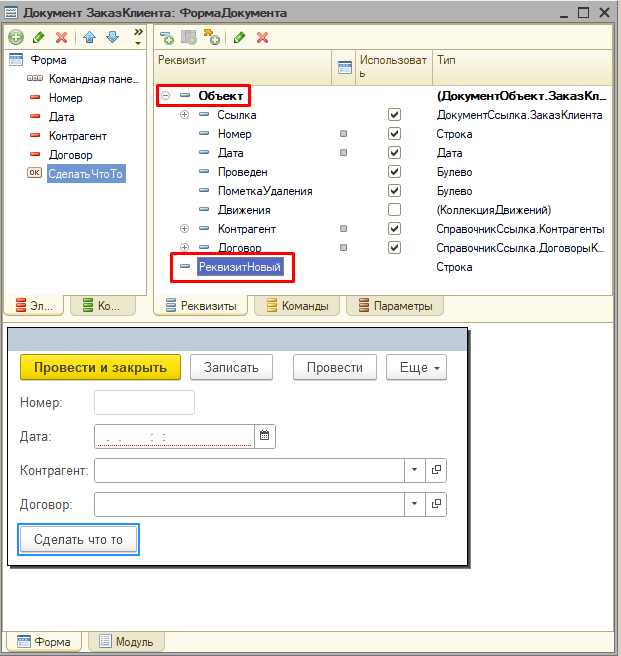
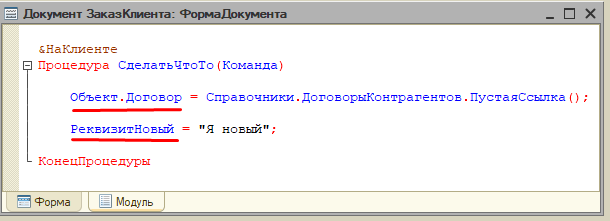
**Управляемые формы**

**1.** В управляемой форме мы не можем напрямую обращаться к реквизитам объекта потому-что у нас нет его контекста. Можем обращаться только к реквизитам формы.





**2.** Все интерактивные дейсвтия (открытие форм, показ диалогов, печать и т. д.) выполняются на стороне клиента.  
Сложные вычисления и работа с данными информационной базы (чтение, изменение, удаление) выполняются на стороне сервера.

**3.** Чтобы обратиться ко всем элементам формы нужно обратиться к «Элементы».

**4.** Для доступа к реквизитам формы используется Имя реквизита.

**5.** Форма всегда создается на стороне сервера, а потом передается на клиент.

**6.** Чтобы обновить данные формы (F5), нужно использовать метод «ОповеститьОбИзменении()».

**7.** Директивы компиляции:

* **&НаКлиенте** — код который внутри такого метода исполняется на клиенте. На стороне клиента можно работать со всеми реквизитами и элементами формы.
* **&НаСервере** — при вызове метода с директивой  **&НаСервере** из клиентской процедуры управление передается серверу 1с.   
  При передаче управления на сервер 1с передается целиком вся форма включая все её элементы и реквизиты формы, а потом эта же форма заеново упаковывается и передается обратно на клиент. Поскольку на сервер передается вся форма, то на стороне сервера мы можем работать с этой формой так же как мы бы работали с ней на клиенте. Тоесть теперь мы так же можем обращаться на сервере к реквизитам и элементам формы.
* **&НаСервереБезКонтекста** — если используется эта директива комплияции, то с клиента на сервер форма передаваться не будет и вообще никакие данные передаваться не будут кроме тех данных которые указаны в параметрах метода. Тоесть при использовании этой директивы уменьшается объем передаваемых данных с клиента на сервер.
* **&НаКлиентеСервереБезКонтекста** —  метод, описанный под данной директивой управления, создаётся в двух копиях – и на стороне клиента, и на стороне сервера. Это позволяет выполнить необходимые действия там, где появилась потребность в них (клиент/сервер), без лишних серверных вызовов. При этом реквизиты формы и элементы формы будут недоступны. Так же нельзя работать с базой данных. В таких методах пишут общий код который будет работать как на клиенте так и на сервере.

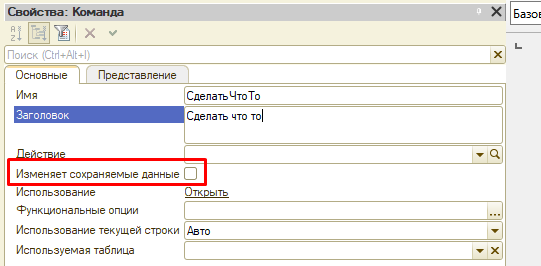
**ВАЖНО,** придерживайтесь при разработке следующих правил:

* По возможности не передавайте контекст формы на сторону сервера
* Минимизируйте количество текущих серверных вызовов
* Длительные и ресурсоёмкие задачи запускайте на выполнение на стороне сервера (при возможности – в фоновом режиме).

