使用 PlantUML 绘制的 UML

PlantUML 语言参考指引

(Version 1.2023.11)

PlantUML 是一个开源项目，支持快速绘制：

- 时序图
- 用例图
- 类图
- 对象图
- 活动图
- 组件图
- 部署图
- 状态图
- 定时图

同时还支持以下非 UML 图:

- JSON Data
- YAML Data
- Network diagram (nwdiag)
- 线框图形界面
- 架构图
- 规范和描述语言 (SDL)
- Ditaa diagram
- 甘特图
- MindMap diagram
- Work Breakdown Structure diagram
- 以 AsciiMath 或 JLaTeXMath 符号的数学公式
- Entity Relationship diagram

通过简单直观的语言来定义这些示意图。

1 序列图

使用 PlantUML 创建序列图非常简单。这种易用性主要归功于其语法的用户友好性，既直观又易记。

- 直观的语法：

首先，用户非常欣赏 PlantUML 所采用的简单直观的语法。这种经过深思熟虑的设计意味着，即使是图表创建新手也能轻松快速地掌握基础知识。

- 文本与图形的关联：

另一个显著特点是文本表示与图形输出之间非常相似。这种和谐的相关性可确保文本草稿准确地转化为图形图表，从而提供连贯、可预测的设计体验，在最终输出中不会出现令人不快的意外。

- 高效的制作过程：

文本和图形结果之间的紧密联系不仅简化了制作过程，还大大加快了制作速度。用户可从更简化的流程中获益，减少耗时的修改和调整要求。

- 起草时的可视化：

在起草文本的同时就能设想最终的图形结果，这是许多人认为非常宝贵的一项功能。它自然而然地促进了从初稿到最终呈现的顺利过渡，提高了工作效率，降低了出错的可能性。

- 易于编辑和修改：

重要的是，编辑现有图表的过程非常简便。由于图表是由文本生成的，用户会发现进行调整比使用图形工具修改图像要容易得多，也精确得多。

PlantUML 为创建和编辑序列图提供了一种简单明了、用户友好的方法，既能满足新手的需求，也能满足经验丰富的设计人员的需求。它巧妙地利用文本输入的简便性来制作具有视觉描述性和准确性的图表，从而使自己成为图表创建工具包中的必备工具。

您可以了解更多有关 PlantUML 中一些常用命令的信息，以增强您的图表创建体验。

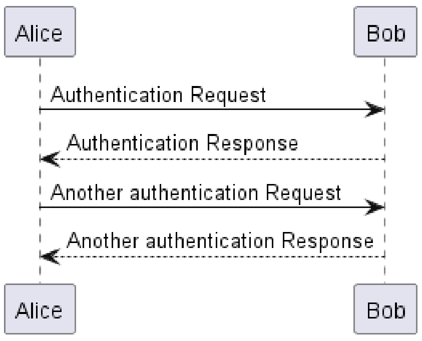
1.1 基本的例子

序列-> 用于绘制两个参与者之间的信息。参与者不必明确声明。

要有一个点状的箭头，就用-->

也可以用 <- 和 <-- 。这不会改变绘图，但可能提高可读性。注意，这只适用于顺序图，其他图的规则不同。

```
@startuml
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

Alice -> Bob: Another authentication Request
Alice <-- Bob: Another authentication Response
@enduml
```


Alice
Bob
Authentication Request
Authentication Response
Another authentication Request
Another authentication Response
Alice
Bob

