

Тренинг **Разработка мобильного приложения на 1С с нуля – за 7 вечеров!**

Модуль 2. Обмен данными с мобильным приложением через веб- сервисы

Оглавление

| | |
|---|----|
| Второй день (модуль). Введение | 3 |
| Веб-сервисы | 4 |
| Что такое веб-сервисы?..... | 5 |
| Веб-сервисы в практике | 5 |
| URI пространство имен | 6 |
| XDTO-пакеты | 7 |
| WSDL-схема | 8 |
| WS-ссылка | 9 |
| Создание XDTO-пакета, заполнение, чтение | 9 |
| Создание XDTO-пакета | 10 |
| Работа с <i>ОбъектXDTO</i> | 16 |
| Создание веб-сервиса | 19 |
| Публикация веб-сервиса | 21 |
| Создание ws-ссылки | 22 |
| Использование ws-ссылки | 23 |
| Динамическое создание ws-ссылки | 24 |
| Сериализация и десериализация..... | 24 |
| Как сериализовать и десериализовать объекты?..... | 25 |
| Отладка веб-сервиса..... | 28 |
| Итоги | 30 |

Второй день (модуль). Введение

Основная цель текущего дня тренинга – продемонстрировать передачу информации между мобильным устройством и сервером. Многие на интуитивном уровне понимают, что обмен возможно сделать при помощи файлов. Например, отправлять файл по почте, или размещать на некий ftp-сервер и т.д.

Однако на практике для использования обмена с мобильным устройством такой подход не удобен. Прежде всего из-за того, что у каждого мобильного пользователя есть своя микробаза, с которой работает только он.

К примеру, если на стационарном компьютере с развернутым образом узла распределенной базы по какой-то причине не пройдет обмен, то администратор может открыть базу и посмотреть, что в ней произошло.

В базе на мобильном устройстве только пользователь будет знать о появлении, например, нового документа заказа или нового контрагента. После передачи данных серверу этот сотрудник не будет иметь возможности проверить успешность загрузки.

При реализации обмена через файлы возможны различные проблемы. Например, из мобильного приложения произошла выгрузка нового заказа на ftp, далее нужно будет ожидать, когда центральная база его загрузит и сформирует ответ. То есть мобильный клиент вынужден опрашивать ftp и проверять наличие файлов. А так как на мобильной платформе не доступны регламентные задания, придется либо подключать обработчик ожидания, либо периодически нажимать на кнопку проверки. А если сервер отработал с ошибкой и, к примеру, из-за ошибки удалил файл выгрузки с ftp, то клиент не дожидается ответа никогда.

Таким образом, синхронизация через файлы сводят на нет многие преимущества использования мобильных устройств в бизнес-процессах компании.

Однако есть и другой вариант синхронизации данных. Существует возможность передавать данные непосредственно на сервер, и получать ответ сразу после загрузки. Как это сделать? На этот вопрос мы и ответим в рамках текущего дня тренинга.

Возникает еще один вопрос, который, как и предыдущий, имеет несколько вариантов решения. Это вопрос о том, как переносить данные на мобильное устройство. Ведь на мобильном устройстве необходимы справочники, документы и прочее. Но как это сделать? Какие варианты для ускорения загрузки данных на мобильное устройство у нас есть?

Согласитесь, что если передавать некий самописный вариант xml-файла, то принимая его следует выполнять ряд операций:

- прочитать *xml*;
- если это справочник или документ – найти его по коду;
- найти по коду (или *UID*) все реквизиты ссылочного типа и т.д.

На самом деле ничего этого делать не нужно! То есть придумывать свой формат *xml*, писать код по его парсингу, искать объекты ссылочного типа и т.д. Об этом мы также расскажем в рамках текущего дня тренинга.

Специалисты, которые ранее работали с планами обмена и конвертацией данных могут возразить, и сказать, что достаточно настроить план обмена и применить к нему правила конвертации.

Однако на мобильном устройстве нет запросов и некоторых других объектов, а без них обработка по конвертации работать не будет.

Теперь посмотрим, почему нельзя использовать планы обмена. Причина в том, что зачастую в мобильной базе будет упрощенный состав метаданных, например, мобильному пользователю не нужны все реквизиты номенклатуры. Можно, конечно, пытаться описать все объекты из клиентской базы через фабрики *XDTO* и настроить планы обмена, но это достаточно трудоемко. В добавок, при каждом изменении метаданных клиента потребуются переписывать код на сервере.

Итак, мы узнаем каким образом передавать любые данные на сервер и после этого получать от него ответ. А также как переносить справочники и документы с минимумом кода и максимальной гибкостью.

Веб-сервисы

Для переноса данных на сервер и получения ответа, мы будем использовать такой объект 1С как веб-сервисы.

Что такое веб-сервисы?

Процитируем справку 1С:

***Web-сервисы** – это один из механизмов платформы, используемых для интеграции с другими информационными системами. Он является средством поддержки SOA (Service-Oriented Architecture) – сервис-ориентированной архитектуры, которая является современным стандартом интеграции приложений и информационных систем.*

Значительным преимуществом сервис-ориентированной архитектуры является то, что она позволяет развивать инфраструктуру предприятия однородным образом, без разрушения уже существующих решений. Ее использование позволяет минимизировать издержки за счет интеграции разнородных и унаследованных систем в современный ландшафт предприятия. Она позволяет реализовывать слабо связанные программные компоненты с тем, чтобы максимально повысить их повторную используемость.

Если попытаться объяснить это проще, то веб-сервисы – это некая внешняя функция, которой передаются параметры и после обработки этих параметров получается ответ.

То есть, можно просто с клиента вызвать некую функцию на сервере, передать туда, например, отборы по регистру взаиморасчетов с контрагентами. А затем на клиенте получить ответ о долге контрагента. При этом, на клиенте вообще может не быть этого регистра.

Хочу обратить Ваше внимание на то, что веб-сервисы – это частная реализация [SOAP](#), эта технология является распространенной. Например, Яндекс [позволяет](#) получать данные при помощи этого протокола. Есть и другие сервисы – по погоде, курсам валют и т.д.

Веб-сервисы в практике

Рассмотрим пример. Допустим, несколько человек программируют одну и ту же конфигурацию.

У одного из них есть некий закрытый модуль, в котором описана функция:

`ПолучитьСреднее (А, В, В=0)`

Второй программист хочет воспользоваться этой функцией. Но сразу появится вопрос – какие типы у параметров *A*, *B* и *V*, какие параметры являются обязательными?

На этот вопрос и отвечают веб-сервисы. И в случае, если даже первый программист добавит новый параметр и ничего не скажет о нем второму – веб-сервис даст полное описание того, что это за тип параметра, может ли он быть пустым.

Но типы параметров и ответа могут быть разные, как же однозначно определить тип переменной?

Например, в 1С можно обмениваться типом *СправочникОбъект.Номенклатура*. Естественно, сайт не сможет принять такой тип данных, если он не определен ранее. Именно по этой причине внедрили еще одно понятие – [URI пространство имен](#).

URI пространство имен

Допустим мы решили подключиться к веб-сервису сайта погоды, он возвращает нам температуру. Вопрос – он вернет температуру как число, или как строку?

И в том и в другом случае он вернет, к примеру – «10».

Как клиент должен понять, что это такое? Важно, если вы пишете синхронизацию зимой, и сервер вернет «-10», то не совсем понятно, как он вернет температуру выше нуля, т.е. как число «10», или как строку «+10»?

Мы понимаем, что сайт погоды может и не знать о существовании 1С и ее типах. Поэтому должен быть некий стандарт, которым однозначно можно описать простые типы, такие как число, строка, булево и т.д.

И такой стандарт типов существует – это стандарт [W3C](#). Его можно использовать для настройки взаимодействия между абсолютно разными системами.

Все данные этого стандарта находятся в одном пространстве имен – <http://www.w3.org/2001/XMLSchema>. Все типы, которые в него входят описаны на сайте этой организации. Рядом с типом указывается пространство имен. Например, если мы определим тип как строку из этого пространства имен, то он будет выглядеть вот так:

string (<http://www.w3.org/2001/XMLSchema>)

Это связано с тем, что имена типов в разных пространствах имен могут совпадать. Поэтому, для однозначного определения типа, необходимо понимать из какого пространства имен он взят.

Кроме международных стандартов, есть и локальные стандарты, которые описывают объекты внутри неких систем. 1С не исключение, у нее есть огромный выбор локальных пространств имен.

С их помощью можно обмениваться объектами 1С, например, структурой, массивом, таблицей значений и т.д. Свои собственные типы описываются при помощи XDTO-пакетов.

XDTO-пакеты

Процитируем к справку 1С:

***XDTO-пакеты** – это общие объекты конфигурации. Они являются частью механизма XDTO. XDTO-пакеты позволяют описать в конфигурации систему типов и значений для взаимодействия с различными внешними источниками данных, например, с Web-сервисами.*

Где можно применять XDTO-пакеты?

Вернемся к примеру с закрытым модулем и представим себе, что параметр А – это не число, а некий набор данных. Для примера – в этом параметре должна быть структура с такими ключами:

- «МассивЧисел» – тип Массив;
- «КоличествоЭлементовВМассиве» – тип число;
- «Комментарий» – тип строка.

Идея заключается в том, что первый программист в своей функции ждет именно такую структуру входящего параметра. Причем допустим, есть ограничения на значения параметров – массив и количество не могут быть пустыми, а длина комментария не должна превышать 100 символов.