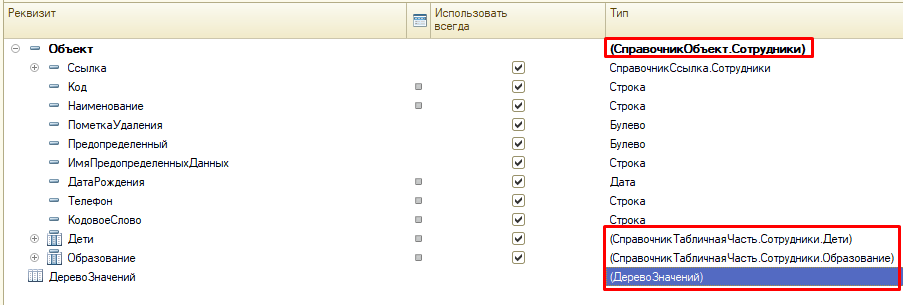
Ссылка для лучшего понимания:

<https://infostart.ru/1c/articles/682305/>

**8.** Если у команды стоит галка «**изменяет сохраняемые данные**», то нажатие на такую команду у формы будет устанавливаться признак модифицированности.



**9.** **ДанныеФормы** — если у реквизита формы тип указан в скобках — это значит, что   
по факту будет не тот тип который указан в скобках, а будет тот тип который **эмулирует**   
тот тип который находится в скобках. Тоесть все типы прикладных объектов (СправочникОбъект, ДокументОбъект, Таблица значений и т. д.) не существуют на стороне тонкого и веб-клиентов, они существуют только на сервере. Для представления в форме данныъ этих прикладных типов введены специальные типы данных, которые предназначены для работы в управаляемых формах.



В управляемых формах на клиенте каждый пользователь по сути работает со слепком данных на какой то определенный момент времени, тоесть пользователь открыл форму — в этот момент данные считались какие то данные и с ними он работает, но если параллельно кто то эти данные поменяет, то пользователю придется перечитать новую версию и уже потом вносить свои какие то изменения.

Для того, чтобы работать с данными объекта на клиенте используются специальные типы, которые образуют **данные формы.**

**ДанныеФормы** работают на клиенте и на сервере, а прикладные объекты (СправочникОбъект, ДокументОбъект, ТаблицаЗначений и т.д.) работают только на сервере.

Существуют 4 типа данных формы:

* **ДанныеФормыСтруктура** — содержит набор свойств произвольного типа.
* **ДанныеФормыКоллекция** — это список типизированных значений, похожий на массив.
* **ДанныеФормыСтруктураСКоллекцией** — это объект, который представлен в виде структуры и коллекции одновременно.
* **ДанныеФормыДерево** — объект предназначен для хранения иерархических данных.

Для того, чтобы выполнять явные преобразования из данных формы в прикладной объект и наоборот используются следующие методы:

**Методы глобального контекста:**

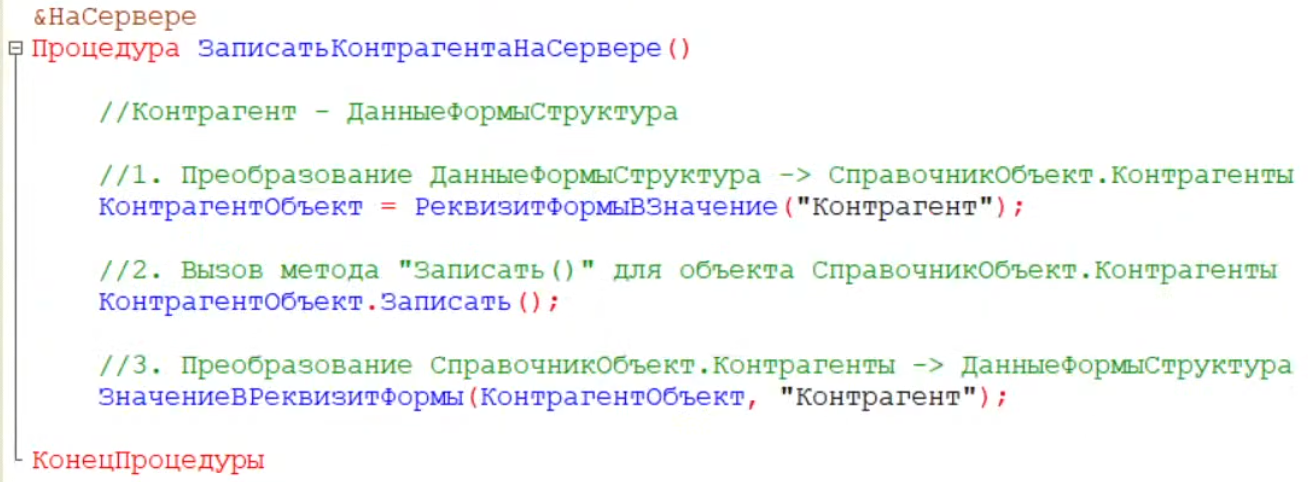
* **ЗначениеВДанныенФормы()** — преобразует объект прикладного типа в данные формы.
* **ДанныеФормыВЗначение()** — преобразует данные формы в объект прикладного типа.

