Справочник по разработке

Оглавление

[Секция панели инструментов формы 3](#_Toc499554556)

[Создание своей кнопки с обработкой JavaScript 3](#_Toc499554557)

[Создание своей кнопки с обработкой COS 4](#_Toc499554558)

[Секция поиска 5](#_Toc499554559)

[Поиск по строке 5](#_Toc499554560)

[Поиск по строке с квалификатором 5](#_Toc499554561)

[Поиск по комбобоксу 5](#_Toc499554562)

[Поиск по дате 6](#_Toc499554563)

[Поиск с вызовом формы 6](#_Toc499554564)

[Поиск по радиокнопке 6](#_Toc499554565)

[Поиск с использованием функции 6](#_Toc499554566)

[Поиск по комбобоксу с независимым списком значений 7](#_Toc499554567)

[Секция таблицы 8](#_Toc499554568)

[Строчное значение, основные свойства элемента секции 8](#_Toc499554569)

[Вычисляемое значение 8](#_Toc499554570)

[Значение комбобокса 8](#_Toc499554571)

[Значение даты 9](#_Toc499554572)

[Вывод древовидной структуры 9](#_Toc499554573)

[Мультиселект 9](#_Toc499554574)

[Стилизация столбцов 10](#_Toc499554575)

[Секция редактирования 11](#_Toc499554576)

[Свойство Readonly 11](#_Toc499554577)

[Свойство init поля формы 11](#_Toc499554578)

[Редактирование с вызовом формы 12](#_Toc499554579)

[Поле без сохранения 12](#_Toc499554580)

[Переопределение поля при выводе формы 12](#_Toc499554581)

[Секция выгрузки 15](#_Toc499554582)

[Секция загрузки 16](#_Toc499554583)

[Обработка событий компонентов формы 17](#_Toc499554584)

[Построение конструкций 19](#_Toc499554585)

[Реализация выполнения команд описания формы с участием сервера $$ 19](#_Toc499554586)

[Клиентская часть 19](#_Toc499554587)

[Серверная часть 21](#_Toc499554588)

[Реализация выполнения в фоне 23](#_Toc499554589)

[Создание компонента JavaScript 24](#_Toc499554590)

# Секция панели инструментов формы

## Создание своей кнопки с обработкой JavaScript

{

"type":"button",

"id":"SendEmail",

"image":"Mail.png",

"caption":"Уведомить по email",

"action":"script",

"hint":"Уведомить по email",

"script":"var numds=this.dataset.ComponentNum,seldata={};\nseldata.caption='Отправка EMail';\nseldata.width=500;\nseldata.height=280;\nseldata.action=function(){\n\n\tvar data={\n\t\tcaption:event.item.hint,\n\t\ttheme:$('#theme').val(),\n\t\tbody:$('#body').val()\n\t};\n\tformSelector.close();\n\taegDatasetsAll[numds].callserver(false,'toolbar',event.item.id,data);\n};\nseldata.items=[\n\t{name: 'theme', type: 'string', caption: 'Тема', init: '', placeholder : 'Введите тему письма', width : 385},\n\t{name: 'body', type: 'area', caption: 'Сообщение', init: '', placeholder : 'Введите сообщение', width : 383, height : 210}\n];\nShowSelector(seldata);"

}

**Скрипт показа формы с выпадающим списком, при нажатии на который показывается еще одна некая форма (например, как при копировании пунктов меню):**

// название свойства, возвращающее id выбранного объекта в data

var selectorSuffix = 'idclass';

// какую форму показать при нажатии на стрелку выбора списка

var formToShow = 'AppAGSClassList\_Find';

// значение какого поля из показанной формы показать в списке

var fieldNameShowedForm = 'gridNameClass';

var numds=this.dataset.ComponentNum,seldata={};

seldata.dataset=this.dataset;

seldata.caption='Выбор объекта для копирования свойств';

seldata.width=420;

seldata.height=70;

seldata.action=function(){

var ffcompM=aegEditfindformsAll[$($('input[id^=\u0022Selector-findform-'+selectorSuffix+'\u0022]')).attr('data-componentnum')],

data={caption : event.item.hint},

el=$('input[data-mode=\u0022data-true\u0022]');

data[selectorSuffix]=(ffcompM.findid[0]!=null)?ffcompM.findid[0].id:'';

data.nameform=$('#nameform').val();

for (var i=0,count=el.length; i<count; i++){

if (el[i].checked){data[el[i].name]=1;}

}

formSelector.close();

ffcompM.destroy();

aegDatasetsAll[numds].callserver(false,'copy',event.item.id,data);

