Предварительная обработка

Некоторые возможности предварительной обработки включены в **PlantUML** и доступн

Эти функциональные возможности очень похожи на препроцессор языка Си, за исклю специальный символ `#` был изменен на восклицательный знак `!`.

Определение переменной [=, ?=]

Хотя это не обязательно, мы настоятельно рекомендуем, чтобы имена переменных на

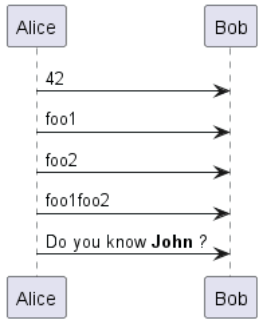
Существует три типа данных:

- **(int)***Целое число*;
- **(str)***String* - они должны быть заключены в одинарные или двойные кавычки;
- **JSON**(JSON) - они должны быть заключены в фигурные скобки.

(для определения и использования переменной JSON смотрите более подробную информа предварительной обработки-JSON)

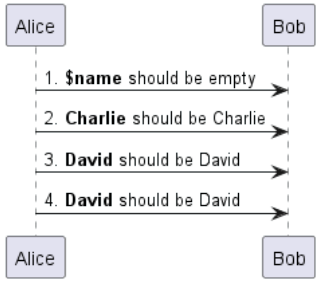
Переменные, созданные вне функции, являются **глобальными**, то есть вы можете пол отовсюду (в том числе из функций). Вы можете подчеркнуть это, используя необязател слово при определении переменной.

```
@startuml
!$a  = 42
!$ab = "foo1"
!$cd = "foo2"
!$ef = $ab + $cd
!$foo = { "name": "John", "age" : 30 }

Alice -> Bob : $a
Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
Alice -> Bob : Do you know **$foo.name** ?
@enduml
```



Вы также можете присвоить значение переменной, только если оно еще не определен синтаксиса: `!$a ?= "foo"`

```
@startuml
Alice -> Bob : 1. **$name** should be empty

!$name ?= "Charlie"
Alice -> Bob : 2. **$name** should be Charlie

!$name = "David"
Alice -> Bob : 3. **$name** should be David

!$name ?= "Ethan"
Alice -> Bob : 4. **$name** should be David
@enduml
```



Логическое выражение

Логическое представление [0 равно false]

- Целое число `0` означает `false`
- и любое ненулевое число (как `1`) или любая строка (как `"1"` , или даже `"0"`) означ

[Ссылка на QA-9702]

Логическая операция и оператор [&&, ||, ()]

Вы можете использовать логическое выражение в тесте с :

- *скобки* `()` ;
- *и оператор* `&&` ;
- *или оператор* `||` .

(Смотрите следующий пример в `if` тесте.)

Встроенные логические функции [%false(), %true(), %not(<exp>)]

Для удобства вы можете использовать эти встроенные логические функции:

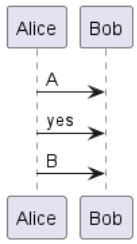
- `%false()`
- `%true()`
- `%not(<exp>)`

[Смотрите также Встроенные функции]

Условия [!if, !else, !elseif, !endif]

- Вы можете использовать выражение в условии.
- *также реализованыelse и elseif*

```
@startuml
!$a = 10
!$ijk = "foo"
Alice -> Bob : A
!if ($ijk == "foo") && ($a+10>=4)
Alice -> Bob : yes
!else
Alice -> Bob : This should not appear
!endif
Alice -> Bob : B
@enduml
```

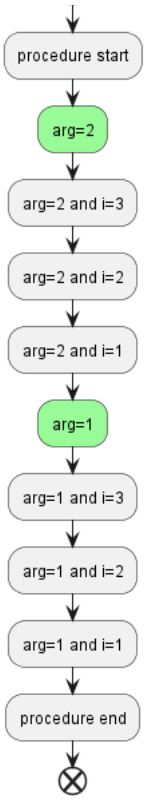


Цикл While [!while, !endwhile]

Вы можете использовать ключевые слова `!while` и `!endwhile` для создания повторяю

Цикл While (на диаграмме действий)