**Методы управляемой формы:**

* **ЗначениеВРеквизитФормы()** — преобразует объект прикладного типа в реквизит управляемой формы.
* **РеквизитФормыВЗначение()** — преобразует реквизит управляемой формы в значенеие прикладного типа.

**Примечание**: эти методы работают только на сервере.

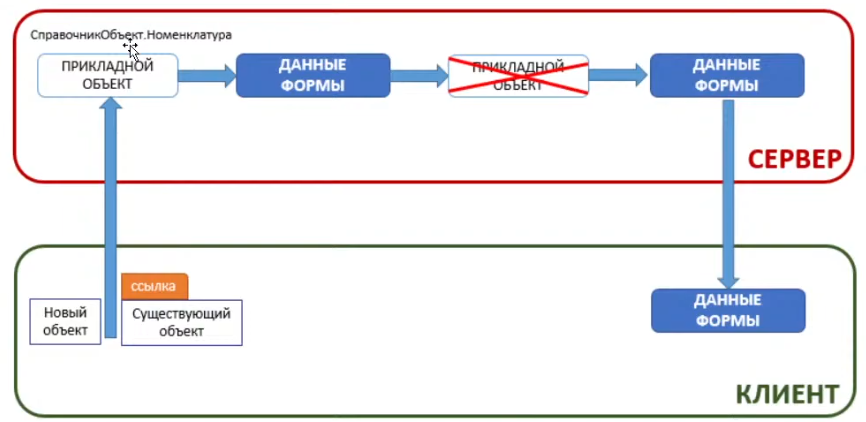
**Пример:**



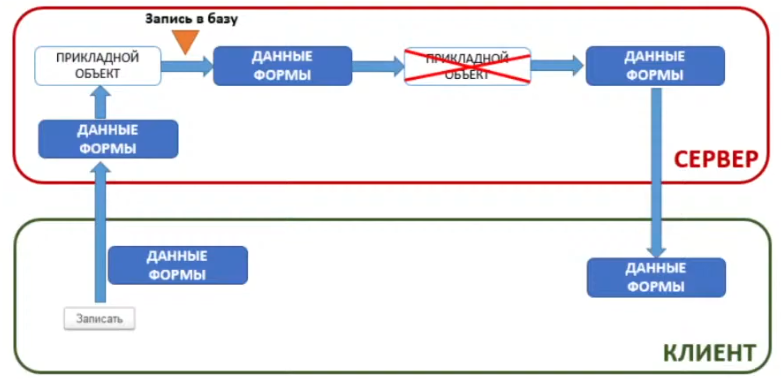
**10.** Функциональную часть алгоритма лучше размещать в модулях прикладных и в общих модулях потому-что:

* Из модуля формы **можно** вызвать метод модуля объекта.
* Из модуля объекта **нельзя** вызвать метод модуля формы.

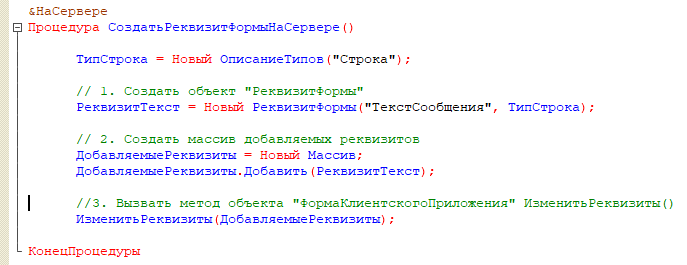
**11.** При открытии формы происходит следующее:



**12.** При записи объекта происходит следующее:



**13.** Чтобы программно создать реквизит формы нужно:

1. Создать объект «РеквизитФормы».
2. Создать массив и добавить в него РеквизитФормы (из шага 1).
3. Вызвать метод формы ИзменитьРеквизиты(), передав массив реквизитов.

**14.** Чтобы программно создать элемент формы нужно:

