Третья лекция

Встроенный язык платформы «1С:Предприятие 8»

Встроенный язык платформы «1С:Предприятие 8»

Встроенный язык является важной частью технологической платформы «1С:Предприятия 8», поскольку позволяет разработчику описывать собственные алгоритмы функционирования прикладного решения.

Встроенный язык имеет много общих черт с другими языками, такими как Pascal, Java Script, Basic, что облегчает его освоение начинающими разработчиками. Однако он не является прямым аналогом какого-либо из перечисленных языков.

Вот лишь некоторые, наиболее значимые особенности встроенного языка:

- предварительная компиляция — перед исполнением модули, содержащие текст на встроенном языке, преобразуются во внутренний код;
- кэширование скомпилированных модулей в памяти;
- мягкая типизация — тип переменной определяется типом значения, которое она содержит, и может изменяться в процессе работы;
- отсутствие программного описания объектов конфигурации — разработчик может использовать либо встроенные в платформу объекты, либо объекты, созданные системой в результате визуального конструирования прикладного решения.

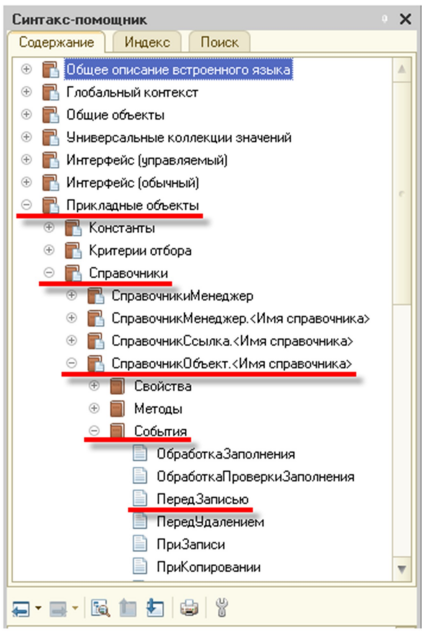
Встроенный язык является важной частью технологической платформы «1С:Предприятия 8», поскольку позволяет разработчику описывать собственные алгоритмы функционирования прикладного решения.
Встроенный язык имеет много общих черт с другими языками, такими как Pascal, Java Script, Basic, что облегчает его освоение начинающими разработчиками. Однако он не является прямым аналогом какого-либо из перечисленных языков.
Вот лишь некоторые, наиболее значимые особенности встроенного языка:
предварительная компиляция — перед исполнением модули, содержащие текст на встроенном языке, преобразуются во внутренний код;
кэширование скомпилированных модулей в памяти;
мягкая типизация — тип переменной определяется типом значения, которое она содержит, и может изменяться в процессе работы;
отсутствие программного описания объектов конфигурации — разработчик может использовать либо встроенные в платформу объекты, либо объекты, созданные системой в результате визуального конструирования прикладного решения.

Событийная ориентированность встроенного языка

Назначение встроенного языка в системе 1С:Предприятие определяется идеологией создания прикладных решений. Прикладные решения в 1С:Предприятии 8 не кодируются целиком. Большая часть прикладного решения создается разработчиком путем визуального конструирования — создания новых объектов конфигурации, задания их свойств, форм представления, взаимосвязей и пр. Встроенный язык используется лишь для того, чтобы определить поведение объектов прикладного решения, отличное от типового, и создать собственные алгоритмы обработки данных.

По этой причине модули, содержащие текст на встроенном языке, используются системой в конкретных, заранее известных ситуациях, которые могут возникнуть в процессе работы прикладного решения. Такие ситуации называются **событиями**. События могут быть связаны с функционированием объектов прикладного решения или с самим прикладным решением, как таковым.

Назначение встроенного языка в системе 1С:Предприятие определяется идеологией создания прикладных решений. Прикладные решения в 1С:Предприятии 8 не кодируются целиком. Большая часть прикладного решения создается разработчиком путем визуального конструирования — создания новых объектов конфигурации, задания их свойств, форм представления, взаимосвязей и пр. Встроенный язык используется лишь для того, чтобы определить поведение объектов прикладного решения, отличное от типового, и создать собственные алгоритмы обработки данных.
По этой причине модули, содержащие текст на встроенном языке, используются системой в конкретных, заранее известных ситуациях, которые могут возникнуть в процессе работы прикладного решения. Такие ситуации называются **событиями**. События могут быть связаны с функционированием объектов прикладного решения или с самим прикладным решением, как таковым.



Событийная ориентированность встроенного языка

Например, с функционированием объекта прикладного решения **Справочник** связан ряд событий, среди которых есть событие **ПередЗаписью**.