1.2 声明参与者

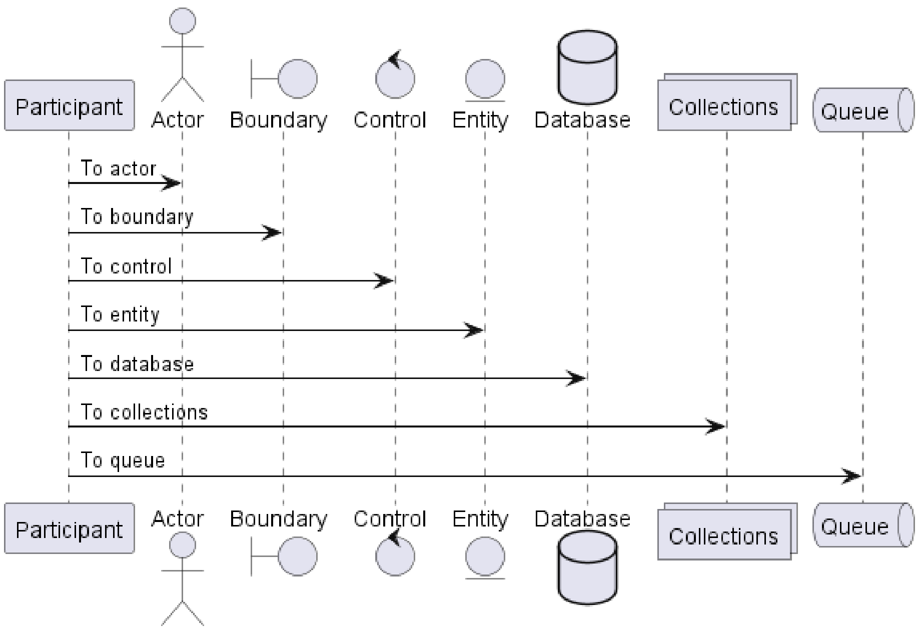
如果使用关键字 participant 来声明参与者，就可以对该参与者进行更多的控制。

声明的顺序将是（默认的）显示顺序。

使用这些其他的关键字来声明参与者，将改变参与者的表示形状。

- actor（角色）
- boundary（边界）
- control（控制）
- entity（实体）
- database（数据库）
- collections（集合）
- queue（队列）

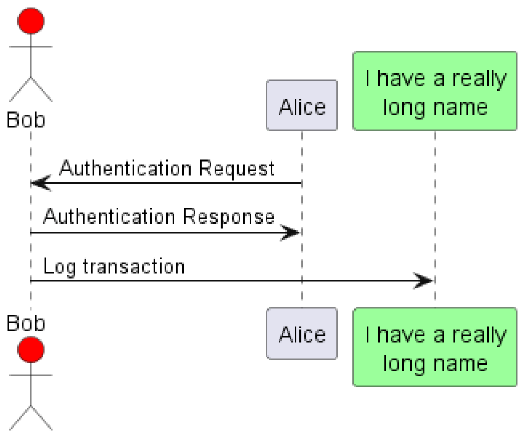
```
@startuml
participant Participant as Foo
actor       Actor       as Foo1
boundary    Boundary    as Foo2
control     Control     as Foo3
entity      Entity      as Foo4
database    Database    as Foo5
collections Collections as Foo6
queue       Queue       as Foo7
Foo -> Foo1 : To actor
Foo -> Foo2 : To boundary
Foo -> Foo3 : To control
Foo -> Foo4 : To entity
Foo -> Foo5 : To database
Foo -> Foo6 : To collections
Foo -> Foo7: To queue
@enduml
```



使用 as 关键字重命名参与者。

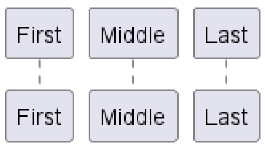
你也可以改变演员或参与者的背景颜色。

```
@startuml
actor Bob #red
' The only difference between actor
'and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
   participant L as "I have a really\nlong name"  #99FF99
  '/

Alice->Bob: Authentication Request
Bob->Alice: Authentication Response
Bob->L: Log transaction
@enduml
```



您可以使用 `order` 关键字来定制参与者的显示顺序。

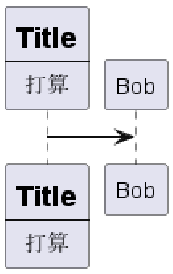
```
@startuml
participant Last order 30
participant Middle order 20
participant First order 10
@enduml
```



1.3 多行定义参与者

你可以对参与者使用多行定义。

```
@startuml
participant Participant [
    =Title
    ----
    ""打算""
]

participant Bob

Participant -> Bob
@enduml
```

