Модули, директивы препроцессора и компиляции

В 1С ебанутые правила, связанные с модульностью. Понять и простить.

Модули служат для хранения кода. Несколько регламентированных точек размещения модулей, однако точка размещения кода для решения конкретной задачи регламентирована на уровне "можно и здесь, а можно и где-то там", поэтому финальная точка размещения зависит от фазы луны в момент создания кода разработчиком. То есть правила вроде есть, но их можно не исполнять. Однако важный нюанс: в каждой точке размещения доступны разные глобальные переменные, поэтому при написании кода набор доступных переменных нужно обязательно уточнять. Часто в интернетах приводят код без указания размещения модуля, потом при вопросе "А у меня не работает" сначала пара страниц троллинга типа они пиздец какие охуевшие спецы, потом кто-то снисходит до лошары. Хотя данное разделение не несет практической цели для разработчика конфигурации кроме искусственного усложнения и скорее артефакт изначально упрощенной архитектуры. Как деление подпрограмм на процедуры и функции.

Интересная статья по модулям, однако есть ряд недосказанных моментов.

Для подпрограмм всех модулей актуальны директивы препроцессора и компиляции. Модули можно сгруппировать по следующим правилам (группировка субъективная на основании изучения различных источников):

Группа	Размещение модулей	Базовое назначение	Подробное описание
Общие	Конфигурация - Общие - Общие модули	Хранение общего кода, доступного из остальных модулей. Настроить обработку событий нельзя.	В этой статье.

Группа	Размещение модулей	Базовое назначение	Подробное описание
Модули конфигурации	Конфигурация - Свойства. Модуль приложения Модуль внешнего соединения Модуль сеанса	Обработка общих событий платформы. Размещение подпрограмм обработки событий платформы. Например запуска приложения или внешнего вызова.	Управление отображением и событиями платформы
Модули классов	Конкретный класс. Модуль формы Модуль объекта Модуль менеджера	События, связанные с конкретным классом. Например нажатие кнопки "Показать группы" в окне списка справочника Номенклатура.	Объекты конфигурации

Структура модуля:
Раздел определения переменных - от начала текста модуля до первого оператора Процедура, Функция, или любого исполняемого оператора. Только операторы объявления переменных Перем.
Раздел процедур - от первого Процедура / Функция до любого исполняемого оператора вне тела описания процедур или функций.
Раздел основной программы - от первого исполняемого оператора вне тела последней процедуры или функции до конца модуля. Могут находиться только исполняемые операторы. Исполняется в момент инициализации модуля. Обычно в разделе основной программы имеет смысл размещать операторы инициализации переменных какими-либо конкретными значениями, которые необходимо провести до первого вызова любой из процедур или функций модуля.

Свойства модуля:

- Способ обращения к подпрограммам
  - «Глобальный». Возможность вызова подпрограмм без указания имени модуля. Имена должны быть уникальными в разрезе всего глобального контекста. Компилируются при старте системы. Чем больше таких модулей, тем медленнее программа будет стартовать. Если не указан, то компиляция будет выполняться в момент первого обращения к нему.
- Область видимости
  - «Клиент». Возможность исполнения подпрограмм модуля на клиенте;
  - «Сервер». Возможность исполнения подпрограмм модуля на сервере;
  - «Внешнее соединение». Возможность исполнения подпрограмм модуля через подключение внешнего источника;
- Возможности со стороны подпрограмм модуля
  - «Вызов сервера». Возможность подпрограмм модуля вызывать сервер, выполняясь на клиенте;
  - «Привилегированный». При работе кода процедур модуля не проверяет права доступа. Вызвать общий модуль с такой настройкой можно только на сервере.

Настройки «Клиент» и «Внешнее соединение» будут сброшены.
Это необходимо, если требуется массовая обработка данных. Контроль прав доступа увеличивает время обращения к базе данных .

- Кэширование
  - Для не глобальных Общих модулей. Варианты: «Повторное использование». Варианты: «Не использовать», «На время сеанса», «На время вызова». При многократном вызове одной процедуры система может использовать рассчитанные ранее данные в рамках процедуры (вызов) или жизни всего сеанса (запуска 1С).
    Цель – ускорить повторные вызовы. Имеет смысл только для тех функций, результат которых зависит исключительно от входных параметров. Если выбрано значение соответствующего параметра На время вызова, то кэш будет действовать до тех пор, пока работает та процедура, откуда был сделан вызов метода Общего модуля. Если выбрано значение На время сеанса, то условно считается, что кэш будет действовать, пока пользователь работает. Существуют временнЫе ограничения, очистка кэша происходит автоматически через 20 минут после попадания значения в кэш. Использовать осознанно.

Директивы препроцессора и компиляции

Есть два режима, определяются при создании конфигурации: на локальном ПК (файловый) и на сервере 1С (клиент-серверный вариант). Файловый вариант объединяет в себе и код клиента, и код сервера, поэтому смысла от деления кода при помощи директив препроцессора нет.