Как объяснить все это второму программисту, который будет обращаться к этой функции? Для этого необходимо написать документацию, а кроме этого и целый ряд проверок на ошибки.

Например, действительно ли ключи присутствуют в структуре, все ли необходимые поля заполнены, не превышает ли длина комментария 100 символов (для того чтобы при записи комментариев не был обрезан) и т.д.

А теперь представим, что к этой функции выполняется 100 вызовов в секунду. На сколько все эти проверки замедлят работу?

Но что, если этого всего можно избежать? Например, создать свой тип, который будет доступен всем. И при его создании будет выполняться проверка всего, что необходимо.

Таким образом, не нужно писать проверки на значения параметров, объяснять остальным участникам проекта какие поля обязательны, какого типа и т.д.

Можно просто создать свой тип, в своем пространстве имен, сообщить всем участникам, что нужно передавать именно этот тип. А в самой функции сделать только одну проверку на то, что параметр имеет нужный нам тип.

Именно это и позволяют сделать XDTO-пакеты.

WSDL-схема

Если мы работаем в одной базе данных, то с вызовом функций проблем нет, так как XDTO-пакет будет доступен всем участникам. А что, если эта функция находится в другой базе данных, на другом сервере? На самом деле, это не является проблемой, так как публикуя веб-сервис, он публикует WSDL-схему в общий доступ, и на ее основе можно создать новый XDTO-пакет или импортировать его в виде WS-ссылки.

WSDL-схема – это xml-файл, который построен специальным образом.

Каждый документ WSDL 1.1 можно разбить на следующие логические части:

- определение типов данных (*types*) — определение вида отправляемых и получаемых сервисом XML-сообщений;
- элементы данных (*message*) — сообщения, используемые web-сервисом;
- абстрактные операции (*portType*) — список операций, которые могут быть выполнены с сообщениями;
- связывание сервисов (*binding*) — способ, которым сообщение будет доставлено.

Таким образом, прежде чем передать данные на сервер, программа получает описание всех типов, функций и т.д., и только после этого передает параметры на обработку, после чего получает ответ. Тип ответа так же прописан в WSDL-схеме.

WS-ссылка

Цитата из справки 1С:

WS-ссылка - это общий объект конфигурации. Она предназначена для описания в прикладном решении "статической" ссылки на некоторый внешний [веб-сервис](#) стороннего поставщика.

WS-ссылка представляет собой WSDL-описание веб-сервиса, импортированное из указанного источника. WS-ссылка недоступна для редактирования, однако можно просмотреть ее структуру и структуру типов данных, которые используются для описания параметров и возвращаемых значений.

То есть, если мы работаем с другой базой, то мы можем подключить в нашей базе ws-ссылку и работать с ней, как с неким аналогом XDTO-пакета.

Создание XDTO-пакета, заполнение, чтение

Реализуем на практике описанный выше пример. Условия таковы:

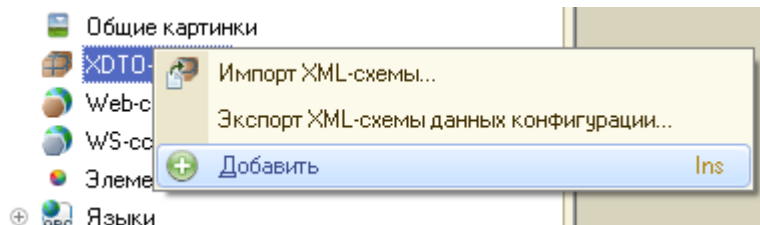
Есть функция *ПолучитьСреднее(СтруктураДанных)*, которая должна возвращать среднее значение элементов массива чисел. В нее передается XDTO-пакет следующего содержания:

- *МассивЧисел* – тип *Массив*;
- *КоличествоЭлементовВМассиве* – тип число, должно быть целым (это же количество элементов), и большим нуля (нам на него делить);
- *Комментарий* – тип строка, длина строки менее 100 символов;

В ответ мы должны получить сумму всех чисел в массиве деленную на *КоличествоЭлементовВМассиве*. Ответ может быть либо пустым – в случае если произошла ошибка, либо число – если все прошло успешно.

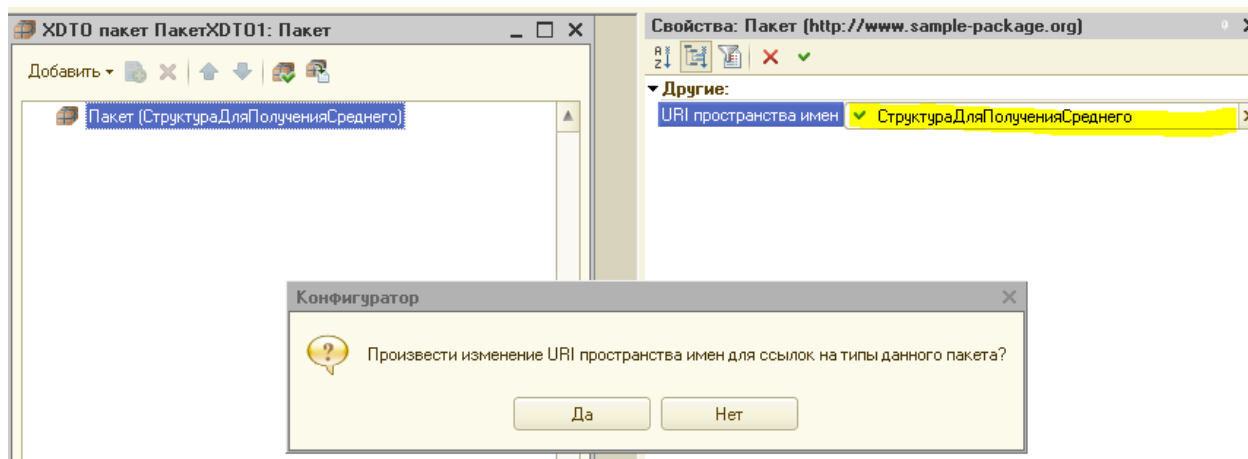
Создание XDTO-пакета

Для начала создадим XDTO-пакет:

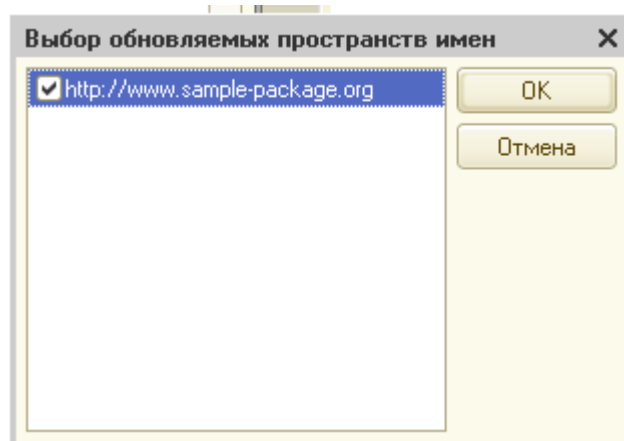


И сразу поменяем URI-пространство имен. Обычно, пространство имен представляют в виде ссылки, по которой должно размещаться описание этого типа. Но можно задать любое имя.

Назовем пространство *«СтруктураДляПолученияСреднего»*, для этого поменяем имя:

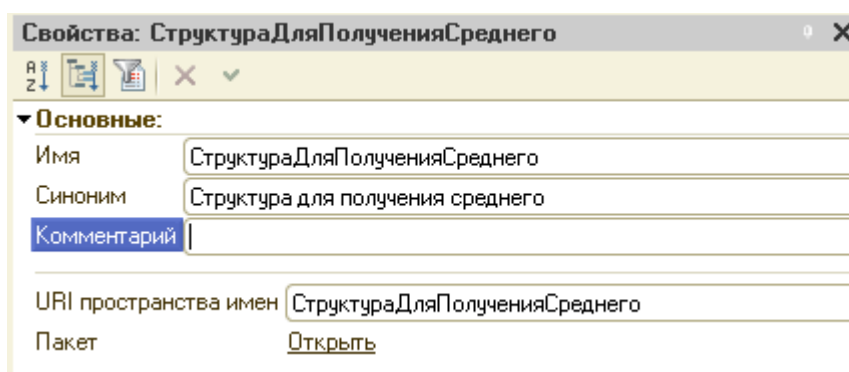


При этом программа спросит – хотим ли мы изменить пространство имен у всех ссылок, соглашаемся.



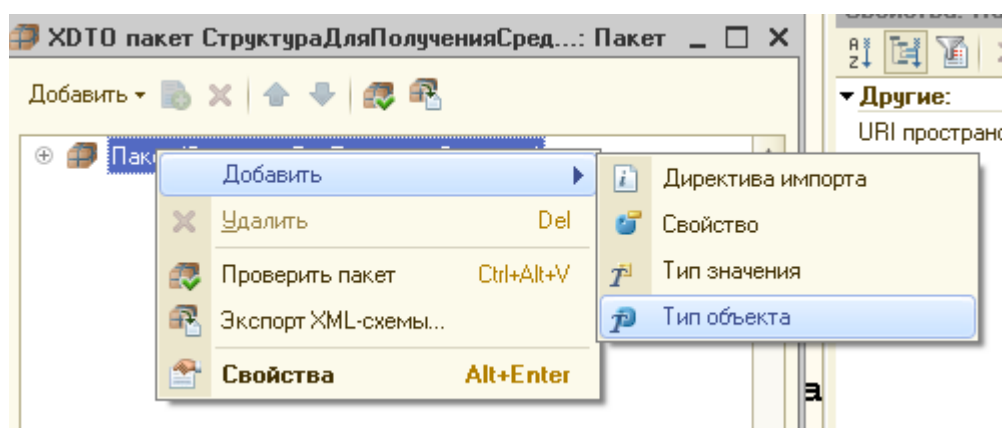
Ставим флажок у старого пространства имен, которое 1С поставило при создании, и нажимаем *OK*.