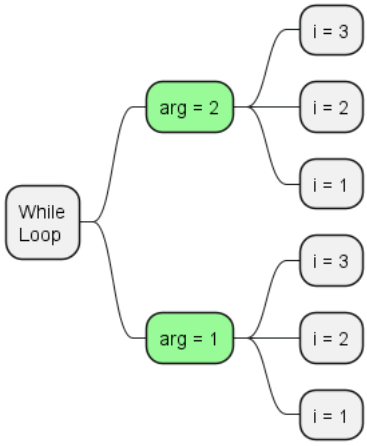
```
!procedure $foo($arg)
  :procedure start;
  !while $arg!=0
    !$i=3
    #palegreen:arg=$arg;
    !while $i!=0
      :arg=$arg and i=$i;
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
  :procedure end;
!endprocedure

start
$foo(2)
end
@enduml
```



[Адаптировано из QA-10838]

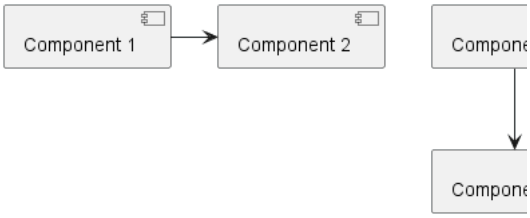
Цикл While (на диаграмме Mindmap)

```
@startmindmap
!procedure $foo($arg)
  !while $arg!=0
    !$i=3
    **[#palegreen] arg = $arg
    !while $i!=0
      *** i = $i
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
!endprocedure

*:While
Loop;
$foo(2)
@endmindmap
```



Цикл While (на схеме компонента / развертывания)

```
@startuml
!procedure $foo($arg)
  !while $arg!=0
    [Component $arg] as $arg
    !$arg = $arg - 1
  !endwhile
!endprocedure

$foo(4)

1->2
3-->4
@enduml
```



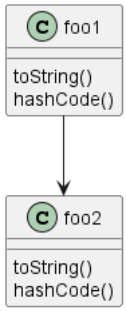
[Ссылка на QA-14088]

Процедура [!procedure, !endprocedure]

- Имена процедур *должны* начинаться с `$`
- Имена аргументов *должны* начинаться с `$`
- Процедуры могут вызывать другие процедуры

Пример:

```
@startuml
!procedure $msg($source, $destination)
  $source --> $destination
!endprocedure

!procedure $init_class($name)
  class $name {
    $addCommonMethod()
  }
!endprocedure


!procedure $addCommonMethod()
  toString()
  hashCode()
!endprocedure


$init_class("foo1")
$init_class("foo2")
$msg("foo1", "foo2")
@enduml
```



Переменные, определенные в процедурах, являются **локальными**. Это означает, что г
по завершении процедуры.

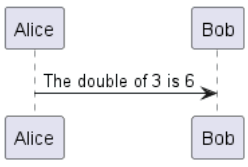
Возвращаемая функция [!функция, !конечная ф

Функция возврата не выводит никакого текста. Она просто определяет функцию, котор

- непосредственно в определении переменной или в тексте диаграммы
- из других возвращаемых функций
- из процедур

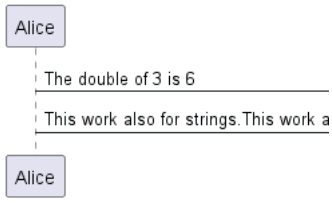
- Имя функции *должно* начинаться с `$`
- Имена аргументов *должны* начинаться с `$`

```
@startuml
!function $double($a)
!return $a + $a
!endfunction

Alice -> Bob : The double of 3 is $double(3)
@enduml
```



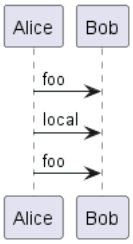
Простое определение функции можно сократить в одну строку:

```
!function $double($a) !return $a + $a

Alice -> Bob : The double of 3 is $double(3)
Alice -> Bob : $double("This work also for strings.")
@enduml
```



Как и в процедуре (функция void), переменные по умолчанию являются локальными (о
выходе из функции). Однако вы можете получить доступ к глобальным переменным из
можете использовать ключевое слово `local` для создания локальной переменной, есл
глобальная переменная с таким же именем.

