FGS Configurator's Guide

This document describes how to use the FGS Configurator to create and export specifications of components**.**

The Configurator is a special web-based application for creating specifications of components. **The Configurator** is created on the base of the FGS Factory framework and provides web user interfaces to its database tables. These tables are called as the system tables of the Confgurator. In this document the "system" means only relating to the **Configurator** only.   It has its own menu, controller **Configurator** and the control script **configurator.php**. Along with standard components of the Framework, **the Configurator** uses also some special components, which can be used only in **the Configurator**. The instances of the components used in **the Configurator** are also crearted on the base of the specifications created in **the Configurator**. The specifications of these components have the «system» attribute set to "Yes".

It is strictly forbidden to remove or change these components' specifications with the «system» attribute set in "Yes".

The specifications used by **the Configurator** are stored in the system/specification directory. All files related to **the Configurator** are stored in the system/configurator directory.

The Configurator's specifications can be also considered as examples of configuring required component features.

Specifications of components are stored in the following system tables:

Table	Contents	Notes
fgs_column	Specifications of Grid's elements	Detail table for fgs_grid
fgs_component	Table of exporting components	
fgs_condition	Specifications of conditions	
fgs_controller	Specifications of controllers	
fgs_converter	Specifications of data converters	
fgs_dataset	Specifications of datasets	
fgs_debug	Debugging data	
fgs_element	Specifications of elements of input and search forms.	Detail table for fgs_form
fgs_export	Specifications of components' export	Detail table for fgs_component
fgs_field	Specifications of tables' fields	Detail table for fgs_table
fgs_filter	Specifications of input and search forms' filters	Detail table for fgs_form
fgs_form	Specifications of input and search forms	
fgs_grid	Specifications of Grids	
fgs_item	Specifications of menu options	Detail table for fgs_menu
fgs_list	Specifications of lists	
fgs_menu	Specifications of menus	
fgs_message	Language versions of messages	

fgs_option	Specifications of lists' options	Detail table for fgs_list
fgs_parameter	Table of exporting parameters of components	
fgs_predicate	Specifications of datasets' predicates	Detail table for fgs_dataset
fgs_role	User's roles.	
fgs_statement	Specifications of statements	Detail table for fgs_condition
fgs_table	Specifications of tables	
fgs_unit	Controllers' Units specifications	Detail table for fgs_controller
fgs_user	Users' data	
fgs_validator	Specifications of validators of input and search forms	Detail table for fgs_form

The Configurator presents web user interfaces to these database tables and exports data of these tables into special files of specifications. The files of specifications contain mainly ordinary PHP arrays. For each language of interface (English, Russian etc), its own set of specifications is created, which is stored in an individual directory. The specifications for each type of components are stored in individual directories. The paths to these directories should be specified in the configuration file of the application.

All specifications created by the developer should have the «system» attribute set in "No".

For all the database tables, which are not system tables of **the Configurator**, the «system» attribute must be set in "No". All applied specifications are stored in the application/specification directory.

In order to render **Form**, **Grid**, and **Search** components, **the Configurator** uses the standard **FgsFormView, FgsGridView** and **FgsSearchView** classes, which are part of the Framework. Under certain conditions, **the FgsFormView, FgsGridView** and **FgsSearchView** components allow to quickly jump to the configuring both Form, Grid, and Search components themselves, as well as individual elements of input, output, or search. This is achieved through the formation of specific hyperlinks, which are formed only for the user with the role of «developer» and when the configuration variable $FGSVersion is set to «development». Doing so, the Configurator opens a new tab with the window code «instant_edit», in which configuring of the selected component takes place.

For example, when you double-click on name of a form or Grid, an additional tab opens to configure this form or Grid. When you double-click on label of an input element, a tab opens for editing the attributes of this input element. When clicking on a column header, tab is opened to edit attributes of this column.

With **the Configurator** assistance, it is possible to create hierarchical menus for the application under development. Specifications of the menus are also exported.

The Configurator uses the menu with sysetm id equal to «configurator». The menu's options are configured in such a way that input of specifications of each component is done in its own window. Owing to this, input of specification can always be resumed from the point it stopped.

The convenience of such approach can be seen in the following example. For example, when entering of an input form element's specification, it has turned out that there is no a required list.

The developer selects the Configurator menu's item "Specification->List" to enter the list's specification.Then he selects the menu item "Specification->Input form" and can resume to enter the element's specification from the place from which he switched over to entering specification of the list.

It is strictly forbidden to remove or change specification of the menu with an ID equal to «configurator»!

The Configurator has a two-level drop-down menu:

Menu of the 1st level	Menu of the 2nd level	Function
Specification	Input form	To enter specifications of input forms
	Grid	To enter specifications of Grids
	Search form	To enter specifications of search forms
	Controller	To enter specifications of controllers
	Dataset	To enter specifications of datasets
	List	To enter specifications of lists
	List's Option	To enter specifications of list options
	Condition	To enter specifications of conditions
	Converter	To enter specifications of converters
	Table's field	To enter specifications of table fields
	Menu	To enter specifications of menu
	DB scheme import	To import database tables' structure
System table	Message	To enter language versions of messages
	Table	To enter specifications of database tables and fields
	Parameter	To enter export parameters
	Export	To enter specifications of export.
Export	List	Export of specifications of lists
	Input form	Export of specifications of input forms
	Search form	Export of specifications of search forms
	Grid	Export of specifications of Grids.
	Dataset	Export of specifications of datasets
	Controller	Export of specifications of controllers.
	Menu	Export of specifications of menus
	Message	Export of language versions of messages

The **ddm** class is used to form menu.

In addition to the drop-down menu, **the Configurator** has an auxiliary panel to hide / display components, to set the desired interface language and to enable / disable displaying of debugging information.

Process of creation a Web-based application

Creation a Web-based application with the FGS Tools package consists in entering components' specifications in **the Configurator** and development of extensions of standard components.

The sequence of creation of components' specifications:

- Import of database tables structure
- Configuring tables' fields
- Configuring tables
- Configuring lists
- Configuring datasets
- Configuring conditions
- Configuring converters
- Configuring text messages
- Configuring input forms
- Configurable search forms
- Configuring Grids
- Configuring controllers
- Configuring menus
- Export of specifications

Creating database tables is not included within the purview of this package, so that the developer should create database tables using other programs.

Import of tables' structure

The first step is to import database tables' structure into the Configurator's system tables. To transfer the tables' structure in the system tables, the menu item of the Configurator Specification-> DB scheme import is provided.

In process of importing, the tables' data is stored in the "fgs_table" table and fields' data is stored in the "fgs_field" table. If a table has been deleted from the database, then all specifications of the components based on this table are removed from the respective tables. When deleting fields from tables, all specifications of the components relating to these fields are removed.

During importing process, on the basis of data about the type of the field, comparison conditions and components are selected to input, output and search the field. For example, if a field type is "date", then the InputDate component is selected as an input component, the ColumnDate is selected as a Column component, the InputDate is selected as a component to input of argument for search and «test range" is selected as a condition of comparison.

Every time you change the structure of tables, you need to import structure of tables and to export all specifications.

Configuring fields

To accelerate creation of specifications and minimize data to be entered, it is necessary to enter database fields' attributes for input, output and search. To configure only the fields, you need to select the menu item Specification->Table's field or you need to select the menu item System table->Table to configure both fields and tables unanimously.

Fields of tables are stored in the table "fgs_field".

Configuring a field consists in selecting an input component, column component, comparison condition, default values, etc. Value of the «field_label" field is used as the input component's label and the column component's header. If a list is used for the input or column component, then it is necessary to enter the list. If a field is a foreign key, then it is necessary to enter the referencing table, the primary key of this table and type of the referencing table. Type of the referencing table has possible values "master table" and "reference table".

If a field is to be entered by using a input component of the type «select multple» (SelectManyCheckbox, SelectManyListbox or SelectManyMenu), then the type of the field should be set in the value «Set», and the field «Dbtype ?» is set in the value "No."

field	Label	Function
field_table	Table	Table of a field
field_name	Name	Name of a field
field_type	Type	Type of a field.
field_dbtype	Dbtype?	An attribute showing that value of the field «field_type» is obtained during importing structure of tables. It is necessary to set value "No" for the fields which used input components of «select multple» type
field_dbcascade	Cascade update and delete by DBMS?	Applicable only for fields that are foreign keys. If DBMS support referential integrity and a field is set as foreign key then this attribute must be set to «Yes». Otherwise you musr set it to «No»
field_default	Default	Default value of a field
field_element	Input component	A component used to input a field
field_label	label	label for input components or header for column components
field_list	list	A list used for input or column components
field_column	Column component	A column component used to display a field in a Grid.
field_argument	Component of argument	A component used to input predicates' argument in a search form
field_predicate	Comparison conditions	Condition of comparison used for predicates for the field
field_lookup	Referencing table	the referencing table referenced by the field which is foreign key
field_primary_key	Referencing table's primary key	primary key of the referencing table referenced by the field, which is foreign key
field_relation	Referencing table's type	Referencing table can be of two types: • Master table • Reference table
field_align	Align	Alighning in a Grid's column.
field_size	Size	Attribute «Size» for InputText component.
field_maxlength	Maxlength	Attribute «Maxlength» for InputText component.
field_rows	Rows	Attribute «rows» for InputTextarea component.
field_cols	Columns	Attribute «cols» for InputTextarea component.

In inline editing mode it is possible to edit input component, Column component, component of input of the argument of the predicate, the comparison condition during search, the label of an input element and a list of input or output. To do this, it is necessary to double click on the desired field of the desired row in the table of rows.

Configuring tables

To configure tables, it is necessary to select the menu item System table-> Table

Configuring the tables consists in setting the attribute «system» and setting a master table if necessary.

For all the tables related to the application under development, the field «table_system» must be set to "No"

For the tables that are tables of the type "detail", it is necessary to input the corresponding master table.

Field	Label	Function
table_system	System?	An attribute showing belonging of a table to the Configurator
table_name	Name	Name of a table
table_primary_key	Primary key	A primary key of a table
table_unique_key	Unique key	A unique key of a table
table_master	Master table	

To configure fields of a table, it is necessary to click on the "Detail" icon of this table.