Теперь поменяем имя самого пакета, сделаем его таким же, как и пространство имен:

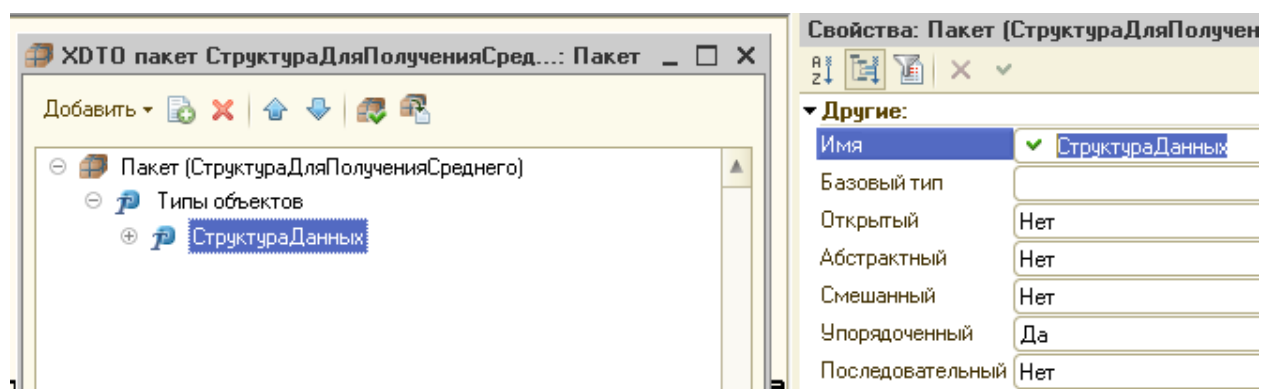


Теперь поэтапно создадим все, что необходимо. То есть опишем структуру пакета. Сначала необходимо создать всю структуру в целом.

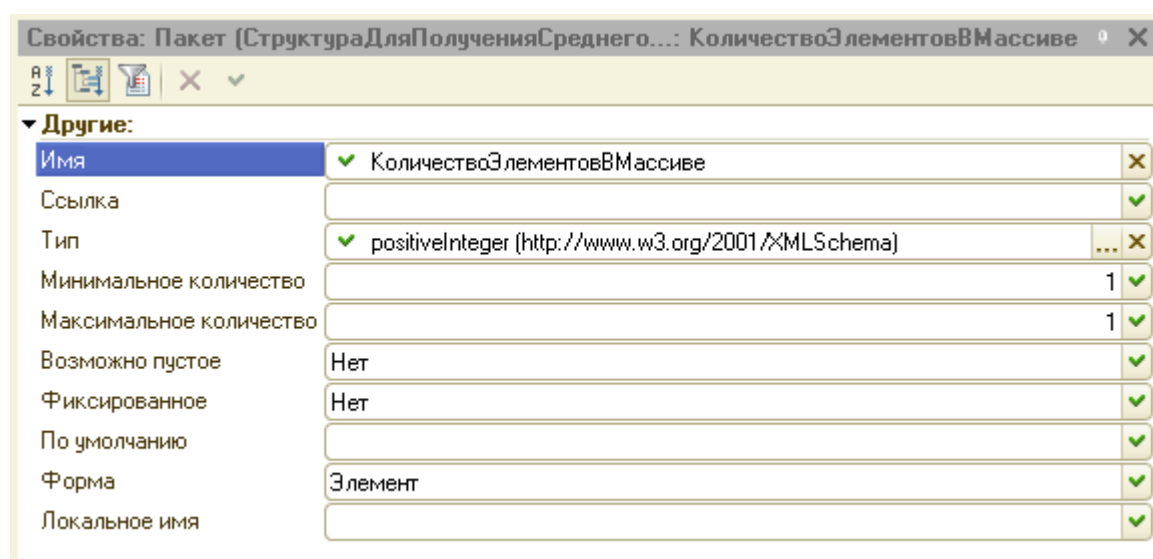
Так как XDTO-пакет – это описание типов объектов, то надо создать новый тип:



Назовем новый тип – *СтруктураДанных*.



Начнем с самого простого – опишем тип *КоличествоЭлементовВМассиве*. Для этого на элементе *СтруктураДанных* вызываем контекстное меню *Добавить – Свойство*, и указываем данные:



Рассмотрим основные свойства:

- *Имя* – по нему будем обращаться к этому типу (грубо говоря – ключ структуры);
- *Тип* – тип из выбранного пространства имен, который описывает тип добавленного свойства. Выбор очень большой, но мы работаем пока только с типами, которые находятся в пространстве имен <http://www.w3.org/2001/XMLSchema>. Так что найдите в дереве это пространство и выберите тип.

Почему выбираем именно это тип?

Вспомним задачу. Нам необходимо, чтобы сюда попадали только целые числа и больше нуля. Поэтому идем открываем страницу http://www.w3schools.com/schema/schema_dtypes_numeric.asp и смотрим внизу описание типов:

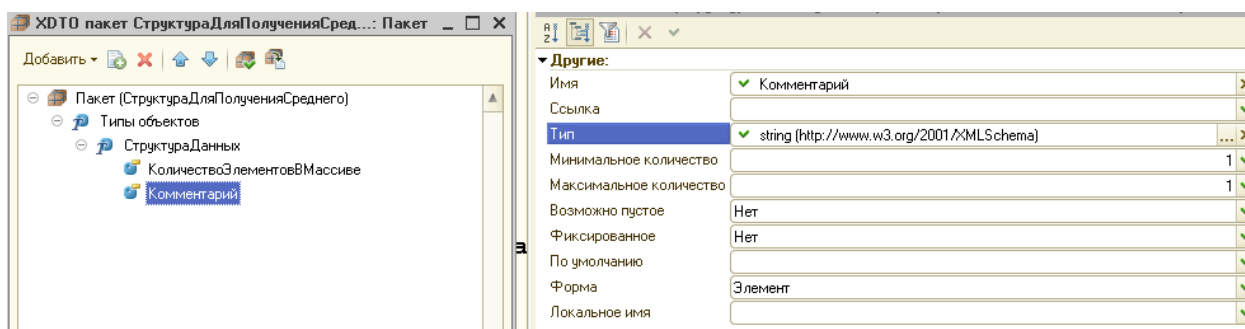
| Name | Description |
|---------------------------|---|
| byte | A signed 8-bit integer |
| decimal | A decimal value |
| int | A signed 32-bit integer |
| integer | An integer value |
| long | A signed 64-bit integer |
| negativeInteger | An integer containing only negative values (..,-2,-1) |
| nonNegativeInteger | An integer containing only non-negative values (0,1,2,..) |
| nonPositiveInteger | An integer containing only non-positive values (..,-2,-1,0) |
| positiveInteger | An integer containing only positive values (1,2,..) |
| short | A signed 16-bit integer |
| unsignedLong | An unsigned 64-bit integer |
| unsignedInt | An unsigned 32-bit integer |

| | |
|----------------------|----------------------------|
| unsignedShort | An unsigned 16-bit integer |
| unsignedByte | An unsigned 8-bit integer |

Нас интересует целое число больше нуля, поэтому берем *positiveInteger*. Если бы требовалось целое число больше либо равное нулю, то следовало бы выбрать *nonNegativeInteger*. А если разницы нет, то можно выбрать *decimal* – оно может быть любым, в том числе и дробным.

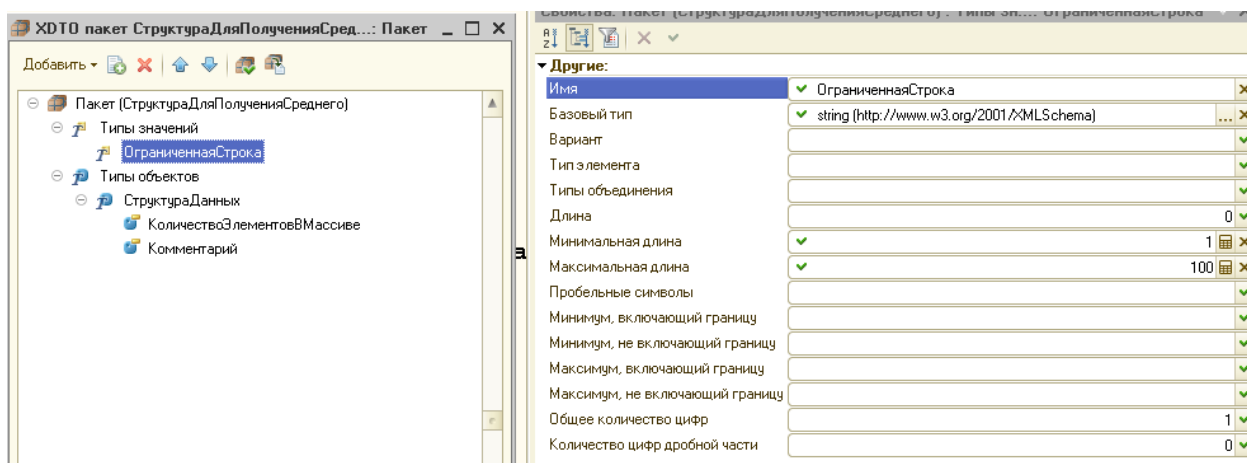
- *Минимальное количество и максимальное* – задает диапазон допустимых значений.
- *Возможно пустое* – не обязательный параметр для заполнения.
- *По умолчанию* – указываем данные, которыми будет заполнен параметр данного типа в момент создания.
- *Форма и локальное имя* – используются, если вы будете выгружать данный пакет в xml.

