**Общие правила доработки конфигурации.**

0. Весь код должен быть закомментирован (Первый Бит - префикс "бит", программисты клиента - префикс "ит"). Все добавленные объекты (объекты конфигурации верхнего уровня, объекты конфигурации нижних уровней, если объект типовой или добавлялся другой стороной, процедуры и функции в типовых объектах, объектах добавленных другой стороной) тоже должны иметь префикс. Если меняем свойства объектов, то пишем в комментарий, в поле «Комментарий» объекта.

Примеры:

//+бит.Фамилия.Дата

//Здесь наш прекрасный код…

//И его очень много

//-бит.Фамилия.Дата

//+бит.Фамилия.Дата

ОднаОченьОченьОченьОченьДлиннаяСтрочкаКодаКотораяВЭкранНепомещаетсяДаже();

//-бит.Фамилия.Дата

ОднаСтрочкаКода(); //+бит.Фамилия.Дата

//+ит.Фамилия.Дата

//Здесь наш прекрасный код…

//И его тоже очень много

//-ит.Фамилия.Дата

//+ит.Фамилия.Дата

ОднаОченьОченьОченьОченьДлиннаяСтрочкаКодаКотораяВЭкранНепомещаетсяДаже();

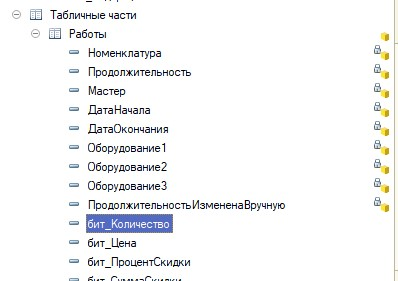
//-ит.Фамилия.Дата

ОднаСтрочкаКода(); //+ит.Фамилия.Дата

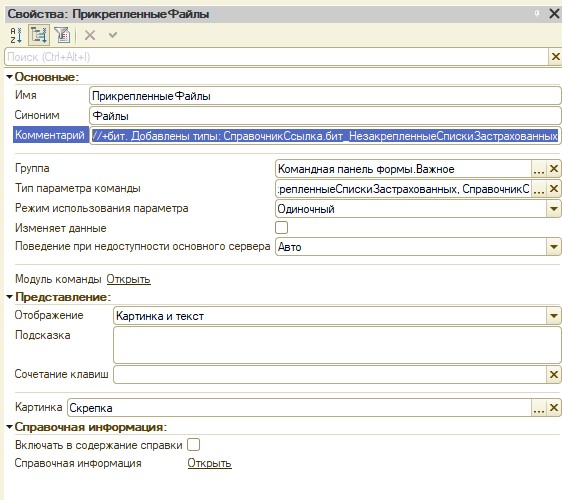
Добавленный справочник:



Добавленные реквизиты в типовые объекты:



Измененное свойство типового объекта:



1. Изменения на форме строго кодом через общие модули бит\_События

1.1. Все обработчики событий формы изменяем так же через бит\_События, т.е. в форме создаем обработчик, кодом его подключаем, а содержимое выносим в общие модули.

1.2 Если процедура используется и при событии формы и при событии элементов формы, то прописываем процедуры в модуле этой же формы и делаем экспортной, когда она вызывается для событий элементов формы то вызываем ее внутри модуля формы. Когда ее нужно вызвать для события формы, то вызываем ее в общем модуле бит\_События как ЭтаФорма.ИмяПроцедуры

Разъяснения:

Есть 3 общих модуля:

1. бит\_СобытияКлиент – содержит обработчики клиентских событий форм

2. бит\_СобытияВызовСервера – содержит обработчики серверных событий форм

3. бит\_СобытияКлиентСерверПовтИсп – содержит служебные функции для определения событий формы по имени формы

Эти функции используются для определения обработчика формы, переопределяются только в случае добавления обработчика формы, который еще никто не использовал.