Confuguring lists

To create lists, it is necessary to select the menu item Specification->List

Lists are used when configuring input forms, Grids and search forms.

Lists' specifications are stored in two tables «fgs_list» and «fgs_option» associated by «master-detail» relationship.

Lists can be constant and variable. Options of constant lists are stored in the system table «fgs_option». Variable lists are created from rows of tables.

Fields «list_table_alias» and «list_field_alias» should be entered and be different in the case of using variable lists based on one and the same table and in one and the same component, such as Form or Grid.

field	label	Function
list_system	System?	An attribute showing belonging of a list to the Configurator. It should be set in "No" for the lists of application which is under development.
list_sid	Sid	System id of a list
list_table	Table	Table of a list. Its value is equal to «fgs_option» for constant lists.
list_table_alias	Table alias	Alias of a table.
list_primary_key	Field as key	A field used as the "value" attribute of options. Its value is equal to «option_sid» for constant lists.
list_numeric	Numeric? (for fgs_option table only)	It should be entered only for constant lists. The value depends on the type of the "value" attribute of options
list_display_field	Display field	A field used as a description of a list's option. Its value is equal to «option_xxx» for constant lists. If the field is different for different languages, then a part of the field's name corresponding to the code of a language has to be replaced with "xxx". When exporting a list, the Configurator will replace "xxx" with the code of an appropriate language
list_field_alias	Field alias	Alias of a field
list_null_option	List's null option	Description of the null option of a list. If it is not specified, 4 whitespaces will be used. Selecting "Null option" is equal "No option selected "
list_order	Clause order by in select query	List sorting order. Its value is equal to «option_index ASC» for constant lists.
list_where	Clause where in select query	Condition on loading rows from a table.
list_export	Export?	An attribute of a list exporting.
list_class	Loader class	Custom class of a list's options loader
list_dataset	Dataset	a dataset used to limit a variable list's options

To configure a constant list of options, it is necessary to choose the action «detail» for this list, and to enter options of this list.

Configuring options of constant lists

field	label	Function
option_sid	Sid	System id of an option
option_en	Option in English	Description of an option in English
option_ru	Option in russian	Description of an option in Russian
option_index	Index	Index of an option during rendering.

If you need access to options of all constant lists, then you need to select the menu item Specification-> Lists's option. Unlike the previous version, you will need to enter the system id of the list:

field	label	Comments

option_list	List	System id of a list
option_sid	Sid	System id of a list's option
option_en	Option in English	Description of an option in English
option_ru	Option in russian	Description of an option in Russian
option_index	Index	Index of an option during rendering.

Configuring datasets

To configure datasets, you need to select the menu item Specification->Dataset and enter its attributes and predicates.

Datasets are used when configuring lists and controllers.

Specifications of datasets are stored in two tables «fgs_dataset» and «fgs_predicate» associated by a «master-detail» relationship.

Dataset's attributes:

field	label	Comments
dataset_system	System?	An attribute showing relation of a dataset to the Configurator. It should be set in "No" for datasets of an application under development.
dataset_sid	Sid	System id of a dataset
dataset_table	table	Table of a dataset

To configure predicates, you need to choose an action «detail» for the desired dataset and to ener data of predicates. A predicate is a condition imposed on one or more fields of a table.

Examples of predicates:

• The field of a table is equal to selected value
• The field of a table is not equal to null (argument is not needed in this case)
• The sum of two fields of a table does not exceed a certain value (here, the condition is imposed on a number of fields of the table)
• The length of a field value does not exceed a certain value (here we use the SQL function for the field in testing).

The predicate usually consists of an argument, an operator and a connector. A connector is needed to connect the prediacte with the previous one. Predicates can be grouped. If a dataset consists of only one predicate, a connector is not used.

During construction of SQL code for a predicate, the argument value is computed by the Evaluator component. Therefore, to enter the value of the argument, you need to know evaluation algorithm of the Evaluator, which is described in detail in the FGS Factory framework guide.

Predicate's attributes:

field	label	comments
predicate_table	Table	Table of a field on which a predicate is imposed.
predicate_table_alias	Table alias	Alias of a table on which a predicate is imposed.
predicate_field	Field	Field on which a predicate is imposed.
predicate_field_alias	Field alias	Alias of a field on which a predicate is imposed.
predicate_function	Function	SQL function used for fields of a predicate.
predicate_argument	Argument	Value of argument of a predicate.
predicate_argument_type	Argument type	Value of argument type of a predicate.
predicate_operator	Operator	operator of a predicate
predicate_connector	Connector	Connector to connect a predicate to a previous one
predicate_fieldset	Group	A group to which a predicate is referred.
predicate_custom	Custom	custom class that construct SQL code of a predicate
predicate_required	Required?	An attribute showing necessity of a predicate, if a value of an argument is not assigned or equal to null.
predicate_index	Index	Index of a predicate determining the order of constructing SQL code of predicate

Configuring conditions

To configure conditions, you need to select the menu item Specification->Condition and enter conditions's and its statements' attributes.

Conditions are used when configuring validators and filters of input forms, search forms and validation of operations with rows displaying in Grid components. Conditions consist of one or more of the so-called statements.

The specifications of conditions are stored in two tables «fgs_condition» and «fgs_statement» associated by the «master-detail» relationship.

To specify a condition, it is necessary to enter:

Field	label	comments
condition_sid	Sid	System id of a condition
condition_type	Type	Type of a condition
condition_error	Error	Abbreviation to be used to denote error of a condition

To configure statements, you need to select the action «detail» for the desired condition and to enter these statements. Statements are further development of the idea of predicates applied to the validation of forms and actions with Grid's rows. As well as for the predicates, there are also comparison operators and connectors. Statements can also be united into groups.

Examples of statements:

• The session variable is equal to selected value
• The value of entered field is within a certain range
• The value of entered field is equal to the value of another field.
• Editing a row is allowed only if the user is registered with a specific role

When configuring statements, you need to know the algorithm of evaluation of the Evaluator component, particularities of exporting components using conditions and which parameters are offered to test conditions.

Statement's attributes

field	label	Comments
statement_connector	Connector	Connector to connect a statement to the previous one
statement_function	Function	PHP function used to calculate the value of the first operand
statement_operand1	Operand 1	First operand
statement_operator	Operator	Operand for comarioson of operands
statement_operand2	Operand 2	Second operand
statement_failure	Failure	An attribute of statement error.
statement_group	Group	A group to which a statement is related.
statement_index	Index	Index of statement determining the order for testing statements

Examples of statements' configuration you can find in "Configuring validators of the Form component".

Configuring converters

To configure converters, you need to select a menu item Specification->Converter and enter converters' attributes.

Converters are used when configuring filters of input and search forms and displaying data in Grids.

The specifications of converters are stored in the tables «fgs_converter»

field	label	Comments
converter_sid	Sid	System ID of a converter
converter_type	Type	Type of a converter
converter_static	Static method?	An attribute of static method of a converter

It should be noted that the system ID of a converter is also name of the converter's class.

Configuring messages

To configure messages, you need to select the menu item Specification->Message.

The specifications of messages are stored in the table «fgs_message»

Field	label	Comments
msg_abbr	Abbreviation	Abbreviation to denote a message
msg_en	Text in english	Text of a message in English
msg_ru	Text in russian	Text of a message in Russian
msg_system	System?	An attribute of a message to the Configurator

Configuring menus

To configure menus, you need to select the menu item Specification->Menu and enter menus' and its items' attributes.

The specifications of menus are stored in two tables, «fgs_menu» and «fgs_item» tables, associated by the «master-detail» relationship.

Field	label	Comments
menu_id	Sid	System Id of a menu
menu_name	Name	Name of a menu
menu_system	System?	An attribute of relation of a menu to the Configurator

To configure the options of a menu, you need to select the action «detail» for the desired menu and to enter these options.

For each menu, there should be allocated a specific range in which the ID of an option should be located. The ID of an option specifies the order for displaying the menu options

field	label	Comments
item_id	Id	Identifier of an option
item_pid	Pid	Identifier of a parent option
item_type	Type	Type of an option
item_name	Name	Name of an option
item_action	Action	A script activated when selecting an option
item_class	Class	CSS class of an option
item_text	Text	Text displayed by JavaScript in dialog box when selecting an option
item_condition	Condition	Condition of visibility of an option
item_target	Target	Code of a box in which the script will be activated
item_childs	Child options functoin	Function to generate options for an option, type of a "menu".

Configuring input forms (the Form component)

To configure **Forms** or input forms, you need to select the menu item Specification->Input form.

The specifications of input forms are stored in four tables, namely: «fgs_form», which is the master table for 3 tables «fgs_element», «fgs_validator» and «fgs_filter». The table «fgs_element» stores specifications of input elements, the table «fgs_validator» stores validators, and the table «fgs_filter» stores filters of the input data.

Configuring the input form consists in entering the following data:

• Common data of the input form itself
• The attributes of elements and buttons of the input form.
• The attributes of validators of input data
• The attributes of filters of input data

Common data of an input form:

Field	label	Comments
form_system	System?	An attibute showing relation of an input form to the Configurator. It should be set in "No" for the nput forms of an application under development.
form_type	Type	A type of a input form. It should be set in "application" for the forms of an application under development.
form_table	Table	Base table of an input form.
form_sid	Sid	System Id of an input form
form_title	Title	Name of an input a form
form_modes	Modes	Input modes of an input form
form_startmode	Stating mode	Start mode of an input form
form_action	Action	Attribute «action» of an input form
form_method	Method	Attribute «method» of an input form
form_id	Id	Attribute «id» of an input a form
form_rowid_after_insert	Rowid after insert	This attribute is necessary if a primary key is of auto increment type and we need it's generated value for just added row. This attribute's value has to equal to name of the primary key /
form_onreset	Onreset	JavaScript function called when the «onreset» event occurs.
form_onsubmit	Onsubmit	JavaScript function called when the «onsubmit» event occurs
form_initial	Initial values of properties	This parameter is used to set initial values of properties or to add custom propetties of the Form component
form_redirect_after_insert	Redirect	If this parameter is set to «Yes», then after inserting a new

	page after insert ?	row will be made a page redirect to avoid inserting a row after the page refreshing

To create a new input form, you can copy the form with the “FormTemplate” system id, which is the input form of the type "template". This will give you a blank form with a set of preconfigured standard buttons. To add input fields, it is necessary to select the action «Add». After that there will be displayed the table with not yet added fields of the form’s base table. Having marked required fields, you need to click on the button "Add Fields". It should be remembered that the fields are added with the attributes set during configuration of these fields.