}

seldata.items=[

{name:selectorSuffix,type:'findform',caption:'Копировать из',width:200,form:formToShow,tableshow:fieldNameShowedForm},

];

ShowSelector(seldata);

**Скрипт для кнопки, который открывает свою форму редактирования (например, как при редактировании настроек пользователя):**

var dataQuery=[

{

command : 'form',

tname : 'LibAppSetupEdit',

mode :'edit',

submode :'user'

idComponent : 'LibAppSetupEdit\_User\_Edit',

count : 1,

id : this.dataset.currentid,

parentdataset : this.dataset.ComponentNum

}

];

mainDataSource.GetServerData(dataQuery);

## Создание своей кнопки с обработкой COS

{

"type":"button",

"id":"DeleteSessions",

"image":"Stop.png",

"caption":"Завершить",

"action":"$$",

"hint":"Завершить сессию пользователей",

"confirm":"Завершить выбранные сессии?"

}

При этом в классе, который подкреплен к форме должен быть переопределен метод

/// Вызов функции с клиента  
/// На входе :  
/// in.form -> Имя формы  
/// in.filter -> фильтр для поиска  
/// in.typecall -> Источник вызова ("toolbar" - кнопка на панели задач и т.д.)  
/// in.namecall -> Имя действия (Для "toolbar" - имя кнопки и т.д.)  
/// in.actualid -> ID текущей строки в форме  
/// in.data -> Дополнительные данные для задачи (определяются клиентом)  
/// На выходе :  
/// out.error -> Сообщение об ошибке или пусто  
/// out.typejob -> Метод исполнения задачи ("fast" - задача исполняется быстро, не в фоне, "job" - задача исполняется медленно, в фоне)  
/// out.message -> Сообщение для клиента (выводится в модальном окне)  
/// out.jobnum -> Номер задания для фоновых задач  
ClassMethod CallClient(ByRef in, ByRef out)

{

…

}

# Секция поиска

Общий вид секции поиска

{

"name":"CLASSNAME-Findpanel",

"caption":"Панель поиска",

"type":"findpanel",

"section":[

{

"name":"LARUSUserFind",

"caption":"Пользователь",

"width":200,

"type":"string",

"findoption":{

"field":"Username",

"type":"содержит",

"data":""

}

}

]

}

## Поиск по строке

{

"name":"LARUSUserFind",

"caption":"Пользователь",

"width":200,

"type":"string",

"findoption":{

"field":"Username",

"type":["содержит",”равно”],

"data":""

}

}

## Поиск по строке с квалификатором

{

…

"type":"string",

"findoption":{

"field":"MaxResponseTime",

"type":"больше",

"data":""

},

"modifier":[

"больше",

"меньше",

"содержит",

"начинается",

"равно",

"пусто",

"не пусто",

]

}

## Поиск по комбобоксу

{

…

**"type":"combo",**

**"findoption":{**

**"field":"Event",**

**"type":"=",**

**"data":""**

**}**

}

В классе, который использует форма при этом должно быть:  
Property Event As Lib.DataTypes.String(DISPLAYLIST = ",Создание,Редактирование,Удаление", IndexValue = 1, VALUELIST = ",New,Edit,Delete");

## Поиск по дате

{

…

**"type":"date",**

**"findoption":{**

**"field":"DatePropertyName",**

**"type":"=",**

**"data":""**

**}**

}

## Поиск с вызовом формы

{

…

**"type":"findform", - тип поля**

**"findoption":{ - опции поиска в вызываемой форме:**

**"field":"idrole", - свойство класса вызываемой формы, по которому искать**

**"type":"равно",**

**"data":""**

**},**

**"form":"LibAppRole", - вызываемая форма**

**"tableshow":"LibAppRoleName", - Имя поля поисковой формы для показа в редакторе**

**"multiselect":false**

}

## Поиск по радиокнопке

{

…

**"type":"check",**

**"findoption":{**

**"field":"Active",**

**"type":"равно",**

**"data":""}**

}

## Поиск с использованием функции

Элемент секции поиска оформляется как обычно, а вот класс, который использует форма должен иметь переопределение метода **addFind**:

/// Поиск по функции  
ClassMethod addFind(ByRef in, ByRef out, ByRef ind)  
{  
// На входе :  
// in.formfield -> Имя поля формы  
// in.typefind -> Тип поиска ("равно", "не равно", "совпадение", "не совпадение", "содержит", "не содержит",   
// "начинается", "не начинается", "меньше", "больше", "пусто", "не пусто", "диапазон")  
// in.datafind -> Поисковый фрагмент  
// in.add.currentid -> id текущей записи в текущей таблице  
// in.add.editid -> Если поиск в поисковой форме из панели редактирования, то id редактируемой записи  
// in.add.parentid -> id-мастер записи  
// in.add.parentform -> Имя формы, откуда осуществляется вызов поисковой формы (если форма не поисковая, то пусто)  
// in.add.parentfield -> Имя поля формы, из которого осуществляется вызов поисковой формы (если форма не поисковая, то пусто)  
// На выходе :  
// out.error -> Сообщение об ошибке или пусто  
// ind -> Результирующий бит-индекс  
// Примеры :  
s err=""  
try {  
 s out.value=""  
 i (in.formfield="LibAppRuleRole") {  
 k indLinks,ptr,id  
 s filter = [{"field":"Role","type":"=","data":(in.datafind)}]  
 s err = ##class(Lib.App.RoleRules).Filter(filter, .indLinks) ; получение индекса всех связок роль-правило для роли (все формы)  
 i err'="" {throw ##class(%Exception.General).%New("<ERROR>",999,,err)}  
 while ..indNEXT(.indLinks,.ptr,.id){ ; пробегаем по отобранным связкам для роли  
 continue:'$ISOBJECT(##class(Lib.App.RoleRules).%OpenId(id).Rule) ; попускаем, если такого правила уже нет  
 s idRule=##class(Lib.App.RoleRules).%OpenId(id).Rule.%Id()   
 d ..indINSERT(.ind,idRule) ; собираем свой индекс с id правил  
 }  
 }  
   
} // try  
catch exp {  
 s out.error=..ErrorDetect(.exp)  
} // catch  
q

}

## Поиск по комбобоксу с независимым списком значений

Нужны определения свойств **type** и **selector**:

{

…

**"type":"combo",**

**"selector":[**

**{**

**"value":"blank",**

**"caption":"Пустая команда"**

**},**

**{**

**"value":"closeapp",**

**"caption":"Остановить приложение"**

**},**

**]**

}

# Секция таблицы

## Строчное значение, основные свойства элемента секции

{

"name":"LibAppRuleType",

"field":"Type",

"caption":"Тип правила",

"width":100,

"dalign":"left",

"visible":1,

"sortable":0,

}

## Вычисляемое значение

{

…

**"field":"$$",**

…

}

В классе при этом должен быть переопределен метод addCalcField

/// Получить вычислимое поле - место на дисках  
Method addCalcField(ByRef in, ByRef out)  
{

// На входе :  
// in.form -> Имя формы  
// in.section -> Тип секции ("findpanel" - панель поиска, "grid" - таблица, "editpanel" - панель редактирования)  
// in.field -> Имя поля  
// ..ИмяПоля -> Поле из тела записи  
// На выходе :  
// out.error -> Сообщение об ошибке или пусто  
// out.value -> Вычесленное значение  
  
s err=""  
try {  
 i ((in.field = "gridDisksInf") ! (in.field = "exportDisksInf")) {

 ;[{"name":"C","free":"3072","total":""},{"name":"D","free":"3072","total":""},...]  
 s disksInf={}.%FromJSON(..DisksInf)  
 s iterator=disksInf.%GetIterator() k key,disk  
 s out.value= out.value\_..GetLocaleCaption("Свободно на диске ")\_disk.name  
   
 }  
} // try  
catch exp {  
 s out.error = ..ErrorDetect(.exp)  
 s out.value = ..DisksInf  
} // catch  
 q  
}

## Значение комбобокса

{

…

**"type":"combo",**

…

}

2 варианта вывода значений:

1. В классе при этом у свойства должно быть задано свойство с DISPLAYLIST и VALUELIST: Property Type As Lib.DataTypes.String(DISPLAYLIST = ",Функция,Поле формы,Фильтр данных,Формула", IndexSpecWord = 1, IndexValue = 1, MAXLEN = 1024, VALUELIST = ",action,field,filter,formula");
2. В описании формы у свойства поля задается selector  
   "selector":[{"value":"blank", "caption":"Пустая команда"}, …]

## Значение даты, времени, даты и времени

Для даты:

{

…

**"type":"date",**

…

}

Для времени:

{

…

**"type":"time",**

…

}

Для даты и времени

{

…

**"type":"datetime",**

…

}

## Вывод древовидной структуры

В описании **секции** типа **grid** необходимо описать свойство **"loadtype":"tree"** или **"loadtype":"treeexpanded".** Значение tree означает «ленивую» загрузку – т.е. подгружаются только родительские элементы и элементы раскрытых узлов. Значение treeexpanded означает загрузку всех элементов дерева.