При запуске конфигурации на выполнение производится загрузка и компиляция конфигурации. Алгоритм компиляции:

- Экземпляры всех общих модулей создаются как на серверной, так и на клиентской стороне. Препроцессор используя директивы препроцессора и области видимостей модулей создает несколько слегка отличающихся копий (в соответствии с количеством платформ) кода. Директивы компиляции игнорируются. По умолчанию (без директив) все подпрограммы попадают во все копии. При наличии, код физически вырезается, и другая платформа не видит чужой код. Нужно для уменьшения объема кода и ускорения работы каждого из блоков.
- Код компилируется для каждой платформы. Используя директивы компиляции, формируются алгоритмы взаимодействия. Например, при вызове из подпрограммы &НаКлиенте подпрограммы &НаСервере форма упаковывается в контейнер (!), переправляется по каналу tcp\ip на сервер, распаковывается сервером, выполняется код целевой процедуры, форма запаковывается обратно в контейнер, передается на клиент, распаковывается клиентом, отображается на экране. В случае с "..БезКонтекста" платформа выполняет код "налегке", т.е. на сервер передаются только параметры функции, а обратно прилетает только результат выполнения. Директивой по умолчанию является &НаСервере.

Директивы препроцессора

При компиляции блоки распределяются в соответствии с директивами. Для указания разрешения использования процедур и функций общих модулей и модулей объектов используют инструкции препроцессора.
Синтаксис:

```
#Если <Логическое выражение> Тогда
#ИначеЕсли <Логическое выражение> Тогда
#Иначе
#КонецЕсли
```

<Логическое выражение> = [НЕ] <Символ препроцессора> [<Булева операция> [НЕ] <Символ препроцессора> [<Булева операция> [НЕ] <Символ препроцессора>]...]
  <Символ препроцессора> = {НаКлиенте | НаСервере | ТолстыйКлиентОбычноеПриложение | ТолстыйКлиентУправляемоеПриложение | Клиент | Сервер | ВнешнееСоединение }
  <Булева операция> = {И | ИЛИ}

Можно использовать И (AND), ИЛИ (OR), НЕ (NOT) Регистр букв не имеет значения.

Области

#Область, #КонецОбласти Нужны только для визуального сворачивания блоков кода. Могут быть вложенными.

```
#Область [<Имя области>]
#КонецОбласти
```

Условия

#Если (#If), #Тогда (#Then), #ИначеЕсли (#ElsIf), #Иначе (#Else), #КонецЕсли (#EndIf)
Можно организовывать выполнение различных процедур и функций на сервере приложения или на клиентском месте. Для того, чтобы процедура присутствовала и была вызвана на стороне сервера, фрагмент кода должен выглядеть следующим образом:

```
#Если Сервер Тогда
Процедура Проц1() Экспорт
КонецПроцедуры
#КонецЕсли
```

Если блок #Если Сервер Тогда ... #КонецЕсли включает только часть процедуры, то процедура будет присутствовать как на стороне клиента, так и на стороне сервера. Только на клиентской стороне она будет без той части, которая заключена в блок, поэтому

результат выполнения процедуры может зависеть от того, где обрабатывается вызов этой процедуры.

Для выполнения на клиентском месте в обычном и управляемом режиме:

```
#Если НаКлиенте Тогда
#КонецЕсли
```

Точки исполнения

Инструкция препроцессора НаКлиенте определена для всех клиентских приложений. Для тонкой подстройки модуля под конкретное клиентское приложение дополнительно введены инструкции ТолстыйКлиентОбычноеПриложение, ТолстыйКлиентУправляемоеПриложение, ТонкийКлиент и ВебКлиент, которые определены в соответствующих приложениях.

В обычном клиенте в обычном и управляемом режиме доступны НаКлиенте, Клиент, ТолстыйКлиентОбычноеПриложение, ТолстыйКлиентУправляемоеПриложение , НаСервере, Сервер.
В файловом варианте инструкции препроцессора #Если Сервер…, #Если Клиент…, #Если ТолстыйКлиентОбычноеПриложение или #Если ТолстыйКлиентУправляемоеПриложение… определены всегда, поэтому экземпляр кода будет присутствовать всегда.
В тонком клиенте доступны – ТонкийКлиент, НаКлиенте, Клиент.
На серверной части тонкого клиента – Сервер, НаСервере.
Во внешнем соединении – ВнешнееСоединение, НаСервере, Сервер.

Директивы компиляции

Каждая процедура и функция модуля формы, модуля команды и общего модуля управляемого приложения предваряется директивой компиляции, определяющей среду исполнения данной процедуры. Директива предваряется символом "&".

Запросы между клиентом и сервером за счет директив компиляции корректно формируются только внутри модуля обработчика. Во внешнем модуле нельзя вызвать подпрограммы разных контекстов. Нужно сначала перейти в соответствующий контекст в модуле, затем вызвать подпрограмму из общего модуля.