бит\_СобытияКлиентСерверПовтИсп.ПолучитьОбработчикСобытияФормыСервер(ИмяСобытия, ИмяФормы)

бит\_СобытияКлиентСерверПовтИсп.ПолучитьОбработчикСобытияФормыКлиент(ИмяСобытия, ИмяФормы)

Функции по такому шаблону используются для сопоставления формы и функции обработчика

бит\_СобытияКлиентСерверПовтИсп.ПолучитьОбработчики[ИмяОбработчика]()

Функции по такому шаблону используются для непосредственного вызова в обработчике форм **(только они должны создаваться экспортными в модулях бит\_События).**

бит\_СобытияКлиент.[ИмяОбработчика](ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], ДополнительныеПараметры = Неопределено)

бит\_СобытияВызовСервера.[ИмяОбработчика](ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], ДополнительныеПараметры = Неопределено)

Примеры использования:

Самый длинный путь, если обработчик, который вы собираетесь править вообще ни на одной форме еще не дорабатывался. Собственно, если что-то уже есть, то шагов меньше:

1. Пишем в форме в обработчике:

[бит\_СобытияКлиент|бит\_СобытияВызовСервера].[ИмяОбработчика]( ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], [ДополнительныеПараметры])

2. Добавляем в общем модуле [бит\_СобытияКлиент|бит\_СобытияВызовСервера] процедуру с именем обработчика.

Процедура [ИмяОбработчика](ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], ДополнительныеПараметры = Неопределено) Экспорт

ИмяФормы = ЭтаФорма.ИмяФормы;

ИмяСобытияФормы = "[ИмяСобытия]";

ПараметрыОбработчика = "ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], ДополнительныеПараметры";

Обработчик = бит\_СобытияКлиентСерверПовтИсп.ПолучитьОбработчикСобытияФормы[Клиент|Сервер](ИмяСобытияФормы, ИмяФормы);

Если Обработчик <> Неопределено Тогда

Выполнить(Обработчик + "(" + ПараметрыОбработчика + ")");

КонецЕсли;

КонецПроцедуры

3. Добавляем в функцию: бит\_СобытияКлиентСерверПовтИсп.ПолучитьОбработчикСобытияФормы[Сервер|Клиент](ИмяСобытия, ИмяФормы) элемент соответствия с нашим обработчиком:

Например:

СоответствиеСобытий.Вставить("[ИмяСобытия]", ПолучитьОбработчики[ИмяСобытия]());

4. Добавляем функцию:

[&НаКлиенте|&НаСервере]

Функция ПолучитьОбработчики[ИмяОбработчика]()

мсФорм = Новый Массив;

мсФорм.Добавить([ПолноеИмяФормы]); //Пример Справочник.Клиенты.Форма.ФормаЭлементаУпр

Шаблон = "[ИмяОбработчика]\_%1";

соотСобытия = СформироватьСоответствиеПоМассивуФорм(Шаблон, мсФорм);

Возврат соотСобытия;

КонецФункции

5. Добавляем процедуру в общий модуль [бит\_СобытияКлиент|бит\_СобытияВызовСервера]:

Процедура [ИмяПроцедурыСобытия] (ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], ДополнительныеПараметры = Неопределено)

//Здесь наш прекрасный код

КонецПроцедуры

Общий шаблон именования таких процедур:

[ИмяСобытия]\_[ОбъектМетаданных]\_[ИмяФормы](ЭтаФорма, [ПараметрОбработчика1], … [ПараметрОбработчикаN], ДополнительныеПараметры = Неопределено)

Например:

ПриСозданииНаСервере\_СправочникКлиенты\_ФормаЭлементаУпр(ЭтаФорма, Отказ, СтандартнаяОбработка, ДополнительныеПараметры = Неопределено)

2. Если есть необходимость что-то сделать предопределенным, то делаем предопределенный в справочнике "бит\_НастройкиУчета" и там делаем ссылку на предопределенный элемент.