To configure the elements and buttons of an input form, you need to select the action "element" for the required input form. After that an input form and table of elements and buttons are displaed. In the inline editing mode, you can change attributes: the input component, label, index and fieldset.

To configure the input elements, there are provided two forms. The first form is required for pre-configuring.

Field	label	Comments
element_table	Table	A table of a field
element_table_alias	Table alias	Alias of a table of a field
element_field	Field	Input field. If this field is different for different languages, then it is possible to change for “xxx” the part of a name of the field corresponding to the code of a language. When exporting, the Configurator will replace “xxx” for the code of an appropriate language.
element_alias	Field alias	Alias of a field
element_sid	Sid	System id of a field
element_name	Attribute name	Attribute “name”.
element_type	Type	Type of a field
element_component	Component	Input component
element_extension	Component’s extension	Input component’s extension
element_index	Index	Index for processing and displaying a field.
element_label	Label	A label of the input component.
element_modes	Input Mode	Possible input modes
element_hidden	Hidden?	“Hidden” attribute of a field. If it set then the fieldis is not rendered
element_fieldset	Fieldset	Fieldset of a field

Final configuration of input elements depends on the selected input component and, therefore, you need to choose action «Attribute» for the desired input element. In doing so, there is displayed is a form with input fields united into 4 groups:

• System attributes
• Input attributes
• Ajax Attributes
• Events

In this case, a set of input fields in these groups depends on the type of the selected input component.

Configuring the system attributes.

Field	label	Comments
element_component	Component	A component which is used for input.
element_table	Table	A table of a field
element_table_alias	Table alias	Alias of a table of a field
element_field	Field	Input field. If this field is different for different languages, then it is possible to change for “xxx” the part of a name of the field corresponding to the code of a language. When exporting, the Configurator will replace “xxx” for the code of an appropriate language
element_alias	Alias	Alias of a field
element_sid	Sid	The system identifier of a field. It is required to recognize the fields of input of various tables having the same name.
element_index	Index	Number of displaying and processing of the field
element_event	Set value by event	See Lesson 9

Configuring input attributes

Field	label	Comments
element_fieldset	Fieldset	Fieldset of a field
element_name	Attribute name	The”name” attribute of an input field
element_label	Label	A label of input
element_required	Required?	An attribute of necessity of input
element_list	List	A list of input
element_filter	Filter	A filter on a list of input
element_layout	Layout	Direction of the options output for SelecetOneRadio and SelectManyCheckbox components
element_null_option	Null option	A null option of a list
element_default	Default	Default value
element_register	Global name	Global name of an input element
element_cai	Clear after insert?	An attribute of clearing a value of input after inserting a new row
element_left_table	Left table	A left table in the clause “join”
element_left_alias	Left table alias	Alias of a left table in the clause “join”
element_left_foreign_key	Left table foreign key	A foreign key to the left table in the clause join
element_where	Where	A value Where in a clause join.
element_cols	Columns	A value of the ‘cols’ attribute for textarea.
element_rows	Rows	A value of the “rows” attribute for textarea.

element_size	Size	A value of the “size” attribute to input type “text” and input components such as SelectOneListbox and SelectManyListbox
element_maxlength	Maxlength	A value of the “maxlength” attribute to input type “text”
element_renderer	Renderer	A renderer of an element
element_converter	Converter	A converter of an element
element_readonly	Readonly	the “readonly”attribute
element_id	Id	The “id” attribute of input element
element_value	Value	A value of an element
element_tabindex	Tabindex	The “tabindex”attribute
element_accesskey	Accesskey	The “accesskey”attribute
element_sequence	sequence	A sequence used to generate values of a field. Reserved for for the future to use DBMS Oracle
element_path	Upload path	A file loading directory
element_filesize	Filesize	A maximal size of an uploaded file
element_file_extension	Extensions	Possibel extensions of an uploaded file
element_filename	Filename	A method of creation of an uploaded file’s name
element_width	Width	A width of an uploaded image. Reserved for the future
element_height	Height	A height of an uploaded image. Reserved for the future
element_trim	Strip whitespace?	An attribute of removal of trailing spaces

Configuring buttons

Field	label	Comments
element_action	Action	An action initiated by a button
element_event	Event	An event generated after completion of an action
element_confirm	Confirm?	An attribute for confirmation of an action when pressing a button
element_file	Image	A name of an image for a button
element_component	Component	A component of a button. Must be set to “InputButton”
element_index	Index	Index of button rendering
element_name	Attribute name	The ”name” attribute of an input button
element_renderer	Renderer	A renderer of a button
element_id	Id	The ”id” attribute of a button
element_value	Value	The ” value” attribute of a button
element_tabindex	Tabindex	The “tabindex”attribute of a button
element_accesskey	Accesskey	The “accesskey”attribute of a button

The buttons of input forms should not have an attribute value "Action" equal to "Clear filter" or "Set filters» which are for search forms only.

Configuring ElementTableJoiner pseudo-component

This pseudo-component was introduced to solve the following problem.

Let's suppose we need to create an input form to a table of invoices per companies. In addition to invoices table, we also have tables of companies, countries and regions. Companies are tied to the countries and the countries to regions.

A company for the invoice must be chosen from a list of companies. The table of companies contains a large number of entries. Therefore, input of a company from a large list of companies is not very convenient. In order to reduce the options of a list, it is convenient to do input of a company with a chain of dependent selects:

Select the region
Select the country from the region
Select the company from the country

However, the table of invoices does not include codes of regions and countries. The table of companies does not include the region code. Therefore, to obtain the necessary data, we need to connect the tables of companies, countries and regions to the table of invoices.

This problem is solved by the ElementTableJoiner pseudo-component, defined by the following attributes:

Field	label	Comments
element_table	Table	table to join
element_table_alias	Table alias	Alias of a table to join
element_field	Primary key	A primary key of a table to join
element_alias	Alias	Alias of a primary key
element_left_table	Left table	Left table
element_left_alias	Left table alias	Alias of a left table
element_left_foreign_key	Left table foreign key	A foreign key of a left table
element_where	Where	Reserved fo the future

Configuring of Ajax attributes.

Configure Ajax attributes is necessary for the following input elements:

• A component of input element is InputAutocomplete or InputMultipleAutocomplete.
• A value of input element should be passed in the Ajax request to another component of the input with InputAutocomplete element.
• An input element participates in a chain of dependent selects either as a participant or as a parameter.

In one form, there may be several elements of the input with InputAutocomplete or InputMultipleAutocomplete component. Therefore, you must indicate the number of autocomplete, which involves an element of the input as either a participant or as a parameter.

Because in one form, there may also be several chains of dependent selects, you must indicate a number of a chain, which involves an element of the input.

Numbering autocomplete and chains of dependent selects should be started from 0.One and the same element can participate in several autocomplete as a parameter. In this case, the numbers of relevant elements of autocomplete should be listed separated from each other by commas.

Similarly, one and the same input element can participate in several chains of dependent selects and a numbers of the chains must also be listed separated from each other by commas

Configuration Ajax attributes of the InputAutocomplete and InputMultipleAutocomplete components

Field	Label	Comments
element_method	Method	Method to send a request
element_autocomplete	Autocomplete number	A number of autocomplete, in which an input element participates
element_token	token	A token to delimit one value from another in the InputMultipleAutocomplete component
element_search_field	Search field	A field of a list's table in which search of the input symbols is carried out.
element_min_chars	Min chars for request	The minimal number of input symbols required to send a request
element_max_options	Max of list options	The maximum options of a list which should be returned from the server in a request.
element_chain	Select chain	A number of a seect chain on which an input element participates.
element_tier	Tier in select chain	A number of a level of a select chain on which an input element participates.
element_before_request	Function before request	A function, which should be called before sending a request
element_after_request	Function after request	A function, which should be called after sending a request
element_url	URL	URL of a request. If it is not specified, then for autocomplete will be used autocomplete.php
element_callback	Callback function	Request's "callback" function
element_timeout	Timeout	The waiting time after pressing a symbol after which a request is sent

Configuring the Ajax attributes for a component of the type "select" either involved as a parameter of the InputAutocomplete or InputMultipleAutocomplete components, either involved in the chain of dependent selects (chained select), or as a participant or as a parameter.

Field	Label	Comments
element_method	Method	A method of sending a request
element_autocomplete	Autocomplete number	A number of autocomplete, in which input element participates

element_chain	Select chain	A number of “select” chain select, in which input element participates
element_tier	Tier in select chain	A number of “select” chain’s level, in which input element participates
element_chain_role	Role in select chain	A role, which input element, plays in select chain.
element_before_request	Function before request	A function, which should be called before sending a request
element_after_request	Function after request	A function, which should be called after sending a request
element_url	URL	URL of a request. If it is not specified, then for chained selects the list.php is used.
element_callback	Callback function	Request’s “callback” function

Configuring events

Field	Label	Comments
element_onblur	onblur	An action done in case of “onblur” event
element_onchange	onchange	An action done in case of “onchange” event
element_onclick	onclick	An action done in case of “onclick” event
element_ondblclick	ondblclick	An action done in case of “ondblclick” event
element_onfocus	onfocus	An action done in case of “onfocus” event
element_onkeydown	onkeydown	An action done in case of “onkeydown” event
element_onkeypress	onkeypress	An action done in case of “onkeypress” event
element_onkeyup	onkeyup	An action done in case of “onkeyup” event
element_onmousedown	onmousedown	An action done in case of “onmousedown” event
element_onmousemove	onmousemove	An action done in case of “onmousemove” event
element_onmouseout	onmouseout	An action done in case of “onmouseout” event
element_onmouseover	onmouseover	An action done in case of “onmouseover” event
element_onmouseup	onmouseup	An action done in case of “onmouseup”
element_onselect	onselect	An action done in case of “onselect” event
element_onblur	onblur	An action done in case of “onblur” event

Configuring validators of the Form component

To configure validators of an input form, it is necessary to choose the action "validator" for the desired form

Field	Label	Comments
validator_field	Field	A form’s field to be checked. If the field is different for different languages, then it is possible to replace with “xxx” the part of a field relared to the code of a language. When doing export, a configurator will replace the “xxx” with the code of an appropriate languge.
validator_condition	Condition	A condition which should be matched by entered

		value of the field.
validator_parameter	Parameter	A parameter of a condition.
validator_parameter_type	Parameter type	A type of a parameter of a condition.
validator_error	Error	An abbreviation of a message, which describes the error of the input data.
validator_class	Class	A class of non-standard conditions
validator_class_static	Static method?	A type of a method of a non-standard condition.
validator_break_onfailure	Break on failure	An attribute for breaking validation of a field in case of input failure.
validator_index	Index	Index for checking a validator

When you export an input form's' specification, all the validators belonging to the same input field are exported to the specification of this field. Therefore, the fields «validator_index» and «validator_break_onfailure» will have meaninhg only if you have several validators for one and the same input field.