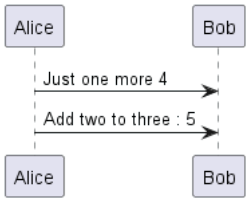
```
@startuml
!function $dummy()
!local $ijk = "local"
!return "Alice -> Bob : " + $ijk
!endfunction

!global $ijk = "foo"

Alice -> Bob : $ijk
$dummy()
Alice -> Bob : $ijk
@enduml
```



Значение аргумента по умолчанию

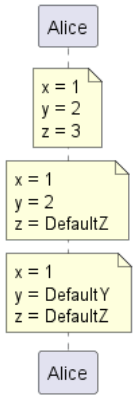
Как в процедурах, так и в функциях возврата вы можете определить значения по умолч

```
@startuml
!function $inc($value, $step=1)
!return $value + $step
!endfunction

Alice -> Bob : Just one more $inc(3)
Alice -> Bob : Add two to three : $inc(3, 2)
@enduml
```



Только аргументы в конце списка параметров могут иметь значения по умолчанию.

```
@startuml
!procedure defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
  x = $x
  y = $y
  z = $z
end note
!endprocedure

defaulttest(1, 2, 3)
defaulttest(1, 2)
defaulttest(1)
@enduml
```



Процедура или функция без кавычек [!без кавы

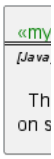
По умолчанию вы должны заключать в кавычки при вызове функции или процедуры. М
ключевое слово `unquoted` , чтобы указать, что функция или процедура не требуют кавы

```
@startuml
!unquoted function id($text1, $text2="FOO") !return $text1 + $text2

alice -> bob : id(aa)
alice -> bob : id(ab,cd)
@enduml
```

Ключевые слова аргументы

Как и в Python, вы можете использовать аргументы ключевых слов :

```
@startuml

!unquoted procedure $element($alias, $description="", $label="", $technolo
rectangle $alias as "
<color:$colour><<$alias>></color>
==$label==
//<size:$size>[$technology]</size>//

  $description"
!endprocedure

$element(myalias, "This description is %newline()on several lines", $size=
@enduml
```

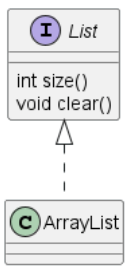


Включая файлы или URL [!include, !include_many,

Используйте директиву `!include` , чтобы включить файл в вашу диаграмму. Используя
включить файл из Интернета / Интрасети. Также можно получить доступ к защищенны
описано в разделе Аутентификация по URL.

Представьте, что у вас есть тот же самый класс, который отображается на многих диаг
дублирования описания этого класса вы можете определить файл, содержащий описа

```
@startuml

interface List
List : int size()
List : void clear()
List <|.. ArrayList
@enduml
```



Список файлов.iuml

```
List : int size()
List : void clear()
```

Файл `List.iuml` может быть включен во многие диаграммы, и любая модификация в эт диаграммы, которые его включают.

Вы также можете поместить несколько `@startuml/@enduml` текстовых блоков во включа какой блок вы хотите включить, добавив, `!0` где `0` - номер блока. Обозначение `!0` об диаграмму.

Например, если вы используете `!include foo.txt!1` , будет включен второй `@startuml/@` `foo.txt` .

Вы также можете присвоить идентификатор некоторому `@startuml/@enduml` текстовому файле, используя `@startuml(id=MY_OWN_ID)` синтаксис, а затем включить добавление бл включении файла, используя что-то вроде `!include foo.txt!MY_OWN_ID` .

По умолчанию файл может быть включен только один раз. Вы можете использовать ! `!include` , если хотите включить какой-либо файл несколько раз. Обратите внимание, ч `!include_once` директива, которая выдает ошибку, если файл включается несколько ра

Включая подраздел [!startsub, !endsub, !includesu

Вы также можете использовать `!startsub NAME` и `!endsub` для указания разделов текст из других файлов, используя `!includesub` . Например:

file1.puml:

```
@startuml

A -> A : stuff1
!startsub BASIC
B -> B : stuff2
!endsub
C -> C : stuff3
!startsub BASIC
D -> D : stuff4
!endsub
@enduml
```

файл1.puml будет отображаться точно так, как если бы он был:

```
@startuml

A -> A : stuff1
B -> B : stuff2
C -> C : stuff3
D -> D : stuff4
@enduml
```

Однако это также позволило бы вам создать другой файл2.puml, подобный этому:

file2.puml

```
@startuml

title this contains only B and D
!includesub file1.puml!BASIC
@enduml
```