В описании класса, данные которого надо вывести в виде дерева должно появиться свойство **Property idparent** которое должно ссылаться на этот же класс, т.е. на самого себя

Пусть есть некий набор записей для вывода в виде дерева, тогда каждая запись record должна обладать следующими свойствами:

record.treeLeaf - признак терминального узла (1, лист дерева) или внутреннего узла (0)

record.treeParentID - ID родительского элемента

record.treeLevel - уровень вложенности выводимой записи, начинается с 0

record.treeExpanded - Открыт/Закрыт

record.treeLoaded - При ленивой загрузке Загружен/Не загружен

Метод вывода в составе входящих параметров получает следующие значения:

in.treelevel: 1, - уровень подгружаемых узлов

in.treeparent: 85, - ID раскрываемого родительского элемента, узлы которого надо подгрузить

in.treeall: false, - признак загрузки всего дерева

in.treeexpanded: [84, 85], -если treeall=false, то массив узлов, которые должны быть раскрыты, и наоборот

in.treeeexpandevent: 1 - Признак ленивой загрузки узла

## Мультиселект

Для отображения множественного выбора в параметрах формы указывается:

{"multiselectshow":true,"multiselectaction":["Reindex"]}

Где multiselectshow – переключатель показа множественного выбора при загрузке формы,

multiselectaction – массив доступных действий для множественного выбора

## Стилизация столбцов

{

**"style":"$$"**

}

Значением свойства может быть $$ или Id стиля из таблицы стилей. При значении $$ должен быть переопределен метод

Method addStyleCell(ByRef in, ByRef out)

**На входе :**

in.form -> Имя формы

in.type -> Тип ("grid" - таблица, "xlsx" - выгрузка в XLSX и т.д.)

in.field -> Имя поля формы  
..ИмяПоля -> Поле из тела записи

**На выходе :**

out.error -> Сообщение об ошибке или пусто

out.style -> Стиль ячейки в виде :

s out.style={"CodeStyle":(КодСтиля)} или

s out.style={"NameStyle":(ИмяСтиля)} или

s out.style={"Align":"left","BgColor":"#cc0000","Bold":1,"FontColor":"#ffffff","Italic":0}

# Секция редактирования

## Свойство Readonly

{

"name":"LibAppUserName",

"field":"Name",

"caption":"Логин",

"width":300,

"visible":1,

"required":1,

**"readonly":"edit"**

}

В качестве значения для свойства возможно установить:

0 – поле доступно для редактирования

1 – поле не доступно для редактирования

Type = [new, edit] – режим, при котором поле будет недоступно для редактирования

## Свойство init поля формы

Позволяет задать начальные значения при загрузке формы

{

…

**"init":"$$",**

**…**

}

В классе при этом должен быть переопрелен метод **addSetInit**:

/// Дополнительная функция для описания поля формы

ClassMethod addSetInit(ByRef in, ByRef out)  
{  
// На входе :  
// in.form - > Имя формы  
// in.namesection - > Имя секции  
// in.typesection - > Тип секции ("findpanel" - панель поиска, "editpanel" - панель редактирования)  
// in.field - > Имя поля  
// На выходе :  
// out.error -> Сообщение об ошибке или пусто  
// out.init -> Описание поля формы  
// Если строка или число, то отдаётся значение :  
// s out.init="Вася" или  
// s out.init=45  
// Если комбо, то отдаётся ID строки списка  
// для Property PeriodUse As Lib.DataTypes.String(DISPLAYLIST = ",Часто,Редко", MAXLEN = 1024, VALUELIST = ",B,N") [ Required ];  
// s out.init="B"  
// Если тип поля "поисковая форма", то возвращает объект с ID и Названием  
// s name="Штука",id=123  
// s out.init={"id":id,"value":name}

s err=""

try {

s out.init="",out.error=""

;-- Заявитель

if in.field="editPersonOwner" {

d ##class(Lib.App.User).GetIdByName(,.idUser) ; Получить id-User  
 s user=##class(Lib.App.User).%OpenId(idUser) ; Открыть его

s out.init={"id":(idUser),"value":(user.FIO)}}      ; отдаем id-User и нужное поле юзера