To properly configure the validators, you need to know how a process of validation in the Form component and export of validators are carried out. This happens as follows:

The Form component calls the "validate" method of its input element in a loop with the "formValue" associative array as an argument. The" formValue" is an array of the Form input components' values as well as the form code in the array element with index «xxx_sid» and mode "form" in the element of array with index «xxx_mode». Input elements, in turn, will organize a cycle of checking all its validators. If a class of non-validator is specified, the static method «test» of the Validator component is called, or, otherwise, the static method «test" of the ConditionTester component is called. Whem doing so, the data of the validator and the "formValue" array are passed to the "test"method of the Validator component, while the array of statements and "formValue" array are passed to the ConditionTester component.

Let's consider how to configure a validator and how the validator is exported for several cases

Case 1 - the value of one field "field1" must be equal to the value of another field "field2".

Field	Label	Comments
validator_field	Field	field1
validator_condition	Condition	Equal
validator_parameter	Parameter	field2
validator_parameter_type	Parameter type	Input element

"Equal" condition consists of a single statement, which is configured as follows:

Field	Label	Comments
statement_connector	Connector	AND
statement_function	Function	
statement_operand1	Operand 1	%statement_operand1
statement_operator	Operator	Equal
statement_operand2	Operand 2	%statement_operand2

statement_failure	Failure	An error in case on non-performance.
statement_group	Group	
statement_index	Index	0

When exporting this statement, a value «% statement_operand1» is replaced for «& arg field1», and the value «% statement_operand2» is replaced for «& arg field2».

Let us now consider how the ConditionTester component checks this statement:

To obtain the values of the operands 1 and 2, the "get" static method of the Evaluator component is used, to which both the estimated value and the "formValue" array are passed. The latter, as said before, is passed also to the "ConditionTester" component. The "Evaluator" component, when gets the value of «& arg field1», returns the value of the "formValue" array with the index "field1", otherwise, on getting the value of «& arg field2», it returns the value of the "formValue" array with the index "field":

```
$ operand1 = $ formValue [field1];
$ operand2 = $ formValue [field2];
```

Further the "ConditionTester"component returns the test result of the Boolean expression:

```
($ operand1 == $ operand2)
```

Case 2 - the value of one "field1" field should be less than certain value of MaxValue

Field	Label	Comments
validator_field	Field	field1
validator_condition	Condition	Less
validator_parameter	Parameter	MaxValue
validator_parameter_type	Parameter type	A scalar value

"Less" condition consists of a single statement, which is configured as follows:

Field	Label	Comments
statement_connector	Connector	AND
statement_function	Function	
statement_operand1	Operand 1	%statement_operand1
statement_operator	Operator	Меньше
statement_operand2	Operand 2	%statement_operand2
statement_failure	Failure	An error in case on non-performance.
statement_group	Group	
statement_index	Index	0

When exporting statement, the value «% statement_operand1» is replaced for «& arg field1», and the value of «% statement_operand2» is replaced for the MaxValue

The values of the operands 1 and 2 are evaluated as follows with the help of the Evaluator component:

$ operand1 = $ formValue [field1];
$ operand2 = MaxValue

Further the “ConditionTester”components returns the test result of the Boolean expression:

($ operand1 <$ operand2)

Case 3 - the value of one “field1” field should be in a range from MinValue to MaxValue

Field	Label	Comments
validator_field	Field	field1
validator_condition	Condition	Range
validator_parameter	Parameter	‘min’=>MinValue,’max’=> MaxValue
validator_parameter_type	Parameter type	An array

“Range” condition consist of a single statement, which is configured as follows

Field	Label	Comment
statement_connector	Connector	AND
statement_function	Function	
statement_operand1	Operand 1	%statement_operand1
statement_operator	Operator	range
statement_operand2	Operand 2	%statement_operand2
statement_failure	Failure	An error in case on non-fulfillment
statement_group	Group	
statement_index	Index	0

When exporting the statement, the value of «% statement_operand1» is replaced for «& arg field1», and the value of «% statement_operand2» is replaced for the “array('min' => MinValue, 'max' => MaxValue)”.

The values of the operands 1 and 2 with the help of the “Evaluator” component are evaluated as follows:

$ operand1 = $ formValue ['field1'];
$ operand2 = array ('min' => MinValue, 'max' => MaxValue);

Further the “ConditionTester”component returns the test result of the Boolean expression:

($ operand1> = $ operand2 ['min'] && $ operand1 <= $ operand2 ['max'])

Configuring filters of the Form component

To configure an input form filters, it is necesary to select the action "filter" for the desired form.

Filters can be both with and without a condition. Filters with no condition shall be applied at all times. Filters with a condition are applied if the condition is proved to be correct. In connection with this, filter configuring is divided into the configuring the filter itself and configuring the filter's condition:

Field	Label	Comments
filter_field	Field	A form's field to be filtered. If the field is different for different languages, then it is possible to replace with "xxx" the part of a field relared to the code of a language. When doing export, a configurator will replace the "xxx" with the code of an appropriate languge.
filter_converter	Converter	A converter applied to the field
filter_parameter	Parameter	A parameter of filtering
filter_parameter_type	Parameter type	A type of the filtering parameter
filter_index	Index	A procedure for using a filter
filter_condition	Condition	A condition to be checked before doing filtering
filter_condition_field	Condition field	A field of the form to be checked
filter_condition_parameter	Condition parameter	A condition parameter
filter_condition_parameter_type	Condition parameter type	A type of a condition parameter
filter_condition_class	Condition class	A class of non-standard condition
filter_condition_class_static	Static method?	A type of method of non-standard condition

Configuring a condition of filtering doesn't differ in anything from configuring validators.

Several filters may be set on one and the same field. Therefore, the value of index is meaningful when there are several filters for one and the same input field.

The process of filtering the entered input data takes place after validation of the data:

The "Form" component invokes the "filter" method of all its input elements with the "formValue" array as argument. An input element, in turn, organizes the filtration cycle for all of its filters by using the Converter component, to which the value of the filtered field is passed, the data of the current filter and the "formValue" array.

Configuring search forms (the Search component)

To configure search forms, it is necessary to select the menu item Specification->Search form.

The specifications of search forms are stored in the same four tables, in which the input forms are stored: the «fgs_form» table, which is the master table to 3 tables, namely, «fgs_element», «fgs_validator» and «fgs_filter» tables. In the «fgs_element» table, there are stored the specifications of search predicates, in the «fgs_validator» table - validators and in the«fgs_filter» table - filters of the input data.

Due to the fact that the general information about the input forms and the search forms are stored in one and the same table and the code of the ID forms must be unique, it is recommended to search forms to assign identifiers beginning with the «Search».

Configuring the search forms in much is the same configuring input forms.

Configuring the search form consists is in entering the following data:

• General data of the search form itself
• The data of the elements and buttons of the search form
• The data of validators of the entered data
• The data of filters of the entered data

General data of the search form:

Field	Label	Comments
form_system	System?	An attribute of relationship of the search form to the Configurator. It should be set in "No"for the search formsof the developed application.
form_type	Type	A type of a form. It should be set in "search of the applied-oriented".
form_table	Table	A table of a form
form_sid	Sid	A code of a form
form_title	Title	A name of a form
form_action	Action	The "action" attributeof a form.
form_id	Id	The «id» attribute of a form
form_onreset	Onreset	A JavaScript function invoked when occurrence of the "onreset" event takes place.
form_onsubmit	Onsubmit	A JavaScript function invoked when occurrence of the "onsubmit" event takes place.
form_method	Method	the "method" attribute of a form
form_initial	Initial values of properties	This parameter is used to set initial values of properties or to add custom propetties of the Search component

To create a new search, you can copy the form with the system id SearchTemplate, which is an search form of the type "search pattern". This will allow getting a blank search form with a set of preconfigured standard buttons. To add the components to input predicates, you need to choose an action «Add». This displays a table with fields not yet added of the base table of the search form. Having marked the desired fields, you need to click on the button "Add Fields". It should be remembered that the fields are added together with the attributes set during configuring the fields.

To configure the components to input predicates, it is necessary to select the action "element" for the required search form. After that, the input form and the table of predicates and buttons are displayed. In the mode of inline editing, it is possible to change the component of argument input, the condition comparison, label, index and fieldset.

We shall remind that the predicate input component in the general case is a container that contains the three components of input:

• Component (SelectOneRadio) to input a connector to connect a predicate to the previous predicate
• Component (SelectOneMenu) to enter the comparison operator
• The component to input value of an argument of the predicate

To configure the search predicates, there are three forms. The first form is required for pre-configuration of the predicate:

Field	Label	Comments
element_table	Table	A table of a field
element_table_alias	Table alias	Alias of a table of a field
element_field	Field	A field of input
element_alias	Field alias	Alias of a field
element_sid	Sid	A system id of a field
element_type	Type	A type of a field
element_label	Label	A label of the input
element_component	Component	A component of an argument input.
element_index	Index	An index of processing and displaying of a predicate
element_predicate	Comparison conditions	Comparison conditions
element_predicate_custom	Custom predicate	A class of non-standard predicate
element_hidden	Hidden?	the “hidden” attribute of a predicate
element_fieldset	Fieldset	“Fieldset” of a predicate

The final configuring of the argument of the predicate depends on the chosen input component and does not differ in anything from the final configuring of the elements of input forms. To do this, it is necessary to select the action «Attribute» for the desired predicate.