Разъяснение:

1. Создаем Предопределенный элемент в справочнике бит\_НастройкиУчета.

2. На форме этого справочника ФормаОбщая, создаем реквизит формы с нужным типом.

3. В функции ПолучитьСписокПростыхНастроек, добавляем соответствие Предопределенного имени и Имени реквизита формы.

4. В дальнейшем получение такой настройки учета необходимо делать через функцию:

бит\_ОбщегоНазначенияПривилегированный.ПолучитьПростуюНастройкуУчета(ИмяПредопределенногоЭлемента)

3. Изменения обработчиков событий по максимуму должно идти в подписки на события, кроме случаев, когда надо вклиниться в типовой алгоритм.

3.1. Подписки на события необходимо создавать в общем модуле бит\_ПодпискиНаСобытия.

3.2. Подписки рекомендуется создавать по одной на каждый документ на каждое событие, например, СправочникиПриЗаписи, ДокументыОбработкаПроведения. Тип объекта устанавливается как СправочникОбъект, ДокументОбъект и т.п. В обработчике такой подписке уже разносим на отдельные процедуры обработчики, в зависимости от типа.

Примеры:

Подписка ДокументыПередЗаписью. Тип объекта: ДокументОбъект, Событие: ПередЗаписью.

#Область ШлюзПодписок\_Документы

Процедура ДокументыПередЗаписью(Источник, Отказ, РежимЗаписи, РежимПроведения) Экспорт

ТипОбъекта = ТипЗнч(Источник);

Если ТипОбъекта = Тип(“ДокументОбъект.ОказаниеУслуг”) Тогда

ДокументОказаниеУслуг\_ПередЗаписью(Источник, Отказ, РежимЗаписи, РежимПроведения);

ИначеЕсли ТипОбъекта = Тип(“ДокументОбъект.Заявка”) Тогда

ДокументЗаявка\_ПередЗаписью(Источник, Отказ, РежимЗаписи, РежимПроведения);

КонецЕсли;

КонецПроцедуры

#КонецОбласти

#Область ПодпискиНаСобытия\_Документы

#Область Документы\_ПередЗаписью

Процедура ДокументОказаниеУслуг\_ПередЗаписью(Источник, Отказ, РежимЗаписи, РежимПроведения) Экспорт

//Мой прекрасный код

КонецПроцедуры

Процедура ДокументЗаявка\_ПередЗаписью(Источник, Отказ, РежимЗаписи, РежимПроведения) Экспорт

//Мой прекрасный код

КонецПроцедуры

#КонецОбласти

#КонецОбласти

4. Изменения типовых ролей и подсистем **только** через расширение.

5. При разработке новых форм (для своих объектов или типовых объектов) использовать следующий шаблон областей формы:

#Область ОписаниеПеременных

#КонецОбласти

#Область ОбработчикиСобытийФормы

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиСобытийЭлементовШапкиФормы

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиСобытийЭлементовТаблицыФормы<ИмяТаблицыФормы>

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиКомандФормы

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1. Раздел «Обработчики событий формы» содержит процедуры-обработчики событий формы: ПриСозданииНаСервере, ПриОткрытии и т.п.
2. Раздел «Обработчики событий элементов шапки формы» содержит процедуры-обработчики элементов, расположенных в основной части формы (все, что не связано с таблицами на форме).
3. В разделах «Обработчики событий элементов таблицы формы <имя таблицы формы>» размещаются процедуры-обработчики таблиц формы и элементов таблиц. Для процедур-обработчиков каждой таблицы должен быть создан свой раздел.
4. Раздел «Обработчики команд формы» содержит процедуры-обработчики команд формы (имена которых задаются в свойстве Действие команд формы).
5. Раздел «Служебные процедуры и функции» имеет такое же предназначение, что и в общих модулях.