} // try  
catch exp {  
 s out.error=..ErrorDetect(.exp)  
} // catch  
q

}

## Редактирование с вызовом формы

{

…

**"form":"DesignClasses",** имя вызываемой формы

**"tableshow":"ClassName",** имя поля вызываемой формы, значение которого нужно в искомой форме

**"tablefind":"Name",** имена полей вызываемой формы, по которым будет поиск при ручном вводе с клавиатуры (на будущее)

"multiselect":0

}

## Поле без сохранения

{

…

**"saved": false,**

…

}

## Переопределение поля при выводе формы

{

"name":"LibAppSetupValue",

"field":"$$",

"caption":"Значение",

"width":500,

"dalign":"left",

"visible":1,

**"deffield":"init",**

"saved":false

}

В классе при этом должен быть переопрелен метод **addDefField:**

// Дополнительная функция для описания поля формы  
ClassMethod addDefField(ByRef in, ByRef out)  
{  
// На входе :  
// in.form - > Имя формы  
// in.namesection - > Имя секции  
// in.typesection - > Тип секции ("findpanel" - панель поиска, "grid" - таблица, "editpanel" - панель редактирования)  
// in.field - > Имя поля  
// in.mode - > Режим "edit" - редактирование, "new" - новая запись  
// in.defmode - > Подрежим : "init" - первоначальная загрузка, "save" - загрузка после сохранения, "change" - изменение связного поля  
// in.count - > Номер записи в секции, нумерация с нуля  
// in.idrec - > ID редатируемой записи  
// На выходе :  
// out.error -> Сообщение об ошибке или пусто  
// out.def -> Описание поля формы  
 s err=""  
 try {  
 s out.def={},out.error=""  
 if (in.mode="edit"){  
 if in.defmode="init" {  
 // Отдаю структуру поля  
 if in.valuename'="" {s name=in.valuename} else {s name=..%OpenId(in.idrec).Name}  
 s out.def=$case(name,  
 // Интерфейс приложения  
 "interfaceType" : {"type":"combo", "selector" : [{"value":"classic","caption":"Классический"},{"value":"vertical","caption":"Вертикальный"},{"value":"touchleft","caption":"Сенсорный для левшей"},{"value":"auto","caption":"Выбирается автоматически"}]}, // ,{"value":"touch","caption":"Сенсорный"}  
 // Модель приложения  
 "modelType" : {"type":"combo", "selector" : [{"value":"tabs","caption":"Закладки"},{"value":"windows","caption":"Окна"}]},  
 // Статус окна  
 "windowStatus" : {"type":"combo", "selector" : [{"value":"maximized","caption":"Распахнуто"},{"value":"normalized","caption":"Не распахнуто"}]},  
 // Подписи кнопок панели инструментов  
 "labelButton" : {"type":"check", "selector" : [{"value":"1","caption":"Есть подписи"},{"value":"0","caption":"Нет подписей"}]},  
 // Количество загружаемых в таблицу строк  
 "rowCount" : {"type":"integer", "width":100},  
 // Список количеств строк в таблице  
 "rowList" : {"type":"string"},  
 // Время показа сообщений разработчику  
 "msgDevelop" : {"type":"integer", "width":100},  
 // Показ ID в таблице  
 "showId" : {"type":"check"},  
 // Цветовая схема  
 "theme" : {"type":"combo", "selector" : ($case(in.reg,"fast":[],:..GetListTheme()))},  
 // Кнопки панели инструментов  
 "themeButtons" : {"type":"combo", "selector" : ($case(in.reg,"fast":[],:..GetListTheme("buttons")))},  
 // Локализация  
 "locale" : {"type":"combo", "selector" : [{"value":"ru","caption":"Русский"},{"value":"en","caption":"Английский (English)"},{"value":"cn","caption":",Китайский (中國)"}]},  
 // Тип компоненты вывода прогресса  
 "progresstype" : {"type":"combo", "selector" : [{"value":"image","caption":"Картинка"},{"value":"progress","caption":"Индикатор загрузки"}]},  
 // Количество месяцев в календаре  
 "calendarnumberOfMonths" : {"type":"integer", "width":100},  
 // Неделя в календаре  
 "calendarshowWeek" : {"type":"check"},  
 // Меню приложения  
 "menu" : {"type" : "findform", "form":"LibAppMenu","tableshow":"LibAppMenuCaption","class":"Lib.App.Menu","field":"Caption","filter":[{"field" : "idparent", "type" : "пусто", "data" : ""}]},  
 // Размер шрифта  
 "fontsize" : {"type":"integer", "width":100},  
 // Запуск редактора записи по клику на выбранной строке  
 "clickedit" : {"type":"check"},  
 // Заголовок  
 "CaptionForm" : {},  
 // Список загрузки/выгрузки  
 "exportList" : {"type" : "json", "height":400},  
 "importList" : {"type" : "json", "height":400},  
 : {"width" : "500", "dalign":"left", "visible":true}  
 )  
 } // if in.defmode="init"  
 } // if in.mode="edit"  
 } // try  
 catch exp {  
 s out.error=..ErrorDetect(.exp)  
 } // catch  
 q  
}