The buttons of the search form should have one of the 3 values of the "Action" attribute:

• Clear filter
• Set filter
• Cancel

The final configuring of a connector and an operator of the predicate depends on the conditions of comparison and to go to it, it is necessary to choose the action "predicate" for the desired predicate.

If you select the comparison condition "Range check", then you need to fill out the form below:

Field	Label	Comments
element_table	Table	A table of a field
element_field	Field	A field of a predicte
element_type	Type	A type of a field
element_function	SQL Function	SQL function

element_layout	layout	Direction of output for the input elements of the argument's maximum and minimum.
element_min_label	Label for min	A label for the input element of the argument's minimum.
element_max_label	Label for max	A label for the input element of the argument's maximum.
element_operators	Operators	Possible operators
element_operator_default	Operator Default	A deafault operator
element_fix_operator	Fix operator?	An attribute of possibility of changing an opearor by a user
element_connector	Connector Default	A default connector.
element_fix_connector	Fix connector?	An attribute of possibility of changing a connector by a user
element_fieldset	Fieldset	Feldset of a predicate

If you select the comparison condition, which is not equal to "Range check", then you need to fill out the form below:

Field	Label	Comments
element_table	Table	A table of a field
element_field	Field	A field of a predicte
element_type	Type	A type of a field
element_function	SQL Function	SQL function
element_operators	Operators	Possible operators
element_operator_default	Default operator	A default operator
element_fix_operator	Fix operator?	An attribute of possibility of changing an opearor by a user
element_connector	connector	A default connector
element_fix_connector	Fix connector?	An attribute of possibility of changing a conector by a user
element_fieldset	Fix operator?	“Fieldset” of a predicator

The attribute "Field" may include not only a name of one field, but the valid combination of fields. Thus, we can organize a search based on various combinations of fields and using the various SQL functions.

Configuring the buttons of a search form

Field	Label	Comments
element_action	Action	An action initiated by a button
element_event	Event	An event generared on ending of an action
element_confirm	Confirm?	An attribute to confirm the action on pressing a button
element_file	Image	A caption of an image for a button
element_component	Component	A component of a button
element_index	Index	An index of displaying of a button

element_name	Attribute name	The “name”attribute of a button
element_renderer	Renderer	A renderer of a button
element_id	Id	The “id” attribute of a button
element_value	Value	The “value” attribute of a button
element_tabindex	Tabindex	The “tabindex” attribute of a button
element_accesskey	Accesskey	The “accesskey’ attribute of a button

Configuring validators and filters for search forms are exactly the same as the configuring validators and filters for input forms.

Configuring Grids (the Grid component)

To configure a Grid, it is necessaryto select the menu item Specification->Grid.

The specifications of Grids are stored in two tables: «fgs_grid» table, which is the master table to «fgs_column» detail table. The table «fgs_column» stores specifications of columns and possible actions on rows.

Configuring of a Grid consists in entering the following data:

• General data of the Grid itself
• The data on column components, actions on individual rows and actions on a set of rows

General data of a Grid:

Field	Label	Comments
grid_system	System?	An attribute of the Grid’s relationship to the Configurator. For Grids of developed application, it should be set to "No".
grid_type	Type	A type of a Grid. It should be set to “Application” for the Grids of developed application.
grid_table	Table	The base table of a Grid
grid_sid	Sid	System id of a Grid.
grid_title	Название	A name of a Grid
grid_pagesize	Размер страницы	A number of rows displayed on a page
grid_order	Сортировка	Initial sorting of a Grid
grid_direction	Direction	Initial direction of sorting
grid_user_offset	User's offset?	An attribute of possible changing “Offset” by a user.
grid_user_order	User's order?	An attribute of possible changing sorting by a user.
grid_user_pagesize	User's pagesize?	An attribute of possible changing number of rows displayed on a page by a user.
grid_headerspan	Headerspan	A number of rows in a header of a Grid.
grid_id	Id	Attribute “id” of a Grid.
grid_modal	Modal?	An attribute of possible hiding a Grid by a user.
grid_inline_edit	Inline edit?	The “inline_editing” attribute.
grid_initial	Initial values of properties	This parameter is used to set initial values of properties or to add custom propetties of the Grid component

Remember that the Grid component allows editing in two modes:

Mode 1 - all editable fields in the Grid are already in edit mode and to save the changed data you need to click on the save button. Mode 1 is to emulate spreadsheet-like editing.

Mode 2 is the so-called inline editing, in which to enter the “edit” mode you need to double-click by mouse on the desired field of the desired row. Saving the changed data is done by pressing the Enter key for the input fields such as ColumnInputText or by selecting the desired option for fields with an input component ColumnSelectOneMenu.

When you set the “Inline editing” mode in the "Yes", editing is carried out in Mode 2, i.e. by double-clicking the mouse on the desired field of the desired item.

To create a new Grid, it is possible to copy the Grid of "template" type. When doing so, the Grid with the system id MasterGridTemplate is a template for the Grids of master tables, and the Grid with the code GridTemplate is a template for all others.Coping allows you to get an empty Grid with a set of preconfigured standard operations on rows. To add columns to a Grid, you need to select the action «Add». This displays a table with not yet added fields of the Grid’s base table. Checking the required fields, you need to click on the button "Add Field". It should be remembered that the fields are added with the attributes set during configuration of the fields.

To configure the columns, the operations on individual rows and operations with a set of rows, it is necessary to select the action "column." After that, the input form and the table of columns are dispalyed with operations on individual rows and operations on a set of rows. In the “inline editing” mode, it is possible to change the Column component, a header and index.

To configure the columns, there are two forms. The first form is for pre-configuring:

Field	Label	Comments
column_table	Table	A table of a column’s field
column_table_alias	Table alias	Alias of a table of a column’s field
column_field	Field	A field displayed in a column. Iif the field is different fordifferent languages, then it is possible a part of the field retating to a code of a language to change for “xxx”. When doing export, a configurator will replace “xxx” for the code of an appropriate language.
column_alias	Field alias	Alias of a column’s field.
column_type	Type	A type of s field.
column_component	Component	A Column component of a field
column_index	Index	Index of displaying of a column in a Grid
column_header	Header	A header of a column.
column_save	Save?	An attribute for saving a value of a field in a session variable.
column_hidden	Hidden?	An attribute for hiding a column. It should be set in “1” for the fields, which shouldno’t be displying in a Grid but which are required for supporting objectives.

column_align	Align	The "align" attribute of a column
column_width	Width	The "width" attribute of a column
column_function	SQL Function in select	SQL function in the "select"clause
column_sid	Sid	A system id of a column.
column_register	Global name	an element's key of the "globals" array of the "Registry" component, in which a field's value is stored.
column_sort	Sort as	If "0" is entered, it means not permitting column sorting; if "1" is entered, it means permitting field sorting. If a name of a field is entered, it means permitting column sorting , but sorting should be not based on a field of a column but sorting by the entered name
column_sort_prefix	Sort prefix	Constant value in the beginning of the "order by" clause
column_sort_suffix	Sort suffix	Constant value in the end of the "order by" clause

The attribute "Field" may include not only the name of the field, but a valid combination of fields. In this case, it is necessary to enter the attribute "Alias of a field." Thus, it is possible to display the sum or product of two or more fields.

The final configuring of columns depends on the selected Column component, and it is necessary to choose for it the action «Attribute» for the desired column. When doing so, there will be displayed a form with input fields, grouped into 5 groups:

• System
• Rendering
• Sorting
• Input Attributes
• Events

In this case, a set of groups of fields and input fields in these groups depends on the type of the selected component. For example, a group of "Input attributes" appears only for the components ColumnInputText and ColumnSelectOneMenu, a group of "Sorting" does not appear for the components GridButton, RowSelector, RowAction, and RowSetAction.

Configuring the attributes of the "System" group

Field	Label	Comments
column_table	Table	A table of a column's field
column_table_alias	Table alias	Alias of a column's field
column_field	Field	A field of a column. If the field is different for different languages, then it is possible to change for "xxx" the part of a name of the field related to a code of a language. When doing export, the Configurator will replace the "xxx" for a code of an appropriate language.
column_alias	Alias	Alias of a column's field
column_type	Type	A type of a column's field

column_sid	Sid	A system id of a colum
column_component	component	A Column component of a field
column_save	Save?	An attribute for saving a value of a column’s field in the sesiona variable.
column_register	Global name	A key of an element of the “globals” array of the “Registry”component in which the value of a column’s field is stored.
column_hidden	Hidden?	An attribute of hiding a column. It should be set in “1” for the fields, which are not required to be displayed in a Grid but which are requird for supporting objectives.
column_calculate	Calculate?	An attribute of summation values of a column. If it is set to "Yes", then the column will display the sum of values of a field of all derived rows.

Configuring the attributes of the “Rendering” group

Field	Label	Comments
column_renderer	Renderer	A column’s renderer.
column_index	Index	Index of displaying of a column in a Grid
column_header	Header	A header of a column.
column_span	Span	A formula for forming a column’s header
column_align	Align	The “align” attribute of a column
column_width	Width	The “width” attribute of a column
column_list	List	The list is used for display a value of a column’s field
column_relation	Relation	A type of relationship of a list’s table and the base table of a Grid.
column_join_lookup	Join lookup table?	The atribute to force making left join of the Grid’s base table with a list’s table
column_converter	Converter	A converter of a field’s value. Applicable for the ColumnText component only
column_format	Format	Ouput format- reserved for the future
column_function	SQL Function in select	SQL function in “select” clause

Let’s consider configuring multy-line headers in the following example. Let’s suppose we need to create the table header of the three lines:

ColumnSet1			ColumnSet2		Column6
ColumnSet3		Column3	Column4	Column5	
Column1	Column2				

To begin, we set the value of the attribute «Headerspan» for the Grid equal to 3.