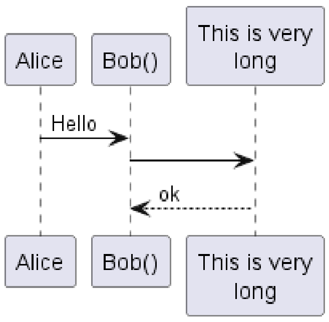


[Ref. QA-15232]

1.4 在参与者中使用非字母

你可以使用引号来定义参与者。而且你可以使用 `as` 关键字来给这些参与者一个别名。

```
@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "This is very\nlong" as Long
' You can also declare:
' "Bob()" -> Long as "This is very\nlong"
Long --> "Bob()" : ok
@enduml
```

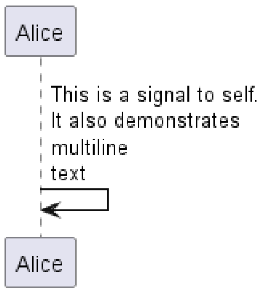


1.5 给自己发消息

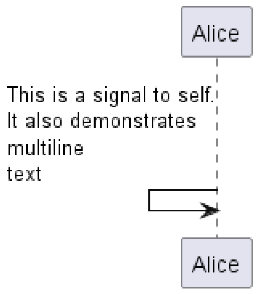
参与者可以给自己发信息，

消息文字可以用来换行。

```
@startuml
Alice -> Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```



```
@startuml
Alice <- Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```



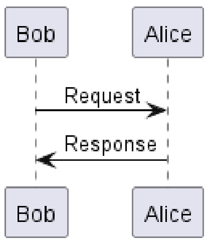
[Ref. [QA-1361](https://forum.plantuml.net/1361)]

1.6 文本对齐

箭头上的文本对齐可以用 `skinparam sequenceMessageAlign`，后接参数 `left`,`right` 或 `center`。

你也可以使用 `direction` 或 `reverseDirection` 来根据箭头的方向对齐文本。更多细节可参考 skinparam。

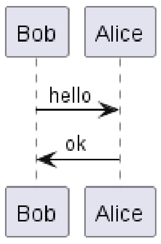
```
@startuml
skinparam sequenceMessageAlign right
Bob -> Alice : Request
Alice -> Bob : Response
@enduml
```



1.6.1 让响应信息显示在箭头下面

你可以使用 `skinparam responseMessageBelowArrow true` 命令，让响应信息显示在箭头下面。

```
@startuml
skinparam responseMessageBelowArrow true
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



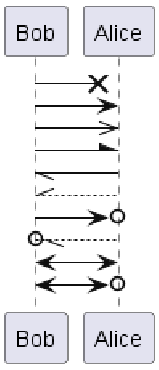
1.7 改变箭头样式

您可以通过以下几种方式改变箭头样式：

- 添加最后的 `x` 表示丢失的信息
- 使用`\` 或`/` 而不是 `<` 或 `>` 只拥有箭头的底部或顶部部分
- 重复箭头头（例如 `>>` 或`//` ）头，拥有一个薄的图纸
- 使用`--` 而不是`-` 拥有一个点状箭头

- 在箭头头添加最后的"o"
- 使用双向的箭头 `<->`

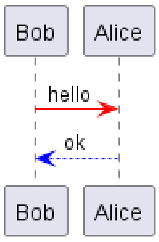
```
@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\- Alice
Bob //-- Alice

Bob ->o Alice
Bob o\\-- Alice

Bob <-> Alice
Bob <->o Alice
@enduml
```



1.8 修改箭头颜色

你可以用以下记号修改箭头的颜色：

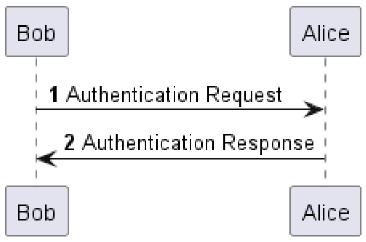
```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```