# Секция выгрузки

# Секция загрузки

# Обработка событий компонентов формы

*На примере компонента grid*

В JSON-описании секции может присутствовать объект описания “events”:

{

…

**"events": {**

**"onSelectRow":"$$"**

**},**

"section":[

{

"name":"gridNum",

"field":"Num",

"caption":"№ заявки",

"width":60,

"dalign":"right",

"visible":1,

"sortable":0

},

…

]

}

Названия свойств этого объекта – это события, которые возможны на данном типе секции, а значение свойств – команды, которые будут выполнены при наступлении события. Например, “onSelectRow” – событие выбора строки таблицы. Список возможных событий зависит от компонента (компонент выбора даты и компонент таблицы, отображающей данные класса, имеют разный набор событий). Обработка событий дополняется по мере необходимости, так как охватить сразу все множество событий всех компонентов невозможно при текущем ресурсе, да и нецелесообразно.

Значением свойства может быть либо JavaScript-инструкции, которые будут выполнены при наступлении события, либо значение “$$”, которое влечет за собой запрос на сервер набора данных, необходимых для редактирования формы. Для получения этого набора должен быть переопределен метод класса addEvent() в том классе, к которому прикреплена форма, создавшая событие (касается и конструкций, где мастер-форма и ракурсы также прикреплены к своим конкретным классам).

**Запрос**, который клиент посылает серверу для получения набора данных, по которому клиент поймет с каким компонентом и что сделать, состоит из:

in = {...} - входные параметры для получения инструкций для манипуляций с формой  
in.event -> событие, которое вызвало обработку  
in.contruct -> имя конструкции  
in.form -> имя формы  
in.type -> тип секции, из которой пришло событие: toolbar, grid,...   
in.idrecord -> id записи  
in.idparent -> id родительской записи

**Ответ**, который генерирует сервер для клиента, состоит из:

out = {“commands”:[]...} – свойство commands содержит массив объектов-комманд, каждый объект содержит информацию с каким элементом какой секции что сделать.

out.commands = [{…},…] -> массив объектов каждый, состоит из

{“form” -> имя формы

“type” -> тип секции, элемент которой нужно обработать: toolbar, grid,... “element” -> id элемента секции

“action” -> действие: enable, disable,...

“data” -> JSON, TODO, задел на будущее, например, новые данные для списка

}

out.error -> ошибки

Сигнатура метода для переопределения:

ClassMethod addEvent(ByRef in, ByRef out)

{

…

}

# Построение конструкций из форм

Конструкция всегда состоит из двух и более форм, каждая из которых создается обычным способом. Одна форма – главная, остальные – ракурсы, т.е. подчиненные формы. В дизайнере конструкций они объединяются при помощи выражения JSON. Например, у нас есть класс AppRQSMtrLine (дочерняя таблица), который связан с классом AppRQSMtr (головная таблица) по полю

Property idRqsMtr As App.RQS.Mtr;

То при создании конструкции с именем AppRQS объединяющее JSON выражение примет вид:

[{

"caption": "Краткий отчет по Заказчику",

"filter": "[{\u0022**field**\u0022:\u0022**idRqsMtr**\u0022,\u0022**type**\u0022:\u0022=\u0022,\u0022**data**\u0022:\u0022**%parentid%**\u0022}]",

**"rakurs": "AppRQSMtrLine",**

**"fieldparent": "idRqsMtr",**

"type": "table",

"parameter": ""

}]

Где каждый объект массива – ракурс конструкции. В примере приведен только один ракурс.

# Реализация выполнения команд описания формы с участием сервера $$

На примере обработки события выбора строки таблицы и влияния этого события на панель инструментов.