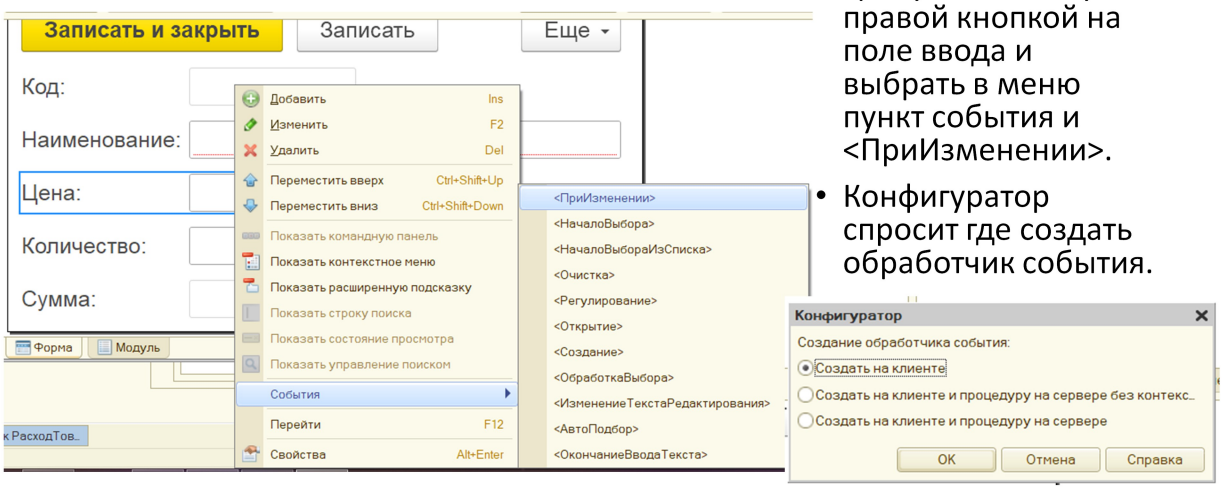
Это событие возникает непосредственно перед тем, как данные элемента справочника должны быть записаны в базу данных.

Например, с функционированием объекта прикладного решения **Справочник** связан ряд событий, среди которых есть событие **ПередЗаписью**.
Это событие возникает непосредственно перед тем, как данные элемента справочника должны быть записаны в базу данных. Разработчик, используя встроенный язык, может описать алгоритм, который, например, будет проверять корректность данных, введенных пользователем. Разместив этот алгоритм в соответствующем модуле, разработчик обеспечит то, что каждый раз, как пользователь будет выполнять запись элемента справочника, система будет выполнять созданный разработчиком алгоритм и проверять, не забыл ли пользователь заполнить обязательные реквизиты справочника.
Таким образом можно сказать, что встроенный язык является скриптовым языком для программирования бизнес-логики, а использование модулей на встроенном языке является событийно-зависимым, т. е. выполнение модулей происходит при возникновении определенных событий в процессе функционирования прикладного решения.

Обработка события

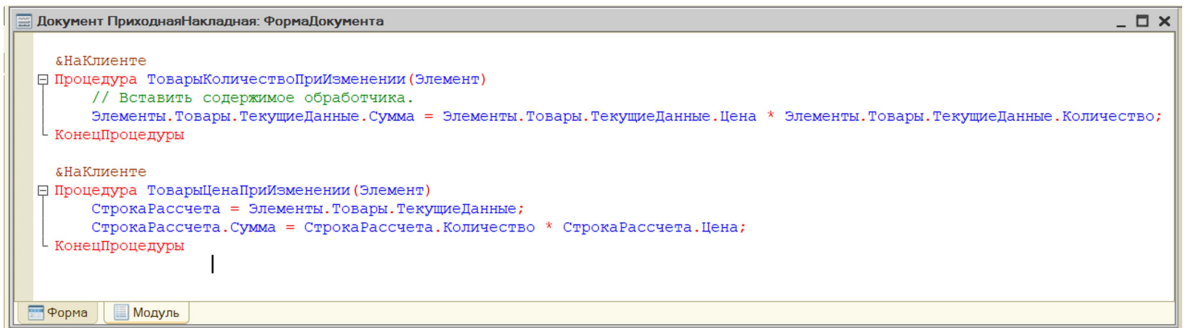
Для создания обработки события изменения суммы при изменении цены (или количества) товара.

- Требуется кликнуть правой кнопкой на поле ввода и выбрать в меню пункт события и <ПриИзменении>.
- Конфигуратор спросит где создать обработчик события.



Для создания обработки события изменения суммы при изменении цены (или количества) товара.
•Требуется кликнуть правой кнопкой на поле ввода и выбрать в меню пункт события и <ПриИзменении>.
•Конфигуратор спросит где создать обработчик события.

Обработка события



Например для изменения суммы при изменении количества или цены товара достаточно создать соответствующие процедуры на клиенте.