1.9 对消息序列编号

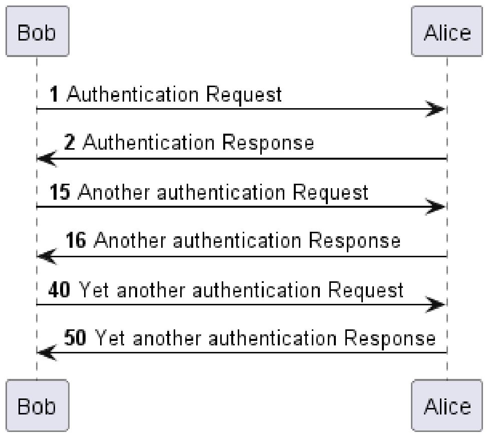
关键字 `autonumber` 用于自动对消息编号。

```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
@enduml
```



语句 `autonumber //start//` 用于指定编号的初始值，而 `autonumber //start// //increment//` 可以同时指定编号的初始值和每次增加的值。

```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

@enduml
```

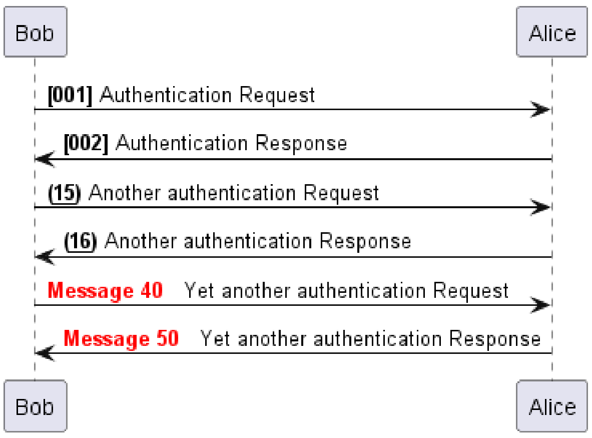

Bob
Alice
1 Authentication Request
2 Authentication Response
15 Another authentication Request
16 Another authentication Response
40 Yet another authentication Request
50 Yet another authentication Response
Bob
Alice

你可以在双引号内指定编号的格式。

格式是由 Java 的 `DecimalFormat` 类实现的：(`0` 表示数字；`#` 也表示数字，但默认为 0)。

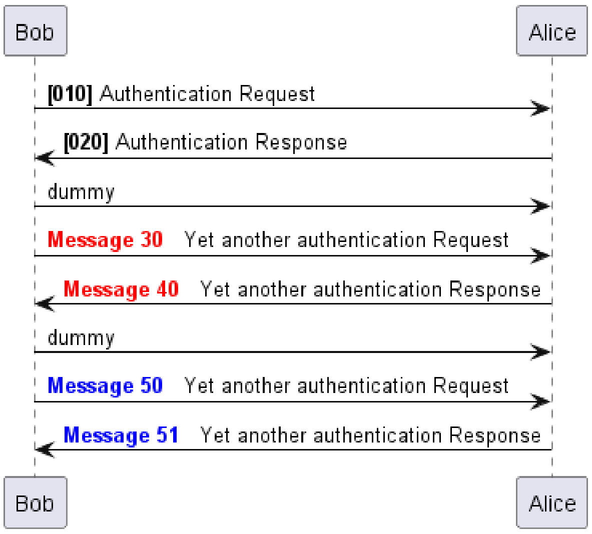
你也可以用 HTML 标签来制定格式。

```
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)"
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

@enduml
```


Bob
Alice
[001] Authentication Request
[002] Authentication Response
(15) Another authentication Request
(16) Another authentication Response
Message 40 Yet another authentication Request
Message 50 Yet another authentication Response
Bob
Alice

你还可以用语句 autonumber stop 和 autonumber resume //increment// //format// 来表示暂停或继续使用自动编号。

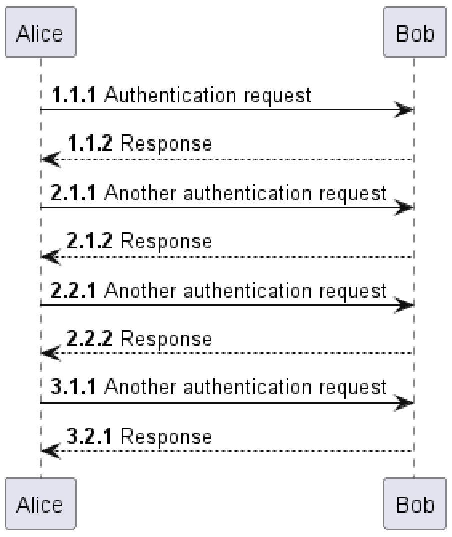
```
@startuml
autonumber 10 10 "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume 1 "<font color=blue><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml
```