The required values of the “Span” and “Header” attributes are shown in the table below:

Column	"Span" attribute	"Header" attribute
Column1	1*3,1*2,1*1	ColumnSet1, ColumnSet3, Column1
Column2	0,0,1*1	Column2
Column3	0,2*1,0	Column3
Column4	0,2*1,0	ColumnSet2, Column4
Column5	0,2*1,0	Column5
Column6		Column6

"0" means that there is no output.
"N*m" denotes the values of the attributes "rowspan" and "colspan", namely rowspan=n and colspan=m.
If the "Span"attribute is not set, this means that the column has a one-line header.

"Span" and "Header" attributes should reflect the algorithm of formation a header in 3 passes:

During the first pass, it is necessary to form ColumnSet1, ColumnSet2 and Column6.
During the second pass, it is necessary to form ColumnSet3, Column3, Column4 and Column5
During the third pass, it is necessary to form Column1 and Column2.

For the column Column1, during the first pass, it is necessary to form a cell 1*3 and output ColumnSet1 as a header. During the second pass, it is necessary to form a cell 1*2 and output ColumnSet3 as a header. During the third pass, it is necessary to form a cell 1*1 and output Column1 as a header.

For the column Column2, during the first and second passes, it is not necessary to output anything. During the third pass, it is necessary to form a cell 1*1 and output Column2 as a header.

For the column Column3, during the first pass, it is not necessary to output anything. During the second pass, it is necessary to form a cell 2*1 and output Column3 as a header. During the first pass, it is not necessary to output anything.

For the column Column4, during the first pass, it is necessary to form a cell 1*2 and ouput ColumnSet2 as a header. During the second pass, it is necessary to form a cell 2*1 and ouput Column4 as a header. During the first pass, it is not necessary to output anything.

For the column Column5, on the first pass, it is not necessary to output anything. During the second run, it is necessary to form a cell 2*1 and ouput Column5 as a header. During the third pass, it is not necessary to output anything.

Configure the fields of attributes of "Sorting" group

Field	Label	Comments
column_sort	Sort as	If "0" is entered, this means that sorting by a column is not permitted. If "1" is entered, this means that sorting a column as by the column's field is permitted. If a name of a field is entered, this means that sorting a column is permited, however, sorting should be done not by the column's field but according to entered name of a field.

column_sort_prefix	Sort prefix	A constant value in the beginning of an "order by" clause
column_sort_suffix	Sort suffix	A constant value in the end of an "order by" clause

Configuring the columns with the output component ColumnLookup

Field	Label	Comments
column_list	List	A list used for the output
column_relation	Relation	A type of relationship of a field and a list's table.
column_join_lookup	Join lookup table?	An attribute of conection a table of variable list and a base table of a Grid.

Let's consider in detaile the attribute "Do join with the table? ».

This attribute is used only for variable lists and only for the case when a field can store value of only one list's option: the field "column_relation" set to "one-to-one relationship ".
The other type of relation is the "one-to-many" relationship. It means only that a field can store values of several list's option. These types of relationship do not have any connection with the "one-to-one" and "one-to-many" types of relationship between two database tables.

If you set the attribute in "Yes", then in the SQL query the Grid component will perform joining the table of the variable list and the base table of the Grid (left join) and it will include the field - description of the option list – in the number of selected fields.

If you set the attribute in "No", then the Grid component will load a variable list only from the options requred to render a set of displaying rows.

If the type of relationship of a field with a variable list is "One to many", then the Grid component will load the variable list from the options, which are necessary to render a set of displaying rows.

Such handling of the ColumnLookup component's list is implemented to optimize the output of rows.

Configuring the ColumnTableJoiner pseudo-component

This pseudo-component is designed to connect the tables to the base table.

Field	Label	Comments
column_table	Table	A connected table
column_table_alias	Table alias	Alias a table of a column's field
column_field	Primary key	A primary key of a connected table
column_alias	Alias	Alias of a primary key
column_left_table	Left table	A left table
column_left_alias	Left table alias	Alias of a left table
column_left_foreign_key	Left table foreign key	A foreign key to the left table
column_join	Тип join	A "join" type

column_where	Where	Reserved
column_required	Required?	Mandatory of connection of a table

Let’s explain the use of the “Required” attribute.

Component **Search** allows searching by fields of a table, which is not the base table of the Search and Grid component. Therefore, the connection of a not base table is necessary only on setting the search condition to be fulfilled through the fields of not base table. In this case, the attribute should be set to "No".

If the connection of not base table is not associated with setting **search** conditions, then the attribute should be set to "Yes."

Configuring the actions over individual rows (RowAction)

Field	Label	Comments
column_index	Index	An index of output
column_action	Action	An executable action
column_immediate	Immediate?	An attribute of execution of an action by the **Grid** component itself
column_condition	Condition	A condition required for performing an action
column_condition_field	Condition field	A checkable field of condition
column_condition_class	Condition class	A class of non-standard condition.
column_condition_class_static	Static condition class	A static class of condition
column_condition_parameter	Condition parameter	A parameter of codition
column_condition_parameter_type	Condition parameter type	A type of condition’s parameter
column_name	name	A name of a colmn
column_file	Image	A file of an icon
column_renderer	Renderer	A class of a non-standard renderer

Configuring the action over a set of rows (RowSetAction)

Field	Label	Comments
column_index	Index	An index of output
column_action	Action	An executable action
column_immediate	Immediate?	An attribute of execution of an action by the **Grid** component itself
column_condition	Condition	A condition required for performing an action
column_condition_field	Condition field	A checkable field of a condition
column_condition_class	Condition class	A class of non-standard condition.
column_condition_class_static	Static condition class	A static class of a condition
column_condition_parameter	Condition parameter	A parameter of a codition
column_condition_parameter_type	Condition	A type of condition’s parameter

	parameter type	
column_name	Name	A name of a column
column_file	Image	A file of an icon

Configuring controllers

To configure the controllers it is necessary to select the menu item Specification->Controller.

The specifications of controllers are stored in two tables "fgs_sontroller" and «fgs_unit» associated by a «master-detail» relationship.

Field	Label	Comments
controller_system	System?	An attribute of relation of a controller to the Configurator. The attribute should be set to “No” for the controllers of athe developed application.
controller_sid	Sid	A system id of a controller
controller_type	Type	A type of a controller
controller_title	Name	A name of a controller
controller_class	Class	A class of a controller
controller_script	Script	A script of a controller
controller_template	Template	A template of a controller
controller_roles	Roles	A list of roles, which is accessable to a contrloller
controller_users	Users	A list of users to whom a controller is accessable
controller_initial	Initial	Initial values of a controller’s properties. An imput format should be as follows: ‘Name of property 1’=>value of property 1’, ‘Name of property 2’=>value of property 2’…
controller_session	Session	A list of properties of a controller, which should be saved in sessional variables. Name of properties should be enclosed in single quotes and separated one from another by commas.

To configure the components of the **Unit Controllers** it is necessary to select the action «detail» for the required controller and to enter the **Unit** data.

Field	Label	Comments
unit_sid	Sid	System id of a Unit
unit_type	Type	Type of a Unit
unit_class	Class	Class of a Unit
unit_form	Form's class	An input form of a Unit
unit_form_class	Form's class	A class of input form of aUnit
unit_form_renderer	Form renderer	A renderer of a Unit’s input form
unit_form_display	Display Form?	An attribute of display of a input form in the beginning
unit_form_hide	Hide form?	An attribute to hide an input form after completion of editing a row.
unit_grid	Grid	A Grid of a Unit
unit_grid_class	Grid's class	A class of a Grid of Unit.
unit_grid_renderer	Grid renderer	A rendere of a Grid of a Unit
unit_grid_display	Display Grid?	An attribute of display of a Grid in the

		beginning
unit_grid_hide	Hide grid?	An attribute to hide a Grid when editing a row in an input form of a Unit.
unit_grid_multimode_hide	Hide in multimode?	An attribute to hide of a Grid when operation over a number of rows selected
unit_search	Search	A search formof a Unit
unit_search_class	Search form's class	A class of a form search of a Unit
unit_search_renderer	Search renderer	A renderer of a search form of a Unit
unit_search_display	Display Search?	An attribute of displaying of a surch form in the beginning
unit_search_hide	Hide search form?	An attribute to hide of a search form when clicking on a search button.
unit_dataset	Dataset	A dataset of a Unit
unit_dataset_class	Dataset class	A class of a dataset
unit_initial	Initial	An initial value of Unit's properties
unit_session	Session	A list of properties of a Unit, which should be stored in session variables.

Configuring CRUD controllers

A type of a controller should be equal to «crud». There should be configured only one Unit and the type of this Unit should be equal to «crud».

Configuring MasterDetail controllers

The type of a MasterDetail controller should be equal to «MasterDetail». There should be configured only two Units and the types of this Unit should be equal to «master» and "detail". The base table of the master Unit's components and the base table of the detail Unit's components have to to assosiated by "master-detail" relationship. Besides, the base table of the detail Unit's components should have a field, which is a foreign key to the primary of the base table of the master Unit's components. The required configuring of the tables щшер "master-detail" relationship and foreign keys of such tables is given in the relevant sections of this document.

Configuring UnitSet controllers

Type of controller should be equal to "UnitSet". There should be configured at least two **Unit** and the type of one of Unit should be equal to "master" The base table of the master Unit's components and the base tables of all the detail Unit's components have to to assosiated by "master-detail" relationship. Besides, the base table of each detail Unit's components should have a field, which is a foreign key to the primary of the base table of the master Unit's components.

Export of Specifications

Export of components' specifications is carried out only for the current interface language. To export specifications for a particular language, this language should be set as the current one.

When selecting a menu item to export specifications of a component, the specifications of all components of the selected type are exported.

Export specifications of a component should be performed every time you change the specifications of both the component and specifications of the related components.

For example, in case of changing the specifications of messages, it is necessary first to carry out export of messages and then of all the other components.

Grids, input forms and search forms can use lists. Therefore, when changing specification of a list, it is necessary to carry out first export of lists and then export of input forms, search forms and Grids.

Controllers and lists can use data sets. Therefore, when changing specification of a dataset, it is necessary first to export datasets and then to export controllers, lists, input forms, search forms and Grids.

When exporting specifications, configurator validates correctness of the entered specifications, and all the bugs are displayed in the window of the output of debugging information.

Testing controllers

After exporting controllers, it is possible to test created controllers. To do this, it is necessary to find the required controller and to click the mouse on the field of the system id of the controller (Sid). After that, the Configurator will open a new tab with the code window «application» with the controller being tested.

We remind that when in development mode, it is possible to quickly switch over to configuring mode both input and search forms and Grids, as well as individual elements of these components. To go to editing specifications of input forms, it is necessary to double click on the name of the input form.

To go to editing a specification of individual item of the input form, it is necessary to double click on the label of the desired item.

To edit the specification of the search form, a double click on the name of the search form is required.

To go to editing a specification of an individual predicate, a double click on the label of the desired predicate is requred.

To go to the editing Grid specification, a double click on the name of the Grid is required.

To go to the editing specification of a separate column, a double-click on the desired column header is required.

Configuring the selected component is carried out on an individual browser tab with the code of the window «instant_edit».

Lessons of the Configurator

In this part of the document, examples of configuring of the Configurator's components are provided. These examples can be used to implement the desired features of the applications under development. To get familiar with the defined attributes, you are sometimes invited to click on the icon for editing components. I emphasize - for reference only! It is strictly prohibited to change anything and to click on the button "Update"! The examples show how the framework uses the set attributes for the implementation of the desired features.

Lesson 1 - Configuring messages

Configuring messages is an example of a user interface to a single table (fgs_message). This type of user interface defines the requirements for configuring the controller only.

Choose Specification->Controller item of the menu; find the controller with system id «message». This controller provides the described interface. Please note the following key attributes of the configuring:

Label	Meaning	Comments
Type	crud	
Class	Crud	
Script	configurator.php	This is a file of controlling script for the Configurator. It is necessary to specify the value of "application.php" for the application under development
Template	crud.php	This file serves as a template for the Crud interface for the Configurator. It is necessary to specify the value of "admin_crud.php" for the application under development

Click on the icon "detail" for this controller. Note that there is only one Unit with the system id "crud**"** and the type **"**crud"

When clicking on the system id of a component in the Grid of the **Unit**, Configurator proceeds to configuring the component selected in the tab with the window's code "instant_edit".

Lesson 2 - Configuring database fields

Configuring database fields is an example of a user interface to a single table (fgs_field). A feature of this interface is using a Grid to edit some fields in the mode of inline editing.

Choose the menu item Specification->Field. Next, double click the mouse on the name of the Grid; this will open a new tab to configure the Grid with the system id "fields". Come over to this tab. Click on the icon "Update". A form will appear with the general attributes of the Grid. A key attribute is the attribute with the label «Inline editing?" set to "Yes."

Then click on the icon "column." A table of the Grid colimns will be displayed. To view the set attributes of the input columns, click by mouse on the icon «Attribute» for fields with components and ColumnSelectOneMenu and ColumnInputText.

Lesson 3 - Configuring database tables

Configurating database tables is an example of a user interface to the two tables related by master-detail (fgs_table and fgs_field) relationship. Let's consider the key points of configuring of this interface.

Choose the menu item Specification->Field, find the field "field_table" for the "fgs_field" table and go to editing this field, just to see the set attributes. You shouldn't change anything when viewing! Note that for this field, the following attributes are defined:

Label	Value
Referencing table	fgs_table
Referencing table's primary key	table_name
Referencing table's type	master table

Choose the menu item Specification->Table, find the table fgs_field and note that the fgs_table table is specified as a "table master».

Next, double click the mouse on the name of the Grid, this will open a new tab (code of the window is "instant_edit") to configure the Grid with the system id "tables". Then go over to the tab with the window's code "instant_edit". Click on the icon "column." A table of the Grid's elements will be displayed. Click on the icon «Attribute» for RowAction component with the action "detail". A form with the defined attributes will appear. Please note the following attributes:

Label	Value
Action	Detail
Immediate?	No

Choose Specification->Controller item of the menu; find a controller with system id "table". This controller provides a described interface. Please note the following key attributes of the configuration:

Label	Value	Comments
Type	MasterDetail	
Class	MasterDetail	
Script	configurator.php	This is a file of controlling script for the Configurator. It is necessary to specify the value of "application.php" for the application under development
Template	md.php	This file serves as a template for the MasterDetail interface for the Configurator. It is necessary to specify the value of "admin_md.php" for the application under development

Click on the icon "Detail" for this controller. Note that there are two Units: one with a system id "detail" and the type of "detail" and the other is with the system id "master" and the type of "master".

Lesson 4 - Configuring of lists

Configuring of lists is an example of an interface to the two tables related by the “master-detail” relationship (fgs_list and fgs_option). This type of configuration is of interest owing to the configuration of chained selects and configuration of a component RowAction with a condition.

Select the menu item Specification->List. If a list’s input form is not visible then make visible by clicking on the hyperlink «Form» located on an extra panel.

The form of list input allows changing the dependent lists used for inputing fields labeled "Table" and "Dataset" in case of changing the attributes of he list “System?” Depending on setting this attribute in "Yes" or "No", there will be displayed a list of tables and datasets that belong, respectively, to the Configurator or the application under development.

In this case, the chain of the dependents select is composed of the upper node «System?» with level 0 and two nodes of the lower level "Table" and "dataset". In addition, the nodes "Table" and "dataset" are on the same level, i.e. level 1.

To view the configuration of the input form, double-click on the name of the form. Then go over to the tab with the window’s code «instant_edit». Then click on the icon "Element." A Grid with elements of the input form will be displayed. Get familiar with the attributes of fields list_system, list_table, and list_dataset by clicking in the Grid on the icon «Attribute» of the corresponding field. Pay particular attention to the attribute group "Attributes Ajax». Note that the field “list_system” has global scope under the name “system”.

Next it is necessary to get familiar with the list with system id “ListTable”, which is a list of tables. To do this, simply click on the name of the list for the field “list_table” in the Grid of elements of the input form. Configurator will proceed over to configuring the list on the tab “instant_edit”. This list is a variable list and it is based on the dataset with systemid “ListTable”. To view the “ListTable” dataset’s attributes, click on the name of the dataset in the Grid. The Configurator will start configurating the datasets limited by only dataset “ListTable”. In the Grid of datasets, click on the icon ”Detail” and see how two predicates of this dataset are configurated.

There should be an explanation of which lists are used to the Configurator and applications. All the variable lists based on tables with an attribute “System?” set to "Yes" and all costant lists, which are based on the table fgs_option are used for the Configurator. For applications, there are used all the variable lists based on the tables with the attribute “System?” set to "No" and all the constant lists, which are based on the table fgs_option.

To include all the constant lists, the predicate is reqiured for the field table_name with the attributes:

Label	Value	Comments
Table	fgs_table	
Field	table_name	
Argument	fgs_option	
Argument type		A type of the field “table_name” will be taken as a type of argument.
Operator	equal	
Connector	AND	It is not used, because it’s the first predicate to process: index is equal to 10

Required?	Yes	
Index	10	

To be included in a list of variable lists, a predicate is required for the field table_system with attributes:

Label	Value	Comments
Table	fgs_table	
Field	table_system	
Argument	&rgv system	This means that the argument is the element with the key “system”of the “globals” array of the Registry component
Argument type		A type of the field table_name will be taken as a type of argument.
Operator	equal	
Connector	OR	
Required?	yes	
Index	10	

Creating the list of tables with system id “ListTable” is done as follows.

Component Form puts the value of the “list_system” field in the element with the key “system”of the “globals” array of the Registry component. When loading the list with the system id “ListTable”, the loader of lists ListLoader uses the PredicateBuilder component for obtaining the conditions imposed by the “ListTable” dataset. The component PredicateBuilder, in turn, uses the Evaluator component to get the value «& rgv system» and returns the condition (table_name = 'fgs_option' or table_system = '0 ') or (table_name =' fgs_option 'or table_system = '1'), which ListLoader uses to load the list of tables with system id “ListTable”.

Creating the list of datasets with system id “UnitDataset” to enter the field list_dataset is carried out similarly. The “UnitDataset” list is based on the “UnitDataset” dataset, which has only one predicate with attributes:

Label	Value	Comments
Table	fgs_dataset	
Field	dataset_system	
Argument	&rgv system	This means that the argument is an element of the globals array of Registry component with a key “system”.
Argument type		A type of the field dataset_system will be taken as a type of an argument.
Operator	equal	
Connector	AND	
Required?	yes	

When loading the list with the system code “UnitDataset”, the ListLoader uses the PredicateBuilder component to obtain the conditions imposed by the “UnitDataset” dataset. The PredicateBuilder component, in turn, uses the Evaluator component to get the value «& rgv system» and returns the condition (dataset_system = '0 ') or (dataset_system = '1'), which uses ListLoader to download the datasets with system id UnitDataset.

Let’s consider how to configure a component RowAction with a condition. Chose again the menu item Specification->List. You can see that in the Grid of lists the lists based on the table have an icon fgs_option «Detail», and the lists, which are not based on the “fgs_option’ table, do not have this icon. To find out how it was configured, double-click on the name of the Grid. The controller will go over to the mode of configuring the Grid with the system id «lists» on the tab «instant_edit». Go over to this tab. Click on the icon "Column." There will be displayed a table of columns of this Grid. Click on the icon «Attribute» for the RowAction component of the action “detail”. There will be dispayed a form of input of attributes of the component. Please note the following attributes:

Label	Value
Condition	Equal
Condition field	list_table
Condition class	
Condition parameter	fgs_option
Condition parameter type	A scalar value

Class of a condition is not specified. This means that this is a standard condition.

Let's see how the condition “**Equal”** is configured. Choose the menu item Specification->Condition and find the condition with the system id “**Equal”**. Please note that type of this condition is "common condition". This condition consists of only one statement with the attributes

Label	Value	Commetaries
Operand 1	%statement_operand1	This means that Operand 1 will be replaced during exporting
Operator	equal	
Operand 2	%statement_operand2	This means that Operand 2 will be replaced during exporting
Failure	An error in case of failure	

During exporting of this condition
Operand 1 will be equal to «&arg list_table»
Operand 2 will be equal to « fgs_option»

The dispay of the icon “Detail” (component GridKit) and validation of appropriate action (a component of Grid) occurs as follows.