Этот файл будет отображаться точно так, как если бы:

```
@startuml

title this contains only B and D
B -> B : stuff2
D -> D : stuff4
@enduml
```

Встроенные функции [%]

Некоторые функции определены по умолчанию. Их название начинается с `%`

Имя	Описание	Пример
`%chr`	Возвращает символ из заданного значения в Юникоде	`%chr(65)`
`%darken`	Возвращает затемнение заданного цвета с некоторым соотношением	`%darken("red", 20)`
`%date`	Извлеките текущую дату. Вы можете указать необязательный формат для даты	`%date("yyyy.MM.dd' at 'HH:mm")`
	Вы можете указать другое необязательное время (в формате эпохи)	`%date("YYYY-MM-dd", %now() + 1*24*3600)`
`%dec2hex`	Возвращает шестнадцатеричную строку (String) с десятичным значением (Int)	`%dec2hex(12)`
`%dirpath`	Извлеките текущий dirpath	`%dirpath()`
`%feature`	Проверьте, доступна ли какая-либо функция в текущей версии PlantUML	`%feature("theme")`
`%false`	Возвращайте всегда `false`	`%false()`
`%file_exists`	Проверьте, существует ли файл в локальной файловой системе	`%file_exists("c:/foo/dummy.txt")`
`%filename`	Извлеките текущее имя файла	`%filename()`
`%function_exists`	Проверьте, существует ли функция	`%function_exists("$some_function")`
`%get_variable_value`	Извлеките некоторое значение переменной	`%get_variable_value("$my_variable")`
`%getenv`	Извлеките значение переменной среды	`%getenv("OS")`
`%hex2dec`	Возвращает десятичное значение (Int)	`%hex2dec("d")` или `%hex2dec(d)`

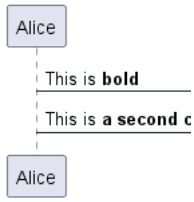
`%hsl_color`	Верните цвет RGBa из цвета HSL `%hsl_color(h, s, l)` или `%hsl_color(h, s, l, a)`	`%hsl_color(120, 100, 50)`
`%intval`	Преобразуйте строку в Int	`%intval("42")`
`%is_dark`	Проверьте, является ли цвет темным	`%is_dark("#000000")`
`%is_light`	Проверьте, является ли цвет светлым	`%is_light("#000000")`
`%lighten`	Возвращает более светлый цвет заданного цвета с некоторым соотношением	`%lighten("red", 20)`
`%load_json`	Загружайте данные JSON из локального файла или внешнего URL	`%load_json("http://localhost:7778/manage`
`%lower`	Возвращает строку в нижнем регистре	`%lower("Hello")`
`%newline`	Возвращает перевод строки	`%newline()`
`%not`	Возвращает логическое отрицание выражения	`%not(2+2==4)`
`%now`	Возвращает текущее время эпохи	`%now()`
`%ord`	Возвращает значение в Юникоде из заданного символа	`%ord("A")`
`%lighten`	Возвращает более светлый цвет заданного цвета с некоторым соотношением	`%lighten("red", 20)`
`%reverse_color`	Измените цвет на противоположный с помощью RGB	`%reverse_color("#FF7700")`
`%reverse_hsluv_color`	Измените цвет на противоположный с помощью HSLuv	`%reverse_hsluv_color("#FF7700")`
`%set_variable_value`	Установите глобальную переменную	`%set_variable_value("$my_variable", "som`
`%size`	Возвращает размер любой строки или структуры JSON	`%size("foo")`
`%string`	Преобразуйте выражение в строку	`%string(1 + 2)`
`%strlen`	Вычислите длину	`%strlen("foo")`

`%strpos`	Поиск подстроки в строке	`%strpos("abcdef", "ef")`
`%substr`	Извлеките подстроку. Принимает 2 или 3 аргумента	`%substr("abcdef", 3, 2)`
`%true`	Возвращайте всегда `true`	`%true()`
`%upper`	Возвращает строку в верхнем регистре	`%upper("Hello")`
`%variable_exists`	Проверьте, существует ли переменная	`%variable_exists("$my_variable")`
`%version`	Верните PlantUML текущую версию	`%version()`

Ведение журнала [!log]

Вы можете использовать `!log`, чтобы добавить некоторые выходные данные журнала
Это никак не влияет на саму диаграмму. Однако эти журналы печатаются в потоке выво
может быть полезно для целей отладки.

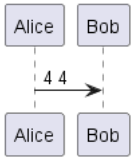
```
@startuml
!function bold($text)
!$result = "<b>"+ $text +"</b>"
!log Calling bold function with $text. The result is $result
!return $result
!endfunction

Alice -> Bob : This is bold("bold")
Alice -> Bob : This is bold("a second call")
@enduml
```



Дамп памяти [!dump_memory]

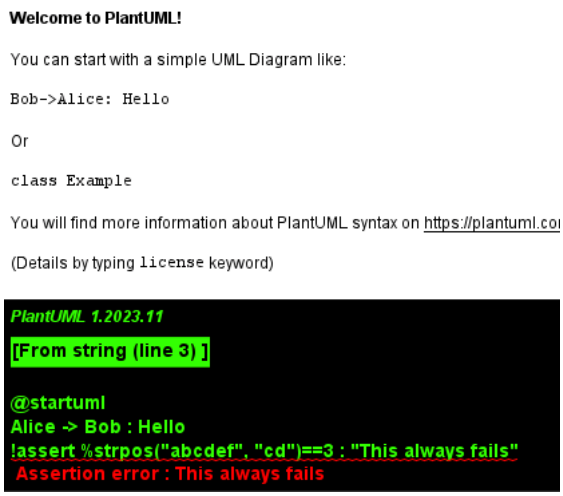
Вы можете использовать `!dump_memory` для выгрузки полного содержимого памяти при
После `!dump_memory` можно поставить необязательную строку. Это никак не влияет на
быть полезно для целей отладки.

```
@startuml
!function $inc($string)
!$val = %intval($string)
!log value is $val
!dump_memory
!return $val+1
!endfunction

Alice -> Bob : 4 $inc("3")
!unused = "foo"
!dump_memory EOF
@enduml
```



Вы можете поместить утверждения в свою диаграмму.

```
@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fails"
@enduml
```



Создание пользовательской библиотеки [!impor

Можно упаковать набор включенных файлов в один zip- или .jar-архив. Затем этот отде импортировать в вашу схему, используя `!import` директиву.

После импорта библиотеки вы можете `!include` создать файл из этого единственного

Пример:

```
@startuml

!import /path/to/customLibrary.zip
' This just adds "customLibrary.zip" in the search path

!include myFolder/myFile.iuml
' Assuming that myFolder/myFile.iuml is located somewhere
' either inside "customLibrary.zip" or on the local filesystem

...
```

Путь поиска

Вы можете указать свойство java `plantuml.include.path` в командной строке.

Например:

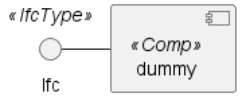
```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

Обратите внимание, что параметр this -D должен быть установлен перед параметром параметра -jar будут использоваться для определения констант в препроцессоре plant

Конкатенация аргументов [##]

К аргументу макроса можно добавить текст, используя `##` синтаксис.

```
@startuml
!unquoted procedure COMP_TEXTGENCOMP(name)
[name] << Comp >>
interface Ifc << IfcType >> AS name##Ifc
name##Ifc - [name]
!endprocedure
COMP_TEXTGENCOMP(dummy)
@enduml
```

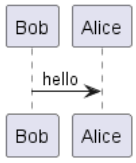


Динамический вызов [`%invoke_procedure()`, `%call_`

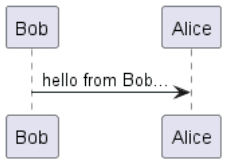
Вы можете динамически вызывать процедуру с помощью специальной `%invoke_proced` процедура принимает в качестве первого аргумента имя фактической вызываемой пр следующие аргументы копируются в вызываемую процедуру.

Например, вы можете иметь:

```
@startuml
!procedure $go()
  Bob -> Alice : hello
!endprocedure

!$wrapper = "$go"

%invoke_procedure($wrapper)
@enduml
```

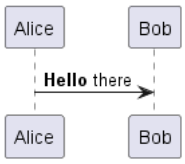



```
@startuml
!procedure $go($txt)
  Bob -> Alice : $txt
!endprocedure

%invoke_procedure("$go", "hello from Bob...")
@enduml
```



Для функций возврата вы можете использовать соответствующую специальную функц