Теперь добавим свойство *Комментарий* по аналогии:

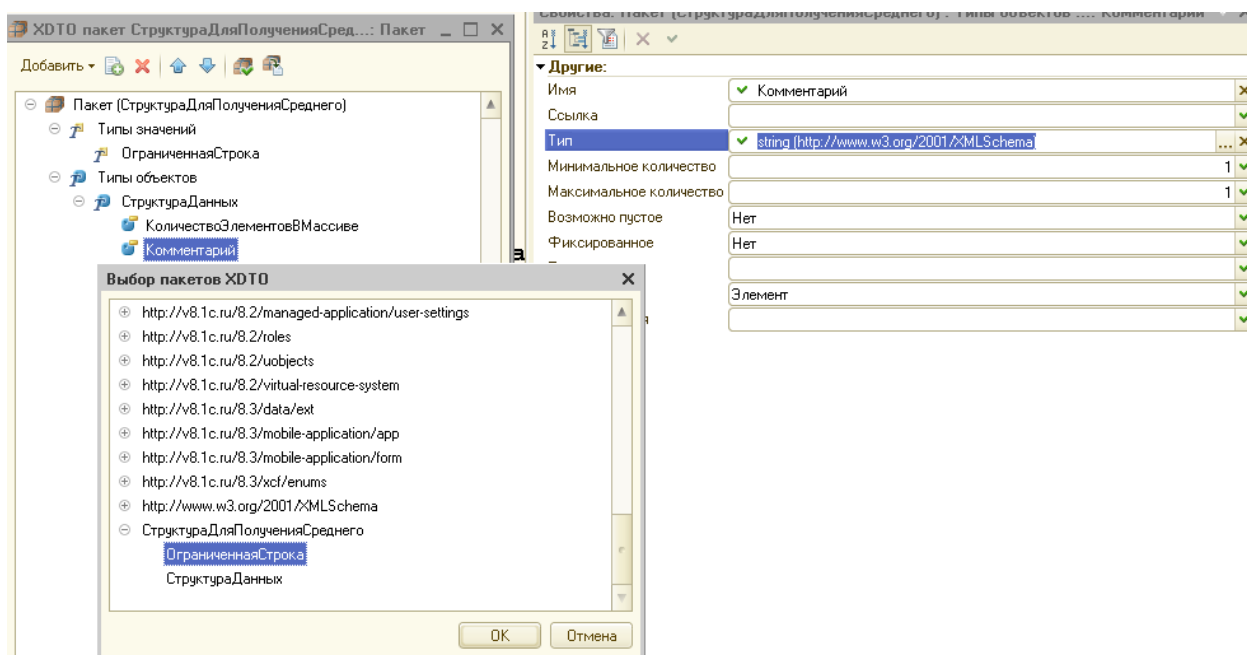


Только тип выберем другой – *string*, из того же пространства имен. Теперь необходимо указать максимальную длину, но указать ее негде. По этой причине мы должны создать свой тип.

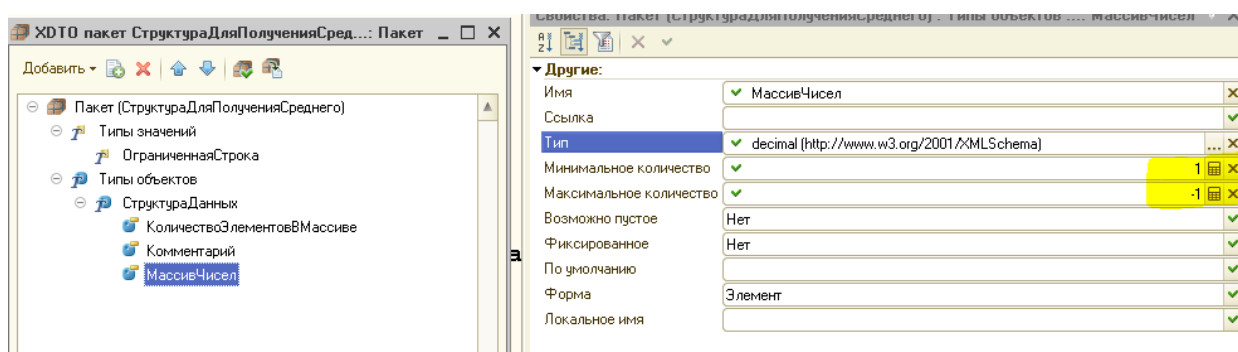
Для этого нажимаем правой кнопкой на самом верхнем уровне XDTO-пакета *Добавить – Тип значения*:



Назовем этот тип *ОграниченнаяСтрока*. Выбираем базовый тип, и далее указываем минимальную и максимальную длину. Теперь выберем новый тип:



И остался последний штрих – добавить массив чисел. Добавляем новый тип объектов:



Обратите внимание на минимальное и максимальное количество. Можем ограничить количество элементов в массиве, а можем и не ограничивать (тогда ставим «-1»).

Тип выбираем *decimal*, таким образом в массив можно передать любое число. В массиве должен быть минимум один элемент.

Работа с ОбъектХДТО

Создадим простую форму с одной кнопкой и ставим вот такой код:

```
&НаКлиенте
Процедура СформироватьПакет (Команда)
    СформироватьПакетНаСервере ();
КонецПроцедуры

&НаСервереБезКонтекста
Процедура СформироватьПакетНаСервере ()
    ТипПакета = ФабрикаХДТО.Тип ("СтруктураДляПолученияСреднего",
"СтруктураДанных");
    СтруктураДанных = ФабрикаХДТО.Создать (ТипПакета);
КонецПроцедуры
```

Вначале мы получаем описание типа (первый параметр – URI-пространство имен, второй – имя типа объекта), далее, на основании типа, создаем новый объект ХДТО *СтруктураДанных*.

Посмотрим в отладчике:

| Табло - 1 | | |
|-----------------------------|-------------|--------------|
| Выражение | Значение | Тип |
| ⊖ СтруктураДанных | ОбъектXDTO | ОбъектXDTO |
| КоличествоЭлементовВМассиве | | Неопределено |
| Комментарий | | Неопределено |
| ⊖ МассивЧисел | СписокXDTO | СписокXDTO |
| ⊕ Владелец | ОбъектXDTO | ОбъектXDTO |
| ⊕ ВладеющееСвойство | МассивЧисел | СвойствоXDTO |

К сожалению, в контекстной подсказке не будет помощи в виде доступных параметров. Заполним все параметры:

```
&НаСервереВезКонтекста
Процедура СформироватьПакетНаСервере ()
    ТипПакета = ФабрикаXDTO.Тип ("СтруктураДляПолученияСреднего",
"СтруктураДанных" );
    СтруктураДанных = ФабрикаXDTO.Создать (ТипПакета);
    СтруктураДанных.КоличествоЭлементовВМассиве = -2;
    СтруктураДанных.Комментарий = "Тест";
КонецПроцедуры
```

Выполним код и увидим следующую ошибку:

{ОбщаяФорма.Форма.Форма(11)}: Ошибка при установке значения атрибута контекста (КоличествоЭлементовВМассиве)

СтруктураДанных.КоличествоЭлементовВМассиве = -2;

по причине:

Ошибка проверки данных XDTO:

*Значение: '-2' не соответствует простому типу:
{http://www.w3.org/2001/XMLSchema}positiveInteger*

Несоответствие фасеты MinInclusive = '1'

Исправим ошибку и заменим «-2» на «10». Выполним код. И никаких ошибок не будет. Но ранее мы указывали, что в массиве должен быть хотя бы один элемент. Следовательно, нужно выполнить проверку объекта вручную. Для этого достаточно дописать еще одну строку кода:

```
СтруктураДанных.Проверить ();
```

И еще раз выполним код. Нас встретит вот такая ошибка:

{ОбщаяФорма.Форма.Форма(13)}: Ошибка при вызове метода контекста (Проверить)

СтруктураДанных.Проверить();

по причине:

Ошибка проверки данных XDTO:

Структура объекта не соответствует типу:

{СтруктураДляПолученияСреднего}СтруктураДанных

Проверка свойства 'МассивЧисел':

форма: Элемент

имя: {СтруктураДляПолученияСреднего}МассивЧисел

тип: {http://www.w3.org/2001/XMLSchema}decimal

Отсутствует обязательное свойство

Тогда допишем код:

```
&НаСервереБезКонтекста
Процедура СформироватьПакетНаСервере ()
    ТипПакета = ФабрикаXDTO.Тип ("СтруктураДляПолученияСреднего",
"СтруктураДанных");
    СтруктураДанных = ФабрикаXDTO.Создать (ТипПакета);
    СтруктураДанных.КоличествоЭлементовВМассиве = 2;
    СтруктураДанных.Комментарий = "Тест";
    СтруктураДанных.МассивЧисел.Добавить (0.1);

    СтруктураДанных.МассивЧисел.Добавить (4);

    СтруктураДанных.Проверить ();
КонецПроцедуры
```

Мы создали свой объект, заполнили его данными и теперь готовы передать его в другую функцию.

Итоговый вид решения нашей задачи будет выглядеть так:

```
&НаСервереБезКонтекста
Процедура СформироватьПакетНаСервере ()
    ТипПакета = ФабрикаXDTO.Тип ("СтруктураДляПолученияСреднего",
"СтруктураДанных");
    СтруктураДанных = ФабрикаXDTO.Создать (ТипПакета);
    СтруктураДанных.КоличествоЭлементовВМассиве = 2;
    СтруктураДанных.Комментарий = "Тест";
    СтруктураДанных.МассивЧисел.Добавить (0.1);
```

```

СтруктураДанных.МассивЧисел.Добавить (4) ;
СтруктураДанных.Проверить () ;
Ответ = ПолучитьСреднее (СтруктураДанных) ;
КонецПроцедуры

&НаСервереБезКонтекста
функция ПолучитьСреднее (СтруктураДанных)
    СтруктураДанных.Проверить () ;
    Сумма = 0 ;
    Для каждого ЭлМассива Из СтруктураДанных.МассивЧисел Цикл
        Сумма = Сумма + ЭлМассива ;
    КонецЦикла ;
    Возврат Сумма/СтруктураДанных.КоличествоЭлементовВМассиве ;
КонецФункции

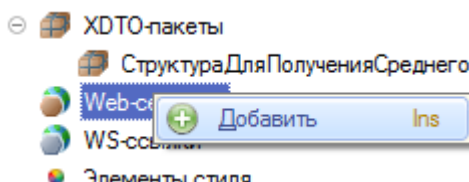
```

Создание веб-сервиса

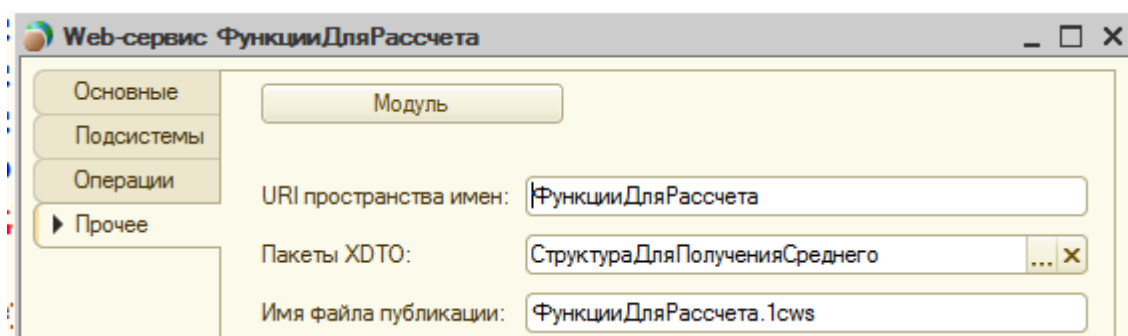
После того, как мы научились создавать XDTO-пакеты, можно перейти к созданию веб-сервисов.

Задача: Создать веб-сервис, разместить в нем функцию для расчета среднего значения элементов массива, передать в параметр структуру данных и получить ответ.

Начнем с создания веб-сервиса.



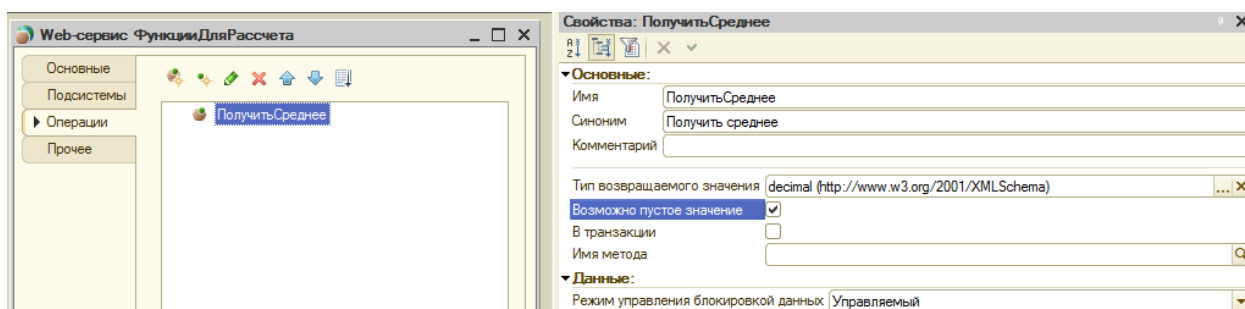
Назовем его «ФункцииДляРасчета». Далее перейдем на вкладку *Прочее* и добавим наш XDTO-пакет. Назначим URI-пространство веб-сервиса и укажем его адрес:



В академических целях, все операции, параметры и прочее – будем писать кириллицей, но это не совсем верно, все, что так или иначе выходит во внешнюю сеть должно быть только на латинице.

Таким образом мы создали веб-сервис в котором мы сможем использовать типы из указанных нами пакетов (типы W3C доступны по умолчанию).

Если вернуться к нашему примеру, то это некий закрытый модуль. Но модуль, сам по себе, не функция. Функции необходимо создать. Для этого переключаемся на закладку *Операции*. Создаем операцию (функцию) и даем ей имя.

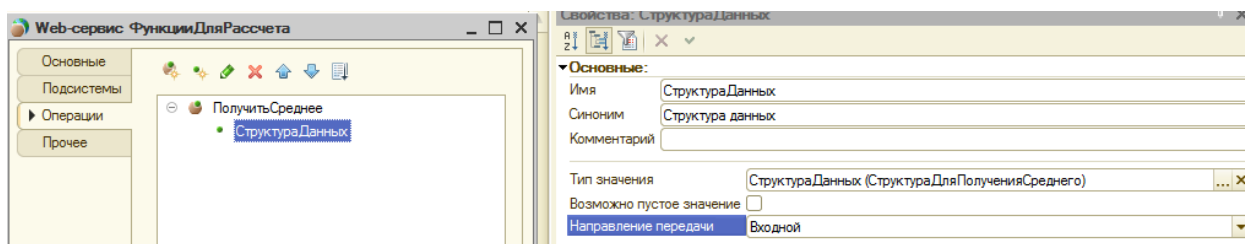


Выбираем *тип возвращаемого значения*. Если тип возвращаемого значения не будет совпадать с указанным – будет вызвано исключение.

Возможно пустое значение – указывает на то, что функция может вернуть *Неопределенно*.

В транзакции – если в модуле функции было вызвано исключение, то произойдет откат всего, что было сделано. Например, если вы создавали элементы справочника, то они исчезнут. Это очень удобно использовать, так как в случае неконтролируемой ошибки лучше сделать полный откат всего, чем искать и исправлять все то, что могла сделать функция не верно. Мы же не будем это использовать, так как у нас нет операций по работе с базой данных.

Далее создадим параметр нашей функции. Нажимаем на созданную операцию правой кнопкой – *Добавить параметр*. Назовем его *СтруктураДанных* и укажем тип значения из нашего *XDTO*-пакета:

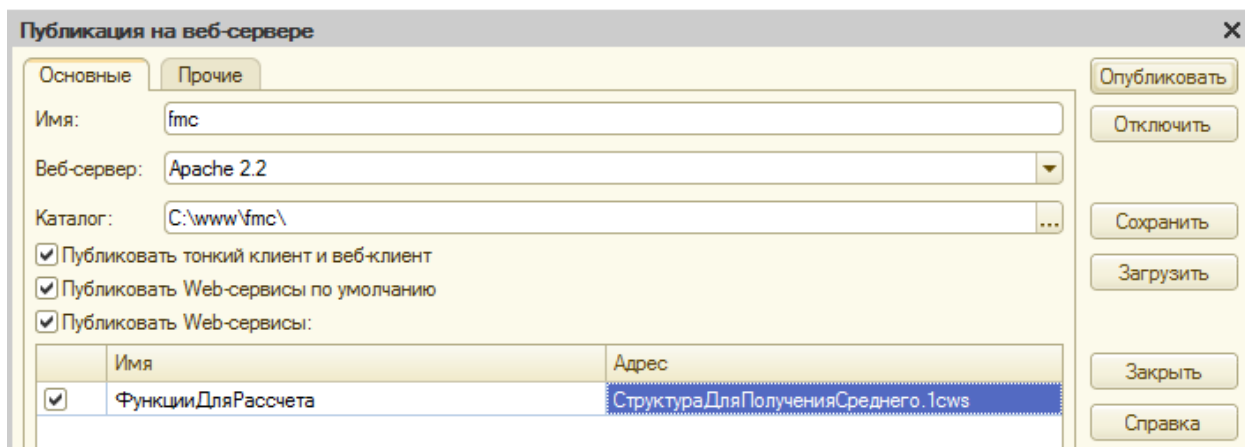


Вернемся в свойство операции и создадим имя метода – нажмем на лупу. Функция будет создана автоматически и сразу будет подставлен параметр.

Скопируем туда код нашей функции для расчета и обновим конфигурацию. Далее необходимо опубликовать его.

Публикация веб-сервиса

Нажимаем в конфигураторе *Администрирование – Публикация на веб-сервере*:



Теперь, при переходе по ссылке в браузере:

<http://127.0.0.1/fmc/ws/ФункцииДляРасчета.1cws?wsdl>

Вы должны увидеть xml-схему. Это и есть описание всего нашего веб-сервиса.

Chrome не сможет открыть это описание, по причине, описанной выше – кириллица в путях и именах, но можно нажать правой кнопкой и просмотреть исходный код страницы.

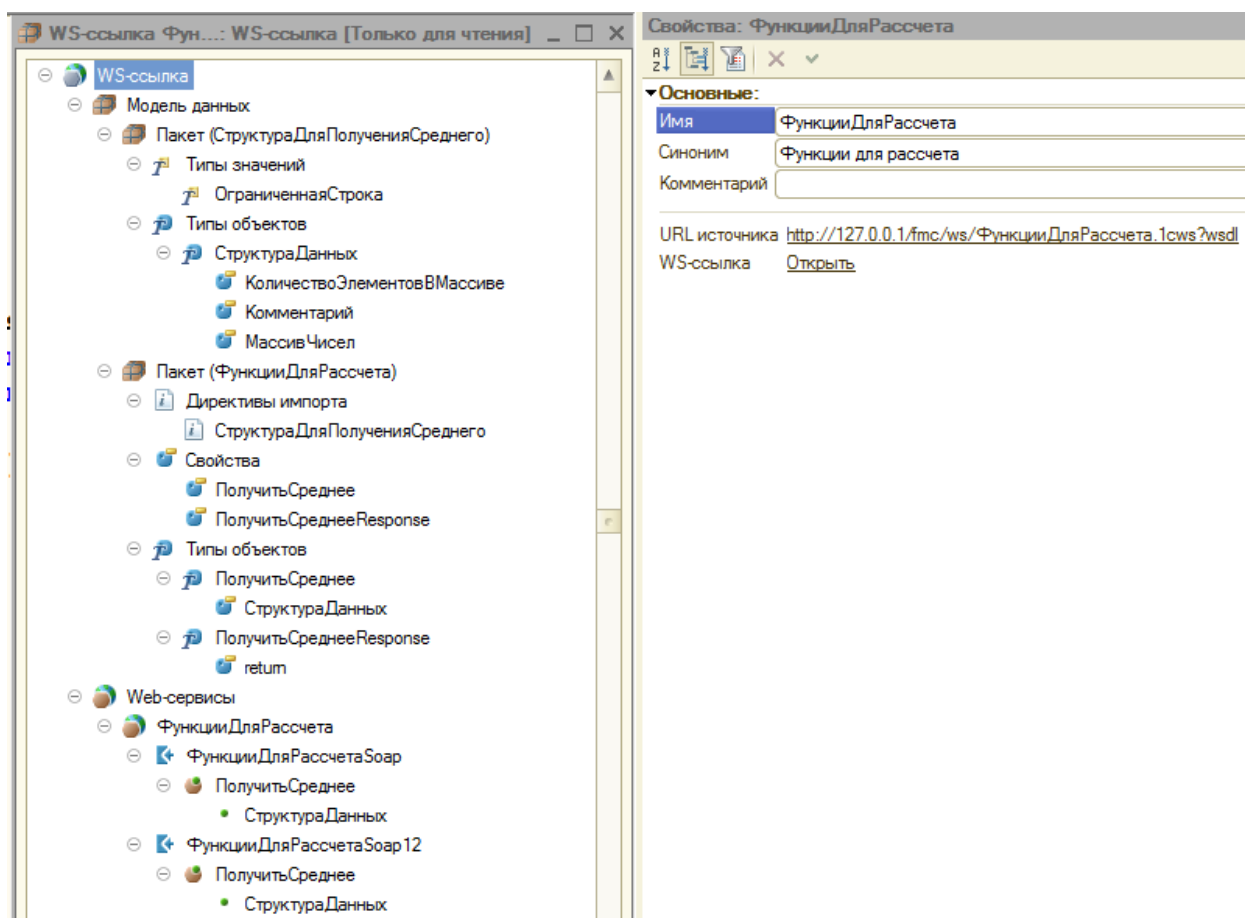
Firefox может скрыть часть определений префиксов пространства имен.

Создание ws-ссылки

Для удобной работы с внешними веб-сервисами, есть такой объект как ws-ссылка. Мы про него ранее говорили. Теперь добавим ws-ссылку и укажем путь:

<http://127.0.0.1/fmc/ws/ФункцииДляРасчета.1cws?wsdl>

После добавления назовем ws-ссылку *ФункцииДляРасчета*:



Видим, что создавалась целая структура, в которой указаны два пакета. Один тот, который мы создавали ранее, второй пакет – это веб сервис. В ws-ссылке есть полное описание всех типов. Так что добавив его мы можем четко понять рамки и возможности серверной функции.

Немного нюансов. Две функции, указанные внизу, на самом деле – одна и та же функция, просто определена она для разных версий *SOAP*, для версии *1.1* и *1.2*. Будем использовать *1.1*, для нас принципиальной разницы нет, так как мы работаем с веб-сервисами при помощи механизмов 1С. А если бы у 1С не было этих механизмов, и мы описывали взаимодействие вручную, т.е. формировали заголовки сообщений, определяли типы и т.д., то тут выбор версии протокола играл бы большую роль. Подробнее про различия можно почитать [тут](#).

Кроме этого, пакет XDTO веб-сервиса содержит два типа. Первый тип описывает параметр, а второй – ответ функции. Так что программируя на клиенте, мы можем ожидать, что функция однозначно вернет требуемый формат.

Использование ws-ссылки

Теперь можем отправить данные на обработку, для этого напишем вот такую функцию:

```
&НаСервереБезКонтекста
Процедура ОтправитьНаСерверНаСервере ()
    Соединение = WSCссылки.ФункцииДляРасчета.СоздатьWSПрокси (
"ФункцииДляРасчета", "ФункцииДляРасчета", "ФункцииДляРасчетаSoap");
    Операция =
Соединение.ТочкаПодключения.Интерфейс.Операции.Получить ("ПолучитьСреднее");
    СтруктураДанных =
Соединение.ФабрикаXDTO.Создать (Операция.Параметры.Получить ("СтруктураДанных")
.Тип);
    СтруктураДанных.КоличествоЭлементовВМассиве = 2;
    СтруктураДанных.Комментарий = "Тест";
    СтруктураДанных.МассивЧисел.Добавить (0.1);
    СтруктураДанных.МассивЧисел.Добавить (4);
    СтруктураДанных.Проверить ();
    Ответ = Соединение.ПолучитьСреднее (СтруктураДанных);
КонецПроцедуры
```

Для начала создаем соединение, в параметрах указываем URI, имя веб-сервиса и порт (версия *SOAP*, так как мы работаем с первой, то просто добавляем к имени сервиса – “Soap”, регистр букв важен).

После этого получаем нужную нам операцию и создаем XDTO-пакет. Обратите внимание на то, как он создается: фабрика объектов берется с ws-ссылки, а не глобальная. Так как глобальная требует встроенный в базу пакет, а в нашем случае на клиенте его может и не быть.

Далее идет обычное заполнение параметров, мы уже делали это ранее.

Ну а в самом конце, мы посылаем данные на обработку на сервер. Подсказки нет, поэтому пишем после соединения имя операции и передаем параметры.

Динамическое создание ws-ссылки

Что делать в случае, если править конфигурацию не хочется, или у нас установлена базовая конфигурация? Т.е. мы не можем добавить новый объект в дерево конфигураций. В этом случае мы можем создать подключение динамически:

```
&НаСервереБезКонтекста
Процедура СозданиеДинамическогоСоединенияНаСервере ()
    ВОпределение = Новый
    WSOпределения ("http://127.0.0.1/fmc/ws/ФункцииДляРасчета.1cws?wsdl");
    ВСервис =
    ВОпределение.Сервисы.Получить ("ФункцииДляРасчета", "ФункцииДляРасчета");
    ВТочкаВхода =
    ВСервис.ТочкиПодключения.Получить ("ФункцииДляРасчетаSoap");
    ВОперация =
    ВТочкаВхода.Интерфейс.Операции.Получить ("ПолучитьСреднее");

    СтруктураДанных = ВОпределение.ФабрикаХДТО.Создать (
    ВОперация.Параметры.Получить ("СтруктураДанных").Тип);
    СтруктураДанных.КоличествоЭлементовВМассиве = 2;
    СтруктураДанных.Комментарий = "Тест";
    СтруктураДанных.МассивЧисел.Добавить (0.1);
    СтруктураДанных.МассивЧисел.Добавить (4);
    СтруктураДанных.Проверить ();

    ВСПрокси = Новый WСПрокси (ВОпределение,
    "ФункцииДляРасчета", "ФункцииДляРасчета", "ФункцииДляРасчетаSoap");

    Ответ = ВСПрокси.ПолучитьСреднее (СтруктураДанных);
КонецПроцедуры
```

Таким образом, нам не обязательно редактировать конфигурацию. Этот способ является более гибким, но требует немного больше кода.