To validate the condition, ConditionTester component is used, to which both testable statements and the current row are passed. To evaluate Operand 1, the component ConditionTester uses the Evaluator component, to which the expression «& arg list_table» and the current row are passed. The Evaluator component returns the value of the field “list_table” to the component **ConditionTester**. Further the component **ConditionTester** compares to see if the resulting value equal to the value «fgs_option» (Operand 2) and returns the result of comparing. Depending on the result of validation, a decision is taken as to display or not to display the icon (GridKit) or to recognize the action as valid or not (Grid).

Lesson 5 - Configuring of Controllers

Configuring controllers is an example of an interface to the two tables related by master-detail (fgs_controller and fgs_unit) relationship. This example of configurating is interesting owing to the following. As you know, all controllers are divided on the basis of an attribute of their relationship to the Configurator or to the application under development. This division is determined on the base of the «System? » attribute of the controller. In accordance with this attribute, during configuration of a controller's Units the appropriate lists of input forms, Grids, search forms and datasets are used. In such a way, the lists of applied input forms, Grids, search forms and datasets are used for applicable controllers; while the lists of system input forms, Grids, search forms and datasets are used for controllers of the Configurator. Configuring the controllers shows how to pass a field value from the master table to the input forms of the subordinated table (detail Form).

To begin, let's choose the menu item Specification->Controller. A table of controllers will be displayed. Double-click on the name of the Grid and the Configuratorwill proceed to configuring the Grid with the system id «controllers» in the window with the code «instant_edit». Go over to this tab. Click on the icon "Column" and a list of columns of the Grid will be displayed. Click on the icon «Attribute» for the field «controller_system». A form of input of the attributes of this field will be displayed. Pay attention to the attribute "Global name", which is equal to «system», and the attribute "Save" set to "Yes."

Return the tab of configuring controllers and click on the icon «Detail» of any controller. An input form and the Grid of Units will be displayed. Click on the name of the input form, and the Configurator will proceed to configuring the input form with the system id "unit" in the window with code «instant_edit». Go over to this tab. Click on the icon "Element" for a single entry corresponding to this input form. A table of input elements, which relate to this input form, will be displayed. We shall be interested in configuring the input fields unit_form, unit_grid, unit_search and unit_dataset.

Let's consider configuring only the field unit_form because configuring other fields is similar to this. To enter this field, the list with the system id "UnitForm" is used, which is based on the dataset with the system id "UnitForm". To view the "UnitForm" list's attributes, click on the name of the "UnitForm" list. The Configurator will proceed to the configuring this list in the same window («instant_edit»). The "UnitForm" list excludes the types of search forms with assistance of the attribute 'where clause in the request" equal to "form_type not like' search% '". Click on the name of the "UnitForm" dataset. In the same window ("instant_edit") the Configurator will proceed to configuring this dataset. This dataset consists of five predicates.

As for the 4 predicates related to the field «form_type» of the table «fgs_form», it is necessary to do the following explanation. All input forms with type equal to «template», «column», «element», «predicate» are service forms and they are not used for entering data. Therefore, these four predicates cut off service forms. More interesting is the fifth predicate belonging to the field «form_system» of the table «fgs_form» and having the attributes:

Label	Value	Comments
Table	fgs_form	
Field	form_system	
Argument	&rgv system	This means that the argument is an element of the globals array

		of Registry component with a key “system”.
Argument type		A type of the field form_system will be taken as a type of an argument.
Operator	равно	
Connector	AND	
Required?	Yes	

Creating the “UnitForm” list occurs as follows.

To enter configurating Units, the developer should click on the icon «Detail» of a corresponding controller. At the same time, master Grid of the controller with the system id «controllers» should load only one row corresponding to the selected controller.

When loading only one row, the component Grid stores columns’ values with set attribute "Global name" into the “globals” array of the Registry component. In our case, this column is for the field controller_system.

When loading the list “ListTable”, the loader of lists ListLoader uses the PredicateBuilder component for getting the condition imposed by the “UnitForm” dataset. The PredicateBuilder component, in turn, uses the Evaluator component to get the value «& rgv system» and returns the condition ((form_system = '1 'and form_type! = 'template 'and form_type! ='column ' and form_type! =' element 'and form_type! = 'predicate')) or ((form_system = '0 'and form_type! = 'template 'and form_type! = 'column 'and form_type! = 'element 'and form_type! =' predicate ')), which the ListLoader adds to condition (form_type not like 'search%') and uses it to load the list of input forms with system id “UnitForm”.

Lesson 6 - Configuring input forms

Configuring input form is an example of configuring a UnitSet interface.
The particulariities of this configuring is using a dataset, a non-standard renderer of input forms and type of Units, differed from “master” and “detail”.

To begin, let’s get familiar with the controller with the system id “forms”, organizing this interface. To do this, choose the menu item Specification->Controller. In the displayed table of controllers locate the controller with the system id “forms” and click on the icon “Detail” of this controller.

Note that the “InputForm” dataset is included in the the master Unit. This is due to the fact that the specifications of input and search forms are stored in the same four tables. Therefore, the search forms should be exluded from the rows displayed in the master Grid. This is what for the “InputForm” dataset serves. To view the “InputForm” dataset, click on the name of the dataset InputForm in the row detail Grid of the corresponding Unit ID with the code master. The Configurator opens an additional tab with the code of a window «instant_edit» for configuring this particular data set. Click on this tab. This dataset has a simple predicate with attributes:

Label	Value	Comments
Table	fgs_form	
Field	form_type	
Argument	search	

Argument type		The field type form_type will be used as the type of an argument
Operator	Doesn’t contain (not like)	
Connector	AND	
Required?	Yes	

This predicate simply reflects the fact that all forms relating to the search forms contain in their type the substring «search».

Return to the tab for configuring controllers and click on the icon "Update" for the row of the detail Grid corresponding to the Unit with the system id “attribute”. This Unit organizes an interface for editing the attributes of input elements depending on the component of the input element. Pay attention that in the displayed input form of this Unit the value “AttributeView” is set as a renderer for the input form, while the value “Attribute” is set as a class of the Unit.

Pay attention also to the Unit with the code “add”, which has a type “auxiliary”. When configuring the input forms, this Unit organizes adding input fields. The Grid of this Unit has the system id “InitForm” and has a non-standard renderer FgsAddGridView. This Grid has the attribute «Modal?» set to "Yes." As you know, the usual components Form, Grid, and Search can be hidden and shown. However, when adding fields, this was deemed impractical. That's what for this attribute serves. The FgsAddGridView renderer does not include the possibility of setting by a user the outputting rows page-by-page, an arbitrary value of the attribute “offset”, pagesize.

Lesson 7 - Configuring menus

Configuring menus is an example of an interface to the two tables associated with master-detail (fgs_menu and fgs_item) relationship. In this example, what provokes interest is configuring “detail Grid” and one aspect of loading lists.

Chose the menu item Specification->Menu item. In the Grid of menus that appears find the menu with the system id “configurator” and click on the icon «Detail» of this menu. An input form and a Grid of options of this menu will be displayed. Menu options are a hierarchical tree.To configure the “detail Grid” double click on the name of the Grid and go over to the configuring “detail Grid”. On this tab, click on the icon "Column". A table of columns of this Grid will be displayed. Pay attention to the field item_pid, which stores the code of a parent node of the menu items. To display this field, component ColumnLookup with a list of MenuItem is used. This list is made up of menu items that are of type "menu". Click on the name of the list MenuItem. Configurator will go to configuring this list. This list is based on the same table fgs_item and, therefore, an alias table and alias fields are set for it. This list uses the dataset MenuItem to set conditions on the selected data. You can independently familiarize yourselves with the attributes of the predicate of this dataset. The most important thing here is that to load the data of this table, a list based on the same table is used. When exporting a specification of the input form, the Configurator resognizes this fact and adds the attribute “listDrop” to the properties of these elements. When creating an instance of the Form component with input elements with the same attribute, an attribute of the same name of the component Form is created. Component Unit, which includes component Form with the same attribute, registers for the list loader ListLoader the event «done» initiated by the Form component. When entering a new row, editing, or deleting a row, the Form component triggers the event «done», which is also sent to the list loader for processing. As arguments of the event, component

Form includes also the name of its base table. Processing this event by the list loader consists in reseting an attribute of loading the lists, which are based on the table passed in the arguments of events. Therefore, when trying to load such lists again, the list loader loads again the options of lists.

Lesson 8 - Configuring conditions

Configuring conditions is an example of an interface to the two tables associated with master-detail (fgs_condition and fgs_statement) relationship. In this example, what provokes interest is switching to entering of statements just after of inserting of a new condition.

An input form with id «condition» is used to input conditions. Usually the «Insert» button doesn't have the «event» attribute and the Configurator exports default value «done» for this attribute But in our case the «event» attribute is set to «detail». So after inserting of a new condition the Form component fires the «detail» event, after which occurs switching to «detail» mode. In this mode you can enter statements of the new condition.

This method is used for configuring controllers of input forms, grids and search forms. These controllers are of UnitSet type and the «event» attribute of «Insert» buttons of appropriate input forms is set to id of a necessary Unit. After inserting of a new input form, grid or search form the Form component fires the event of switching to necessary mode. For example after adding of a new input form occurs switching to «element» mode in which you can enter elements of the new form.

Lesson 9 - Configuring of an input elemnt's changing via an event

Let's suppose we have got an input form with an input element SelectOneMenu that uses a variable list from rows of a reference table. During editing or inserting rows may be happened that there is not an appropriate option in the list. So we need:

- Add the missing option, i.e. insert a row in the reference table
- Primary key of this row has to be assigned to the input element SelectOneMenu value

Of course, entered data into the input form under consideration has to be preserved.

To implement all of this it's nessecary:

The interface controller has to be of UnitSet type
The input form under consideration must have an element InputButton with the attribute «action» equal to "Fire event" and with not empty value of the attribute «event».
The interface controller has to include a Unit of type «auxiliary» with the only input form for the reference table. Id of this Unit has to be equal to the attribute «event» of the element InputButton
The attribute "Event" of button «Insert» of the «auxiliary» Unit's input form has to set
The attribute "Set value by event" of the input element SelectOneMenu must be equal to the attribute "Event" of button «Insert»