Bob
Alice
[010] Authentication Request
[020] Authentication Response
dummy
Message 30 Yet another authentication Request
Message 40 Yet another authentication Response
dummy
Message 50 Yet another authentication Request
Message 51 Yet another authentication Response
Bob
Alice

你也可以使用一个 2 或 3 位的序列，中间采用一种或几种分隔符，如.,;,,,:。例如：1.1.1 或 1.1:1 。最后一位数字会自动递增。

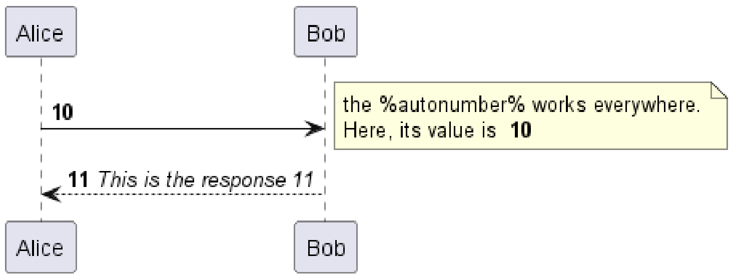
要增加第一个数字，请使用：autonumber inc A 。要增加第二位数字，请使用：autonumber inc B 。

```
@startuml
autonumber 1.1.1
Alice -> Bob: Authentication request
Bob --> Alice: Response

autonumber inc A
'Now we have 2.1.1
Alice -> Bob: Another authentication request
Bob --> Alice: Response

autonumber inc B
'Now we have 2.2.1
Alice -> Bob: Another authentication request
Bob --> Alice: Response

autonumber inc A
'Now we have 3.1.1
Alice -> Bob: Another authentication request
autonumber inc B
'Now we have 3.2.1
Bob --> Alice: Response
@enduml
```


Alice
Bob
1.1.1 Authentication request
1.1.2 Response
2.1.1 Another authentication request
2.1.2 Response
2.2.1 Another authentication request
2.2.2 Response
3.1.1 Another authentication request
3.2.1 Response
Alice
Bob

你也可以用 autonumber 的值，带有%autonumber% 变量。

```
@startuml
autonumber 10
Alice -> Bob
note right
  the <U+0025>autonumber<U+0025> works everywhere.
  Here, its value is ** %autonumber% **
end note
Bob --> Alice: //This is the response %autonumber%//
@enduml
```


Alice
Bob
10
the %autonumber% works everywhere.
Here, its value is 10
11 This is the response 11
Alice
Bob

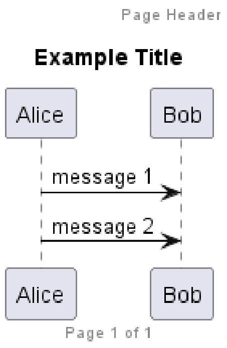
[Ref. QA-7119]

1.10 页面标题、页眉和页脚

`title` 关键字用于为页面添加标题。

页面可以使用 `header` 和 `footer` 显示页眉和页脚。

```
@startuml

header Page Header
footer Page %page% of %lastpage%

title Example Title

Alice -> Bob : message 1
Alice -> Bob : message 2

@enduml
```



1.11 分割示意图

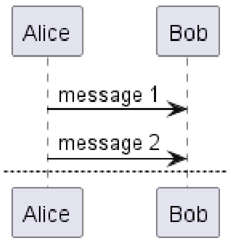
关键字 `newpage` 用于把一张图分割成多张。

在 `newpage` 之后添加文字，作为新的示意图的标题。

这样就能很方便地在 *Word* 中将长图分几页打印。

```
@startuml

Alice -> Bob : message 1
Alice -> Bob : message 2

newpage

Alice -> Bob : message 3
Alice -> Bob : message 4

newpage A title for the\nlast page
```

```
Alice -> Bob : message 5
Alice -> Bob : message 6
@enduml
```