Сериализация и десериализация

Если мы только начали писать код, и как должен выглядеть конечный вариант пакета пока неизвестно, или над проектом работает только один человек, то, конечно, заниматься

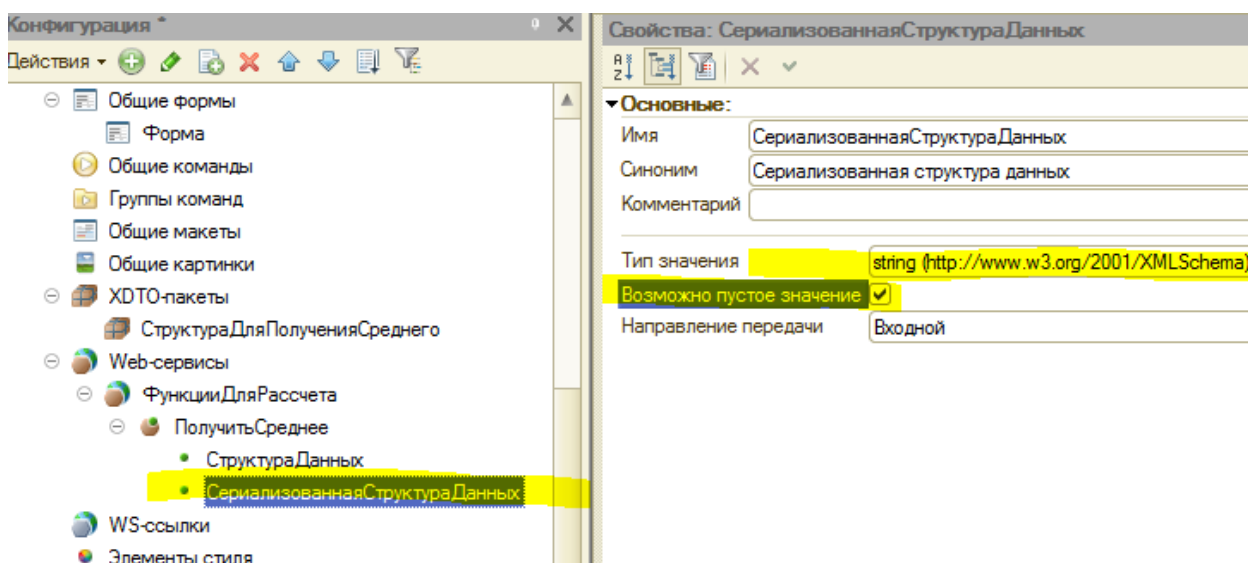
такими трудоемкими сценариями как создание XDTO-пакета, заполнение, проверка и т.д. не всегда нужно.

Также для двух конфигураций, между которыми нужно настроить обмен одного справочника, нет необходимости создавать XDTO-пакеты для обмена.

Платформа 1С имеет очень много механизмов, которые облегчают рутинную работу программиста. Один из таких механизмов – [СериализаторXDTO](#).

При помощи этого механизма возможно практически любой объект (таблица значений, структура, ссылка на объект и т.д.) преобразовать в xml, и преобразовать из xml обратно в объект.

Решим эту же задачу без использования XDTO-пакета. Так как *СериализаторXDTO* вернет нам строку, то добавим еще один параметр входящий у веб-сервиса – СериализованнаяСтруктураДанных. Этот параметр будет строковый, и у первого, и у второго параметра поставим возможность быть пустыми.



Как сериализовать и десериализовать объекты?

Для этого мы будем использовать две функции:

```

Функция Сериализовать (ОбъектСериализации)
    ДеревоВОбъектеXDTO = СериализаторXDTO.ЗаписатьXDTO (ОбъектСериализации) ;
    МойXML = Новый ЗаписьXML;
    МойXML.УстановитьСтроку ();
    
```

```
ФабрикаХДТО.ЗаписатьXML (МойXML, ДеревоВОбъектеХДТО);
Возврат МойXML.Закреть ();
КонецФункции
```

```
Функция Десериализовать (XMLСтруктураСериализованногоОбъекта)
ЧтениеXMLДанных = Новый ЧтениеXML;
ЧтениеXMLДанных.УстановитьСтроку (XMLСтруктураСериализованногоОбъекта);
ТЗ = СериализаторХДТО.ПрочитатьXML (ЧтениеXMLДанных);
ЧтениеXMLДанных.Закреть ();
Возврат ТЗ;
КонецФункции
```

В первую мы передаем объект, и она нам возвращает строку xml, а во вторую мы передаем строку xml, и она возвращает первоначальный объект.

На стороне сервера немного изменим код функции веб-сервиса:

```
Функция ПолучитьСреднее (СтруктураДанных, СериализованнаяСтруктураДанных)
Если ЗначениеЗаполнено (СтруктураДанных) Тогда
    Структура = СтруктураДанных.Проверить ();
Иначе
    Структура = Десериализовать (СериализованнаяСтруктураДанных);
КонецЕсли;

Сумма = 0;
Для каждого ЭлМассива Из Структура.МассивЧисел Цикл
    Сумма = Сумма + ЭлМассива;
КонецЦикла;
Возврат Сумма/Структура.КоличествоЭлементовВМассиве;
КонецФункции

Функция Десериализовать (XMLСтруктураСериализованногоОбъекта)
ЧтениеXMLДанных = Новый ЧтениеXML;
ЧтениеXMLДанных.УстановитьСтроку (XMLСтруктураСериализованногоОбъекта);
ТЗ = СериализаторХДТО.ПрочитатьXML (ЧтениеXMLДанных);
ЧтениеXMLДанных.Закреть ();
Возврат ТЗ;
КонецФункции
```

В функции *ПолучитьСреднее* сделана проверка на заполнение параметра. Если заполнен первый параметр, мы возьмем структуру из него, иначе значение второго параметра будет преобразовано в начальный объект.

Теперь напишем функцию, которая при помощи ws-ссылки будет передавать параметр. Однако обратите внимание, что сейчас в ws-ссылке нет второго параметра и необходимо перезаполнить ее.

Для этого нажмем на ней правой кнопкой: *Загрузить WSDL* – *OK*. После этого появится второй параметр (если вы перед этим не забыли обновить конфигурацию «Сервер»).

Код будет следующий:

```
&НаСервереБезКонтекста
Процедура ОтправитьСериализованнуюСтруктуруНаСервере ()
    СтруктураДанных = Новый
Структура ("КоличествоЭлементовВМассиве, Комментарий, МассивЧисел", 0, "", Новый
Массив);
    Соединение =
WSСсылки.ФункцииДляРасчета.СоздатьWSПрокси ("ФункцииДляРасчета", "ФункцииДляР
асчета", "ФункцииДляРасчетаSoap");
    Операция =
Соединение.ТочкаПодключения.Интерфейс.Операции.Получить ("ПолучитьСреднее");

    СтруктураДанных.КоличествоЭлементовВМассиве = 2;
    СтруктураДанных.Комментарий = "Тест";
    СтруктураДанных.МассивЧисел.Добавить (0.1);
    СтруктураДанных.МассивЧисел.Добавить (4);

    СериализованнаяСтруктураДанных = Сериализовать (СтруктураДанных);
    Ответ = Соединение.ПолучитьСреднее (Неопределено,
СериализованнаяСтруктураДанных);

КонецПроцедуры
&НаСервереБезКонтекста
Функция Сериализовать (ОбъектСериализации)
    ДеревоВОбъектеXDTO = СериализаторXDTO.ЗаписатьXDTO (ОбъектСериализации);
    МойXML = Новый ЗаписьXML;
    МойXML.УстановитьСтроку ();
    ФабрикаXDTO.ЗаписатьXML (МойXML, ДеревоВОбъектеXDTO);
    Возврат МойXML.Закрыть ();
КонецФункции
```

Обратите внимание, мы просто создали структуру данных, заполнили ее и сериализовали, потом, без создания фабрики и определения типов, передали на сервер.

Так как это простой тип – нет необходимости его описывать, программа сама постарается привести его к типу, который указан на сервере. Если не получится, то будет исключение.

Как выглядит сериализованная нами структура?

```
<Structure xmlns="http://v8.1c.ru/8.1/data/core"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Property name="КоличествоЭлементовВМассиве">
```

```
<Value xsi:type="xs:decimal">2</Value>
</Property>
<Property name="Комментарий">
  <Value xsi:type="xs:string">Тест</Value>
</Property>
<Property name="МассивЧисел">
  <Value xsi:type="Array">
    <Value xsi:type="xs:decimal">0.1</Value>
    <Value xsi:type="xs:decimal">4</Value>
  </Value>
</Property>
</Structure>
```

Видим, что это уже знакомые нам пространства имен и типы из этих пространств.