**Задача:** Изменять состояние кнопок панели управления в зависимости от статуса выбранной строки таблицы.

**Решение:** в описании секции типа «Таблица» grid есть свойство "events":{ "onSelectRow":"$$"}

Общий вид последовательности отработки частей приложения:

|  |  |  |
| --- | --- | --- |
| Клиент |  | Сервер |
| grid.js |  |  |
| ↓ |  |  |
| dataset.js |  |  |
| ↓ |  |  |
| datasource.js | **→** | ##class(Lib.AJAX.Abstract).OnPage() |
|  |  | **↓** |
| datasource.js | **←** | … |
| ↓ |  |  |
| dataset.js |  |  |
| ↓ |  |  |
| form.js |  |  |
| ↓ |  |  |
| [toolbar.js, grid.js, …] |  |  |

## Клиентская часть

1. Дополнение скрипта компонента «Таблица» **grid.js**  
   Дополняем определение gridSetup, свойство onSelectRow:  
   if ((this.data.events!=null)&&(this.data.events.onSelectRow!=null)&&(this.data.events.onSelectRow!="")) {  
    var text=this.data.events.onSelectRow;  
    if (text=="$$") { // Вызов Cache  
    **this.dataset.CallEvent(false,this,"SelectRow");**  
    } // if text  
    else { // Скрипт  
    text=text.replace(/\\n/g, "\n").replace(/\\t/g, "\t").replace(/\\u0022/g, '"');  
    eval(text); // Выполнить какой-то скрипт  
    } // else  
   } // if
2. Дополнение скрипта компонента «Набор данных» **dataset.js**Создаем функцию **CallEvent**  
   *// CallEvent(this,"SelectRow");  
   // Вызов обработки события на сервере приложений  
   // Если reg=true, то это ответ с сервера  
   // ... и в data - ответ с сервера  
   // Если reg=false, то запуск функции  
   // ... и в element - объект события  
   // ... и в event - название события*aegDataset.prototype.**CallEvent**=function(reg,element,event,data){  
    **if (reg) { // Ответ с сервера**  
    // перебор массива действий  
    for (i in data.commands) {  
    var command = data.commands[i];  
    var formId = findFormByName(command.form);  
    if (formId){aegFormsAll[formId].CallEvent(command);} // Передать в форму  
    } // for  
    } // if reg  
    **else { // Запуск**  
    var query=[],formTemp=aegFormsAll[this.form],  
    nameConst="",  
    idparent="";  
    if (formTemp.DataForm.place=="main") { // Главная форма  
    nameConst=formTemp.DataConst.name;  
    } // if  
    else if (formTemp.DataForm.place=="detail") { // Ракурс  
    var numMainForm=aegFormsAll[this.form].DataForm.mainForm;  
    nameConst=aegFormsAll[numMainForm].DataConst.name;  
    idparent=aegFormsAll[numMainForm].dataset.currentid;  
    } // else if  
    query.push({  
    command : "event", // Тип запроса  
    idComponent : formTemp.DataForm.idComponent, // Id формы  
    event : event, // Имя события  
    construct : nameConst, // Имя конструкции  
    form : formTemp.DataForm.name, // Имя формы  
    type : element.data.type, // Тип секции  
    idrecord : this.currentid, // ID записи  
    idparent : idparent // ID записи родителя  
    }); // query.push  
    mainDataSource.GetServerData(query);  
    } // else  
    return true;  
   }; // callserver
3. Дополнение скрипта компонента «Источник данных для всего проекта» **datasource.js**  
   Это единое место всех вызовов на сервер  
   Добавляем Case успешного выполнения инструкцией  
   case "event" : // События  
    var formId = findForm(data[i].data.idComponent);  
    if (formId){aegFormsAll[formId].dataset.CallEvent(true,"","",data[i].data);} // Передать в набор данных  
   break;
4. Дополнение скрипта формы **form.js**  
   // Вызов обработки события на сервере приложений

aegForm.prototype.CallEvent=function(data){  
 switch (data.type) {  
 case "toolbar" : this.toolbar.CallEvent(data); break; // Панель Инструментов case "toolbaredit" : break; // Панель Инструментов  
 case "findpanel" : break; // Панель Поиска

case "starterpanel" : break; // Панель Стартёра отчёта

case "grid" :break; // Таблица

case "editpanel" : break; // Панель редактирования  
 } // switch  
 return true;  
}; // CallEvent  
Форма передаст управление нужному компоненту, в данном случае toolbar-у, так как по условиям задачи требуется изменять состояние его кнопок.