6. При разработке новых общих модулей использовать следующий шаблон областей:

#Область ПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область СлужебныйПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1. Раздел «Программный интерфейс» содержит экспортные процедуры и функции, предназначенные для использования другими объектами конфигурации или другими программами (например, через внешнее соединение).
2. Раздел «Служебный программный интерфейс» предназначен для модулей, которые являются частью некоторой функциональной подсистемы. В нем должны быть размещены экспортные процедуры и функции, которые допустимо вызывать только из других функциональных подсистем этой же библиотеки.
3. Раздел «Служебные процедуры и функции» содержит процедуры и функции, составляющие внутреннюю реализацию общего модуля. В тех случаях, когда общий модуль является частью некоторой функциональной подсистемы, включающей в себя несколько объектов метаданных, в этом разделе также могут быть размещены служебные экспортные процедуры и функции, предназначенные только для вызова из других объектов данной подсистемы.

7. При разработке своих объектов конфигурации использовать следующий шаблон для модулей объекта, менеджера, набора записей и т.д.

#Область ОписаниеПеременных

#КонецОбласти

#Область ПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область ОбработчикиСобытий

// Код процедур и функций

#КонецОбласти

#Область СлужебныйПрограммныйИнтерфейс

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

#Область Инициализация

#КонецОбласти

1. Раздел «Программный интерфейс» содержит экспортные процедуры и функции, предназначенные для использования в других модулях конфигурации или другими программами (например, через внешнее соединение). Не следует в этот раздел помещать экспортные функции и процедуры, которые предназначены для вызова исключительно из модулей самого объекта, его форм и команд. Например, процедуры заполнения табличной части документа, которые вызываются из обработки заполнения в модуле объекта и из формы документа в обработчике команды формы не являются программным интерфейсом модуля объекта, т.к. вызываются только в самом модуле и из форм этого же объекта. Их следует размещать в разделе «Служебные процедуры и функции».
2. Раздел «Обработчики событий» содержит обработчики событий модуля объекта (ПриЗаписи, ПриПроведении и др.)
3. Раздел «Служебный программный интерфейс» имеет такое же предназначение, как и в общих модулях.
4. Раздел «Служебные процедуры и функции» имеет такое же предназначение, как и в общих модулях.

8. При разработке своих общих команд, команд объекта использовать следующий шаблон областей:

#Область ОбработчикиСобытий

// Код процедур и функций

#КонецОбласти

#Область СлужебныеПроцедурыИФункции

// Код процедур и функций

#КонецОбласти

1. Раздел «Обработчики событий» содержит процедуру-обработчик команды ОбработкаКоманды.
2. Раздел «Служебные процедуры и функции» имеет такое же предназначение, что и в общих модулях.

9. При создании функций/процедур в типовых объектах создавать эти функции/процедуры в области [Префикс]\_Доработки (например, БИТ\_Доработки, ИТ\_Доработки)

10. Если нужно вывести на форму (куда-то не в командный интерфейс, где они сами появляются) общую команду (или команду объекта), возникает проблема что нельзя указать кнопке ИмяКоманды, потому что его нет, и Экспортной функцию из общей команды не сделаешь, поэтому, если такая необходимость возникла, то делаем следующим образом:

В общем модуле: бит\_КомандыКлиент создаем экспортную процедуру с аналогичным названием (как команда называется), и вставляем в нее код команды из модуля команды.

В модуле самой команды вызываем эту процедуру и делаем после неё возврат.

Обязательно пишем комментарий для обновлятора: // КОД ЭТОЙ ОБЩЕЙ КОМАНДЫ ВЫНЕСЕН В ОБЩИЙ МОДУЛЬ, ПРИ ОБНОВЛЕНИИ ИЗМЕНЕНИЯ НУЖНО ВНЕСТИ В ЭТОТ ОБЩИЙ МОДУЛЬ

Ну и далее используем нашу процедуру также, как и обычно с любой другой экспортной процедурой.

НО вообще лучше избегать вывода общих команд куда-либо кроме командного интерфейса.