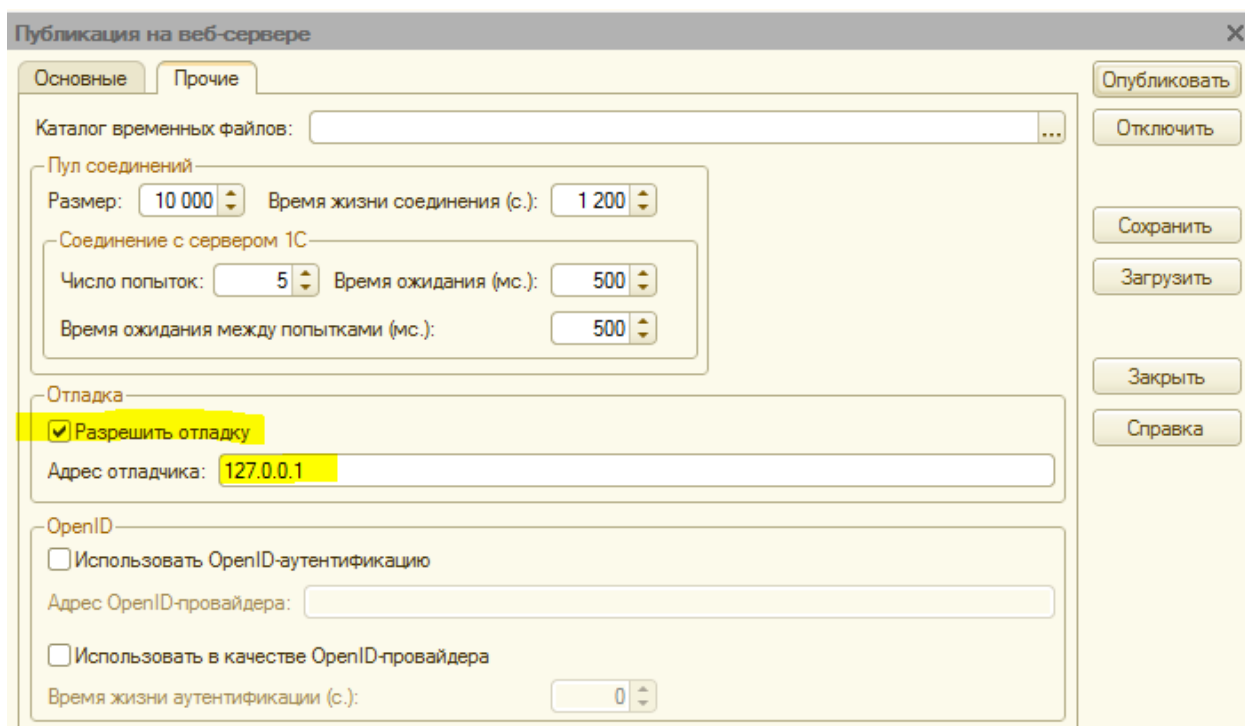
Точно так же можно сериализовать, например, таблицу значений *Номенклатура*, *Характеристика*, *Количество*, *Цена*, *Сумма* и т.д. Передавать эту таблицу в другую базу и в другой базе, после десериализации, работать как с первоначальной таблицей.

При этом есть два условия – имена метаданных должны совпадать и объекты грузятся по *UID*. Т.е. если мы сделаем тоже самое между распределенными базами данных, получится, что мы словно работаем в одной базе.

Отладка веб-сервиса

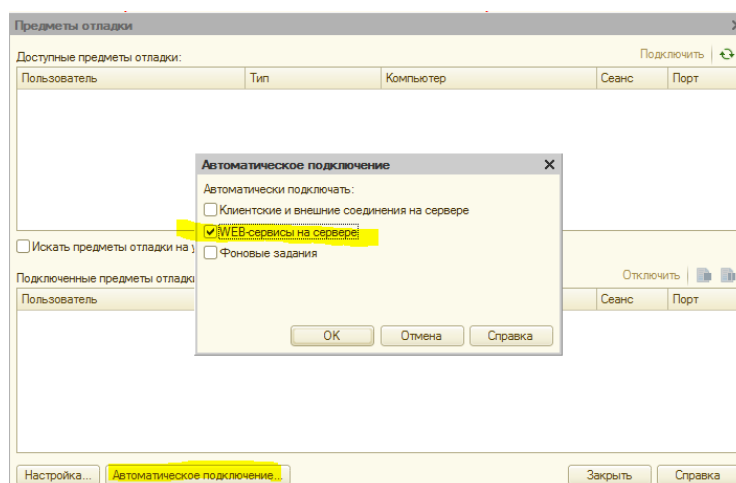
Те, кто пытался сделать отладку на веб-сервисе могли заметить, что она не работает. Это действительно так, но ее можно включить.

Для этого нужно перейти в меню *Администрирование – Публикация на веб-сервере* – переключится на вторую закладку, заполнить поле и поставить флаг «Разрешить отладку»:



Не забудьте после этого перезагрузить *Apache*. Если база не файловая, то надо ее перевести в режим отладки:

1. Остановить службу *1C:Enterprise 8.2 Server Agent*
2. Изменить ветку в реестре:
`[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\1C:Enterprise 8.2 Server Agent\]` для параметра *ImagePath* добавляем *-debug* и сохраняем.
 Получается что-то вроде такого: «*C:\Program Files\1cv82\8.2.19.72\bin\ragent.exe*» -*srvc -agent -regport 1541 -port 1540 -range 1560:1591 -d «C:\Program Files\1cv82\srvinfo» -debug*
3. Запустить службу *1C*
4. В конфигураторе зайти в меню *Отладка – Подключение – Автоматическое подключение* – поставить галочку «*WEB сервисы на сервере*»:



Итоги

Структурируем весь модуль. Теперь мы знаем, что есть возможность обмениваться информацией с сервером без файлов, почты и прочего. После вызова сервера, мы сразу будем знать о том, успешно ли принял сервер данные или была некая ошибка.

Данные на сервер можно передавать сериализуя их. Однако в этом случае обязательно надо проверять их правильность на сервере. Этот подход удобен, если программист работает один над всем проектом.

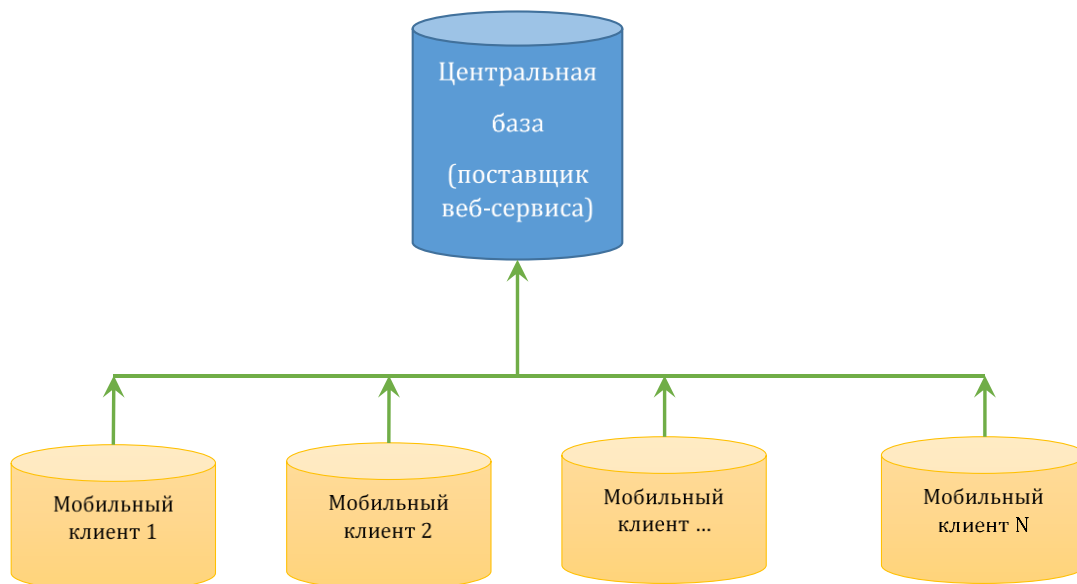
С другой стороны, если работа идет в команде, то тут конечно, необходимо нормировать все данные. Для этого можно использовать XDTO-пакет. Этот пакет необходимо создать на сервере. На клиенте вся структура пакета будет получена из wsdl-описания, причем не важно используется статическая ws-ссылка или динамическая. Стоит заметить, что статическая работает быстрее, но в случае изменения веб-сервиса, требуется обновление ссылки.

Перед Вами может стоять задача по синхронизации с внешними системами. Например, с веб-сайта в 1С должны выгружаться заказы. Тогда сайт может использовать для идентификации UID документа, вручную описать xml-структуру документа, и тогда в 1С можно десериализовать эту структуру в готовый документ.

И точно так же, выгружая документ на сайт, можно его просто сериализовать в xml и сайт, прося этот файл, будет его загружать. Это очень важно. Так как в этой структуре

описываются все свойства данных и программисту 1С не потребуется объяснять веб-разработчику, что за колонки есть, какие у них типы, максимальная длина и т.д.

В случае применения мобильной платформы в качестве клиента используется следующая схема:



Эта схема должна напоминать, что только центральный сервер может выступать в роли поставщика веб-сервиса, а мобильные – только в роли потребителей.

Только мобильное устройство может инициализировать вызов сервера и сервер ответит на него. Обратная ситуация, когда сервер инициализирует вызов мобильного устройства, передает ему некую информацию и получает ответ – невозможна.

То есть мобильные устройства, в том числе, не могут обмениваться данными друг с другом напрямую

Это касается только веб-сервисов. Если настроить обмен через почту, ftp и т.д., то, конечно, мобильные клиенты могут обмениваться информацией между собой.

Если Вам интересны аналогичные материалы

Рекомендуем пройти регистрацию на материалы по мобильной платформе:

kursy-po-1c.ru/mobile-apps

Регистрация для получения текущих и будущих бесплатных материалов проекта:

kursy-po-1c.ru/free


Дополнительные материалы

Все статьи проекта **Курсы-по-1С.рф**: <http://курсы-по-1с.рф/blog/articles/>

Бесплатные материалы проекта **Курсы-по-1С.рф**: <http://курсы-по-1с.рф/free/>

Бесплатный курс «Программирование в 1С – за 21 день»:

<http://курсы-по-1с.рф/prog1C-21days/lp1/>




Программирование в 1С - за 21 день

Бесплатный курс: Вы создадите систему с торговлей, бухгалтером, зарплатой и CRM


Курсы по программированию в 1С v.8

Базовый и Продвинутой курсы по Программированию на Платформе 1С 8
<http://www.Spec8.ru/>



Базовый курс по программированию в 1С v.8

Курс про **готовые приемы и решения 90% задач** по программированию в 1С



Продвинутой курс по программированию в 1С v.8

Больше, чем Вы можете себе представить
Детальнее требований на **1С:Специалист**