Например для изменения суммы при изменении количества или цены товара достаточно создать соответствующие процедуры на клиенте.

Обработка события

Документ РасходнаяНакладная: ФормаДокумента

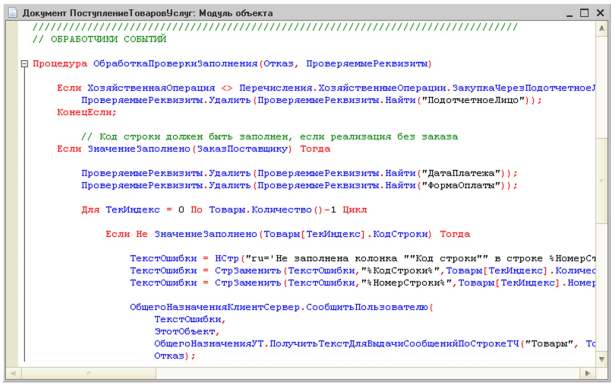
```
&НаСервереБезКонтекста
Функция ПолучитьЗначениеСкидки(Ссылка)
    Возврат Ссылка.ПроцентСкидки;
    // Вставить содержимое обработчика.
КонецФункции

&НаКлиенте
Процедура РасчетСуммы()
    // Вставить содержимое обработчика.
    СтрокаРасчета = Элементы.Товары.ТекущиеДанные;
    ЗначениеСкидки = ПолучитьЗначениеСкидки(СтрокаРасчета.Скидка);
    СтрокаРасчета.Сумма = СтрокаРасчета.Количество * СтрокаРасчета.Цена;
    СтрокаРасчета.СуммаСкидки = СтрокаРасчета.Сумма *  ЗначениеСкидки / 100 ;
    СтрокаРасчета.Всего = СтрокаРасчета.Сумма - СтрокаРасчета.СуммаСкидки;

КонецПроцедуры
```

Форма | Модуль

Если же нужно создать функцию возвращающую значение, то удобнее сделать это на сервере и вызывать в процедурах на клиенте.

Если же нужно создать функцию возвращающую значение, то удобнее сделать это на сервере и вызывать в процедурах на клиенте.

Выделение цветом синтаксических конструкций

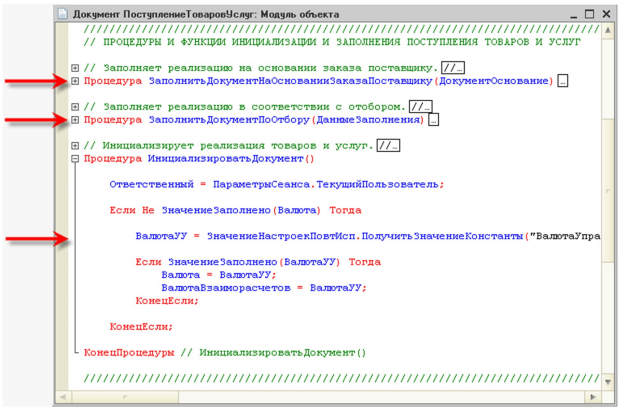


Для удобства редактирования текстов модулей редактор выделяет цветом элементы встроенного языка: ключевые слова, языковые константы, операторы, комментарии и пр.

Разработчик может использовать цвета выделения, установленные по умолчанию, или настроить их самостоятельно. В общем случае система сама отслеживает необходимость включения режима выделения цветом.

Для удобства редактирования текстов модулей редактор выделяет цветом элементы встроенного языка: ключевые слова, языковые константы, операторы, комментарии и пр.
Разработчик может использовать цвета выделения, установленные по умолчанию, или настроить их самостоятельно. В общем случае система сама отслеживает необходимость включения режима выделения цветом.

Группировка синтаксических конструкций



При просмотре модулей редактор позволяет объединять некоторые синтаксические конструкции языка в группы, сворачивать и разворачивать их. Использование группировки синтаксических конструкций позволяет лучше воспринимать различные части текста, а также переносить и копировать группы целиком

При просмотре модулей редактор позволяет объединять некоторые синтаксические конструкции языка в группы, сворачивать и разворачивать их. Использование группировки синтаксических конструкций позволяет лучше воспринимать различные части текста, а также переносить и копировать группы целиком

