depends on the value in the `IsPresence` column.

Use case

```
Terrasoft.sdk.Model.addBusinessRule("ActivitySKU", {
    ruleType: Terrasoft.RuleTypes.Activation,
    events: [Terrasoft.BusinessRuleEvents.Load, Terrasoft.BusinessRuleEvents.ValueChanged],
    triggeredByColumns: ["IsPresence"],
    conditionalColumns: [
```

```
            {name: "IsPresence", value: true}
        ],
        dependentColumnNames: ["Stock"]
    });
```

The [*Filtration*] business rule (`Terrasoft.RuleTypes.Filtration`)

This business rule can be used for filtration of lookup columns by condition, or by another column value.

Configuration object properties

`ruleType`

Should contain the `Terrasoft.RuleTypes.Filtration` value for this rule.

`triggeredByColumns`

The column array that triggers the rule.

`events`

An event array, defining the time of running business rules. It should contain values included into the `Terrasoft.BusinessRuleEvents` enumeration.

`filters`

Filter. The property should contain the `Terrasoft.Filter` class instance.

`filteredColumn`

The column used for filtering values.

The [*Mutual Filtration*] business rule (`Terrasoft.RuleTypes.MutualFiltration`) C#

This business rule enables mutual filtering of two lookup fields. Works only with columns with the “one-to-many” relationship, e.g., [*Country*] – [*City*]. Create a separate business rule for every field cluster. For example, for the [*Country*] – [*Region*] – [*City*] and the [*Country*] – [*City*] clusters, create three business rules:

- [*Country*] – [*Region*];
- [*Region*] – [*City*];

- [*Country*] – [*City*].

Configuration object properties

`ruleType`

Should contain the `Terrasoft.RuleTypes.MutualFiltration` value for this rule.

`triggeredByColumns`

The column array that triggers the rule.

`connections`

Object array that configures cluster relationship.

Mutual filtration of the [Country], [Region] and [City] fields

```
Terrasoft.sdk.Model.addBusinessRule("ContactAddress", {
    ruleType: Terrasoft.RuleTypes.MutualFiltration,
    triggeredByColumns: ["City", "Region", "Country"],
    connections: [
        {
            parent: "Country",
            child: "City"
        },
        {
            parent: "Country",
            child: "Region"
        },
        {
            parent: "Region",
            child: "City"
        }
    ]
});
```

Mutual filtration of the [Contact], [Account] fields

```
Terrasoft.sdk.Model.addBusinessRule("Activity", {
    ruleType: Terrasoft.RuleTypes.MutualFiltration,
    triggeredByColumns: ["Contact", "Account"],
    connections: [
        {
```

```
            parent: "Contact",
            child: "Account",
            connectedBy: "PrimaryContact"
        }
    ]
});
```

The [*Regular expression*] business rule (`Terrasoft.RuleTypes.RegExp`) C#

Verifies the conformity of the column value with the regular expression.

Configuration object properties

`ruleType`

Should contain the `Terrasoft.RuleTypes.RegExp` value for this rule.

`RegExp`

Regular expression whose conformity with all the `triggeredByColumns` array columns is verified.

`triggeredByColumns`

The column array that triggers the rule.

Use case

```
Terrasoft.sdk.Model.addBusinessRule("Contact", {
    ruleType: Terrasoft.RuleTypes.RegExp,
    regExp : /^([0-9\(\)\/\+ \-]*)$/
    triggeredByColumns: ["HomeNumber", "BusinessNumber"]
});
```

Custom business rules

When adding a custom business rule via the `Terrasoft.sdk.Model.addBusinessRule(name, config)` method you can use properties of the `config` configuration object of the base business rule. In addition, the `executeFn` property is also provided.

Configuration object properties

`ruleType`

Rule type. For the custom rules it should contain the `Terrasoft.RuleTypes.Custom` value.

`triggeredByColumns`

Array of columns which initiates trigging of the business rule.

`events`

Array of events determining the start time of the business rule. It should contain values from the `Terrasoft.BusinessRuleEvents` enumeration. Default value: `Terrasoft.BusinessRuleEvents.ValueChanged`.

Possible values (`Terrasoft.BusinessRuleEvents`)

`Save`	the rule trigs before saving the data
`ValueChanged`	the rule trigs after changing the data (at modification)
`Load`	the rule trigs when the edit page is opened

`executeFn`

A handler function that contains the user logic for executing the business rule.

Properties of the `executeFn` handler function

Handler function signature

```
executeFn: function(record, rule, checkColumnName, customData, callbackConfig, event) {    }
```

Parameters

`record`	a record for which the business rule is executed
`rule`	an instance of the current business rule
`checkColumnName`	a column name that calls business-rules firing
`customData`	an object that is shared between all rules. Not used. Left for compatibility with previous versions
`callbackConfig`	a configuration object of the `Ext.callback` asynchronous callback
`event`	an event that triggered the business rul.

After the completion of function operation it is necessary to call either the `callbackConfig.success` or `callbackConfig.failure`.

Use cases options

```
Ext.callback(callbackConfig.success, callbackConfig.scope, [result]);
Ext.callback(callbackConfig.failure, callbackConfig.scope, [exception]);
```

Where:

- `result` – the returned boolean value obtained when the function is executed (`true`/`false`).
- `exception` – the exception of the `Terrasoft.Exception` type, which occurred in the handler function.

Methods

In the source code of the handler function, you can use the following methods of the model passed in the `record` parameter:

`get(columnName)`

Po get the value of a record column. The `columnName` argument should contain the column name.

`set(columnName, value, fireEventConfig)`

To set the value of the record column.

Parameters

`columnName`	the name of the column
`value`	the value assigned to the column
`fireEventConfig`	a configuration object to set the properties that are passed to the column modification event

---

`changeProperty(columnName, propertyConfig)`

For changing column properties except its value. The `columnName` argument should contain the column name and the `propertyConfig` object that sets the column properties.

The `propertyConfig` object properties

`disabled`	activity of the column. If `true`, the control associated with the column will be inactive and disabled for operation
`readOnly`	“Read only” flag. If `true`, the control associated with the column will be available only for reading. If `false` – the access for reading and writing
`hidden`	column visibility. If `true`, the control associated with the column will be hidden. If `false` – the control will be displayed
`addFilter`	add filter. If the property is specified, it should have a filter of the `Terrasoft.Filter` type that will be added to the column filtration. Property is used only for lookup fields
`removeFilter`	remove the filter. If the property is specified, it should have a name of the filter that will be removed from the column filtration. Property is used only for lookup fields
`isValid`	flag of column validity. If the property is specified, it will change the validity flag of the control associated with the column. If the column is invalid, then this can mean canceling of saving the record, and can also lead to the determining the record as invalid

Example of changing the properties (but not the values) of the `Owner` column

```
record.changeProperty("Owner", {
    disabled: false,
    readOnly: false,
    hidden: false,
    addFilter: {
        property: "IsChief",
```

```
            value: true
        },
        isValid: {
            value: false,
            message: LocalizableStrings["Owner_should_be_a_chief_only"]
        }
    });
```