```
@startuml
!function bold($text)
!return "<b>"+ $text +"</b>"
!endfunction

Alice -> Bob : %call_user_func("bold", "Hello") there
@enduml
```



Оценка сложения в зависимости от типов данны

Оценка `$a + $b` в зависимости от типа `$a` или `$b`

```
title
<#LightBlue>|= |=  $a |=  $b |=  <U+0025>string($a + $b)|
<#LightGray>| type | str | str | str (concatenation) |
| example |= "a" |= "b" |= %string("a" + "b") |
<#LightGray>| type | str | int | str (concatenation) |
| ex.|= "a" |=  2  |= %string("a" + 2)   |
<#LightGray>| type | str | int | str (concatenation) |
| ex.|=  1  |= "b" |= %string(1 + "b")   |
<#LightGray>| type | bool | str | str (concatenation) |
| ex.|= <U+0025>true() |= "b" |= %string(%true() + "b") |
<#LightGray>| type | str | bool | str (concatenation) |
| ex.|= "a" |= <U+0025>false() |= %string("a" + %false()) |
<#LightGray>| type |  int  |  int | int (addition of int) |
| ex.|=  1  |=  2  |= %string(1 + 2)     |
<#LightGray>| type |  bool  |  int | int (addition) |
| ex.|= <U+0025>true() |= 2 |= %string(%true() + 2) |
<#LightGray>| type |  int  |  bool | int (addition) |
| ex.|=  1  |= <U+0025>false() |= %string(1 + %false()) |
<#LightGray>| type |  int  |  int | int (addition) |
| ex.|=  1  |=  <U+0025>intval("2")  |= %string(1 + %intval("2")) |
end title
@enduml
```

	$a	$b	%str
type	str	str	str (
example	"a"	"b"	ab
type	str	int	str (
ex.	"a"	2	a2
type	str	int	str (
ex.	1	"b"	1b
type	bool	str	str (
ex.	%true()	"b"	1b
type	str	bool	str (
ex.	"a"	%false()	a0
type	int	int	int (
ex.	1	2	3
type	bool	int	int (
ex.	%true()	2	3
type	int	bool	int (
ex.	1	%false()	1
type	int	int	int (
ex.	1	%intval("2")	3

Предварительная обработка JSON

Вы можете расширить функциональность текущей предварительной обработки с пом
предварительной обработки JSON:

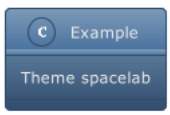
- Определение переменной в формате JSON
- Доступ к данным JSON
- Цикл над массивом JSON

(Смотрите более подробную информацию на странице предварительной обработки-JSO

Включая тему [!theme]

Используйте `!theme` директиву, чтобы изменить тему вашей диаграммы по умолчаник

```
@startuml
!theme spacelab
class Example {
  Theme spacelab
}
@enduml
```



В начало
Что нового ?
Быстрый старт
Online Server
Запуск
F.A.Q.
Скачать
Форум
Theme
Препроцессинг
Стандартная библиотека
Hitchhiker's Guide
PDF Guide

Примечания по миграции

Текущий препроцессор является обновлением от какого-либо устаревшего препроцес

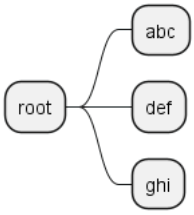
Даже если некоторые устаревшие функции по-прежнему поддерживаются текущим пр не следует их использовать (они могут быть удалены в отдаленном будущем).

- Вам больше не следует использовать `!define` и `!definelong` . Вместо этого испол `!procedure` или определение переменной.
  - `!define` должно быть заменено на return `!function`
  - `!definelong` следует заменить на `!procedure` .
- `!include` теперь допускается множественное включение: вам больше не нужно и
- `!include` теперь принимает URL-адрес, поэтому вам не нужно `!includeurl`
- Некоторые функции (например, `%date%`) были заменены встроенными функциями
- При вызове старого `!definelong` макроса без аргументов вам придется использо должны использовать `my_own_definelong()` потому что `my_own_definelong` новый распознает макрос без круглых скобок.

Пожалуйста, свяжитесь с нами, если у вас возникнут какие-либо вопросы.

`%Splitstr` builtin function

```
@startmindmap
!$list = %splitstr("abc~def~ghi", "~")

* root
!foreach $item in $list
  ** $item
!endfor
@endmindmap
```



[Ссылка на QA-15374]