&НаКлиенте — определяет клиентскую процедуру (функцию); исполняется в среде клиентского приложения. В такой процедуре доступен клиентский контекст формы и вызовы любых процедур модуля.

&НаСервере — определяет серверную процедуру (функцию); исполняется в среде серверного приложения. В такой процедуре доступны данные формы, доступен серверный контекст формы и вызовы серверных и серверных внеконтекстных процедур модуля. При вызове такой процедуры данные формы будут передаваться с клиента на сервер и обратно (по окончанию вызова).

&НаСервереБезКонтекста — определяет серверную процедуру (функцию), исполняемую на сервере вне контекста формы. Переменные не могут быть внеконтекстными. В таких методах недоступен контекст формы (включая данные формы). Допустимыми являются вызовы только других внеконтекстных методов. При вызове этих методов не выполняется передача данных формы на сервер и обратно. Применение внеконтекстных методов позволяет существенно уменьшить объем передаваемых данных при вызове серверной процедуры из среды клиентского приложения;

&НаКлиентеНаСервереБезКонтекста — определяет процедуру (функцию), исполняемую в модуле формы на клиенте и на сервере, не имеющую доступа к контексту формы, данным формы, переменным, но имеющую доступ к процедурам и функциям общих модулей – серверных, не глобальных и серверных и клиентских одновременно. Сама процедура (функция) доступна для клиентский, серверных контекстных и неконтекстных процедур и функций модуля формы. Из серверных внеконтекстных методов формы допускается вызов серверных методов общих модулей;

&НаКлиентеНаСервере — определяет процедуру (функцию), исполняемую в модуле команды, выполняемую на клиенте и на сервере, имеющую доступ к процедурам и функциям общих модулей – серверных, не глобальных и серверных и клиентских одновременно, не имеющую доступ к переменным. Сама процедура (функция) доступна для клиентских серверных процедур и функций модуля команды.

Серверная процедура (функция), исполняемая вне контекста формы, (внеконтекстная) исполняется в среде серверного приложения. В такой процедуре (функции) недоступен контекст формы (включая данные формы). Допустимыми являются вызовы только других внеконтекстных процедур (функций). При вызове этих процедур (функций) не выполняется передача данных формы на сервер и обратно. Применение внеконтекстных процедур (функций) позволяет существенно уменьшить объем передаваемых данных при вызове серверной процедуры (функции) из среды клиентского приложения.

В модуле команды предопределенная процедура-обработчик ОбработатьКоманду должна предваряться директивой &НаКлиенте, так как выполнение команды происходит в клиентском приложении.

Модуль формы
В модуле формы доступны директивы компиляции – &НаКлиенте, &НаСервере, &НаСервереБезКонтекста, &НаКлиентеНаСервереБезКонтекста.
Модуль команды
В модуле команды доступны директивы компиляции – &НаКлиенте, &НаСервере, &НаКлиентеНаСервере.
Общий модуль
В общем модуле доступны директивы компиляции – &НаКлиенте, &НаСервере.

Общие модули.

Описание подпрограмм, вызываемых из различных мест. Для доступности подпрограмм извне необходимо ключевое слово Экспорт. То есть модули-то общие, но есть нюанс.

Даже при средней сложности конфигурации, подпрограммы нужно группировать в разные Общие модули. Для удобства имена Общих модулей должны отражать содержание описываемых в них процедур. При создании Общего модуля, правилом хорошего тона считается не указывать директивы компиляции. Т.е. доступность процедур и функций должна определяться свойствами самого модуля. При таком подходе в отдельных Общих модулях будут располагаться клиентские процедуры, и в отдельных Общих модулях – процедуры серверные. В названиях общих модулей рекомендуется это указывать. Например: РегламентныеПроцедурыСервер, РегламентныеПроцедурыКлиент.

Нельзя описывать глобальные переменные модуля и раздел основной программы. Т е все внешние переменные должны передаваться в подпрограммы извне.

Доступность модулей зависит от типа приложения и режима работы (Клиент, Сервер).

В отличие от остальных типов модулей (в них по умолчанию директива компиляции &НаСервере), если директиву компиляции для подпрограммы не указывать, то она будет скомпилирована во всех контекстах, определенных для модуля. Будет сделано несколько копий подпрограмм. Выбор конкретного скомпилированного экземпляра зависит от места вызова процедуры (по правилу ближайшего вызова). При этом следует учитывать, что код такой процедуры должен быть написан с учетом его доступности во всех определенных для модуля контекстах.

Модули с несколькими флагами компиляции на практике используются крайне редко. Это некоторые общие действия, доступные как на Клиенте, так и на Сервере. Обычно, это какие-то простейшие вычисления.

Revision #14
Created 4 January 2025 18:04:45 by Admin
Updated 17 January 2025 03:36:57 by Admin