## Серверная часть

1. Добавление в ##class(Lib.AJAX.Abstract) обработки при наличии свойства «events» в описании компонента формы  
   В блоке обработки команд добавляем:  
   "event":..eventCommand(.value,.out2), // Вызов обработчика события с сервера  
   и пишем соответствующий метод:  
   /// Вызов клиентом обработчика события с сервера  
   ClassMethod eventCommand(ByRef in, ByRef out) As %String [ ServerOnly = 1 ]  
   { s err = ""  
    try {  
    // Получить из формы имя класса  
    s err=##class(Lib.App.Form).GetIdByName(in.form,.idForm)  
    q:err'=""   
    s obj=##class(Lib.App.Form).%OpenId(idForm),cls=obj.Table  
    s out={},out2={}  
    s err=$CLASSMETHOD(cls,**"EventClient"**,.in,.out2),out.error=err  
    s out.data=out2  
    }  
    catch exp {  
    s err=..ErrorDetect(.exp),out.error=err  
    }  
    q err  
   }

Пишем метод **EventClient**  в классе ##class(Lib.Abstract.Tables), который будет вызывать метод-болванку **addEvent**. Метод addEvent и будет переопределяться от класса к классу.  
/// Вызов обработки события на клиенте с сервера  
/// Вход: in = {} - входные параметры для получения инструкций для манипуляций с формой</br>  
///  in.event -> событие, которое вызвало обработку </br>  
///  in.construct-> имя конструкции </br>  
///  in.form -> имя формы </br>  
///  in.type -> тип секции: toolbar, grid,... </br>  
///  in.idrecord -> id записи </br>  
///  in.idparent -> id родительской записи </br>   
/// Выход: out = {} - содержит с каким элементом какой секции что сделать  
///  out.form -> имя формы </br>  
///  out.type -> тип секции: toolbar, grid,... </br>  
///  out.element -> id элемента секции </br>  
///  out.action -> действие: enable, disable,... </br>  
///  out.data -> JSON, TODO, задел на будущее, например, новые данные для списка</br>  
///  out.error -> ошибки</br>  
ClassMethod EventClient(ByRef in, ByRef out)  
{  
 s err = ""  
 try {  
 s out.idComponent = in.idComponent   
 d ..addEvent(.in,.out)  
 }  
 catch exp {  
 s err=..ErrorDetect(.exp),out.error=err  
 }  
 q err  
}

/// Вызов обработки события на клиенте с сервера </br>  
/// Вход: in = {} - входные параметры для получения инструкций для манипуляций с формой</br>  
///  in.event -> событие, которое вызвало обработку </br>  
///  in.construct-> имя конструкции </br>  
///  in.form -> имя формы </br>  
///  in.type -> тип секции: toolbar, grid,... </br>  
///  in.idrecord -> id записи </br>  
///  in.idparent -> id родительской записи </br>   
/// Выход: out = {} - содержит с каким элементом какой секции что сделать  
///  out.form -> имя формы </br>  
///  out.type -> тип секции: toolbar, grid,... </br>  
///  out.element -> id элемента секции </br>  
///  out.action -> действие: enable, disable,... </br>  
///  out.data -> JSON, TODO, задел на будущее, например, новые данные для списка</br>  
///  out.error -> ошибки</br>  
ClassMethod addEvent(ByRef in, ByRef out)  
{  
 s err = ""  
 try {  
 // !!! Здесь пишем свой код  
 s out.form = in.form  
 s out.type = in.type  
 }  
 catch exp {  
 s err=..ErrorDetect(.exp),out.error=err  
 }  
 q  
}

# Реализация выполнения в фоне

Выгрузка динамического количества столбцов

# Создание компонента JavaScript

Форма на клиенте – это контейнер, который в свою очередь тоже содержит различные контейнеры. Например, форма редактирования контрагента МДМ состоит из контейнеров editPanel и toolbarEdit, поэтому при создании нового компонента, например, editDate , который будет выводить датапикер для поля ввода даты, мы должны добавить сам код компонента и его обработку для контейнеров, которые потенциально могут содержать editDate – editPanel и findPanel.

**Последовательность обработки компонета:**

Конструкция -> Форма -> editPanel [ findpanel, startpanel ] -> Компонент

Панели – это элементы более высокого уровня, чем компоненты. Панели сначала создают компоненты (CreateContent), а затем инициализируют их (Init).