Области синтаксических конструкций

```
////////////////////////////////////////////////////////////////////////////////
// ПРОГРАММНЫЙ ИНТЕРФЕЙС
#Область ПрограммныйИнтерфейс

// Заполняет условия продаж в заказе поставщику //...
Процедура ЗаполнитьУсловияЗакупок(Знач УсловияЗакупок) Экспорт...

// Заполняет условия закупок по торговому соглашению с поставщиком //...
Процедура ЗаполнитьУсловияЗакупокПоУмолчанию(ПересчитатьЦены = Истина) Экспорт...

// Заполняет условия продаж по соглашению в заказе поставщику //...
Процедура ЗаполнитьУсловияЗакупокПоСоглашению(ПересчитатьЦены = Истина) Экспорт...

// Заполняет табличную часть Расхождения //...
Функция ЗаполнитьРасхождения(ЕстьИзлишки = Ложь, ЕстьНедостачи = Ложь) Экспорт...

#КонецОбласти

////////////////////////////////////////////////////////////////////////////////
// ОБРАБОТЧИКИ СОБЫТИЙ
ОбработчикиСобытий

////////////////////////////////////////////////////////////////////////////////
// СЛУЖЕБНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ
СлужебныеПроцедурыИФункции
```

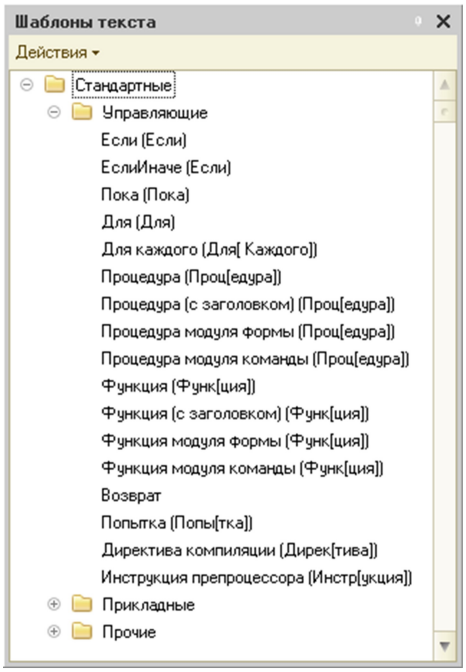
Разработчик может выделять произвольные области текста, группировать и сворачивать их подобно тому, как сворачиваются инструкции циклов, условий, процедур и функций.

Каждой области текста, которую выделяет разработчик, он может дать собственное имя. Это позволяет простым и понятным образом выделять части модуля, имеющие сходный смысл.

Области выделяются с помощью двух инструкций препроцессора: **#Область** и **#КонецОбласти**. Единственное назначение этих инструкций — обозначить группируемые и сворачиваемые строки модуля.

Области могут быть вложены друг в друга или в другие группируемые конструкции языка.

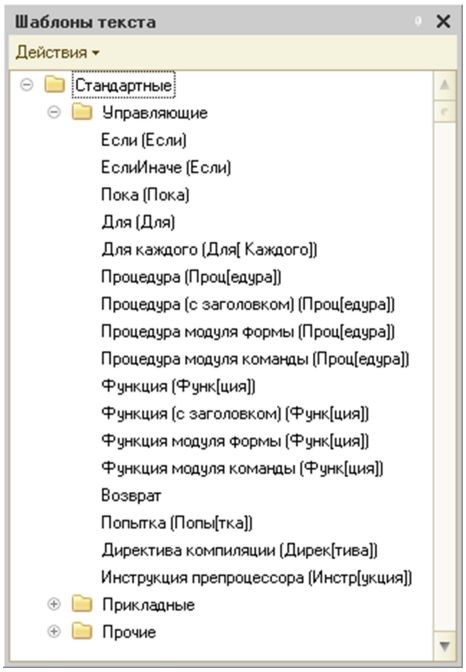
Разработчик может выделять произвольные области текста, группировать и сворачивать их подобно тому, как сворачиваются инструкции циклов, условий, процедур и функций.
Каждой области текста, которую выделяет разработчик, он может дать собственное имя. Это позволяет простым и понятным образом выделять части модуля, имеющие сходный смысл.
Области выделяются с помощью двух инструкций препроцессора: **#Область** и **#КонецОбласти**. Единственное назначение этих инструкций — обозначить группируемые и сворачиваемые строки модуля.
Области могут быть вложены друг в друга или в другие группируемые конструкции языка.



Шаблоны

Конфигуратор 1С:Предприятия 8 поддерживает возможность создания, сохранения и быстрой вставки часто используемых фрагментов текста. Такие фрагменты текста называются шаблонами, и для управления списком шаблонов разработчик может использовать специальное окно

Конфигуратор 1С:Предприятия 8 поддерживает возможность создания, сохранения и быстрой вставки часто используемых фрагментов текста. Такие фрагменты текста называются шаблонами, и для управления списком шаблонов разработчик может использовать специальное окно



Шаблоны

Кроме этого существует возможность непосредственной вставки любого шаблона путем выбора его из контекстного меню.

Шаблоны текста поддерживают использование в своей динамической части конструктора запросов и конструктора форматной строки.

Кроме этого существует возможность непосредственной вставки любого шаблона путем выбора его из контекстного меню.
Шаблоны текста поддерживают использование в своей динамической части конструктора запросов и конструктора форматной строки.