1.12 组合消息

我们可以通过以下关键词来组合消息：

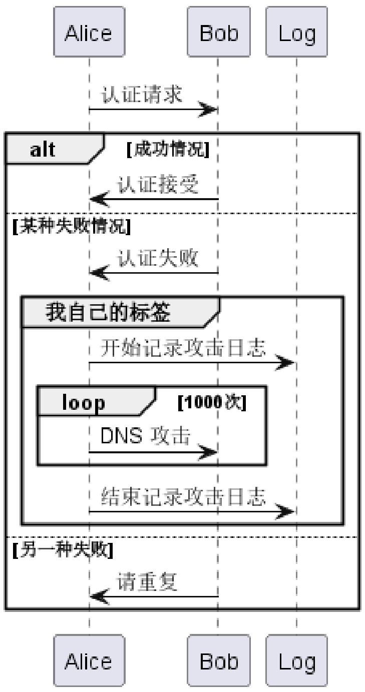
- alt/else
- opt
- loop
- par
- break
- critical
- group, 后面紧跟着消息内容

可以在标头 (header) 添加需要显示的文字 (对于 group 关键字，参看下一章节 ’次级分组标签’)。

关键词 end 用来结束分组。

注意，分组可以嵌套使用。

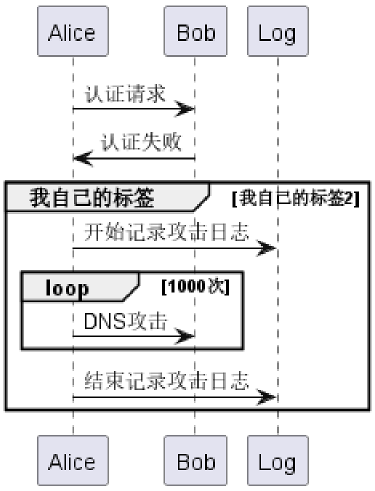
```
@startuml
Alice -> Bob: 认证请求

alt 成功情况

    Bob -> Alice: 认证接受

else 某种失败情况

    Bob -> Alice: 认证失败
    group 我自己的标签
    Alice -> Log : 开始记录攻击日志
        loop 1000次
            Alice -> Bob: DNS 攻击
        end
    Alice -> Log : 结束记录攻击日志
    end

else 另一种失败

   Bob -> Alice: 请重复

end
@enduml
```



1.13 次级分组标签

对于 group 而言，在标头处的 [和] 之间可以显示次级文本或标签。

```
@startuml
Alice -> Bob: 认证请求
Bob -> Alice: 认证失败
group 我自己的标签 [我自己的标签2]
    Alice -> Log : 开始记录攻击日志
    loop 1000次
        Alice -> Bob: DNS攻击
    end
    Alice -> Log : 结束记录攻击日志
end
@enduml
```



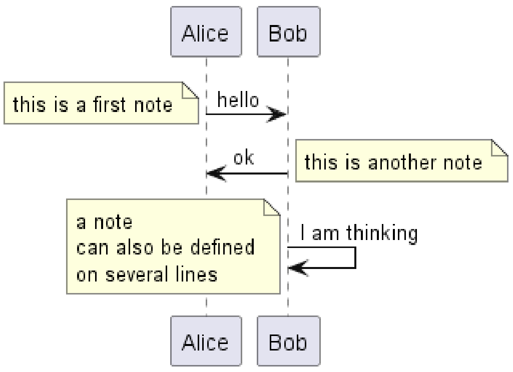
[参看 QA-2503]

1.14 注释信息

可以使用 note left 或 note right 关键字在信息后面加上注释。

你可以使用 `end note` 关键字有一个多行注释。

```
@startuml
Alice->Bob : hello
note left: this is a first note

Bob->Alice : ok
note right: this is another note

Bob->Bob : I am thinking
note left
a note
can also be defined
on several lines
end note
@enduml
```



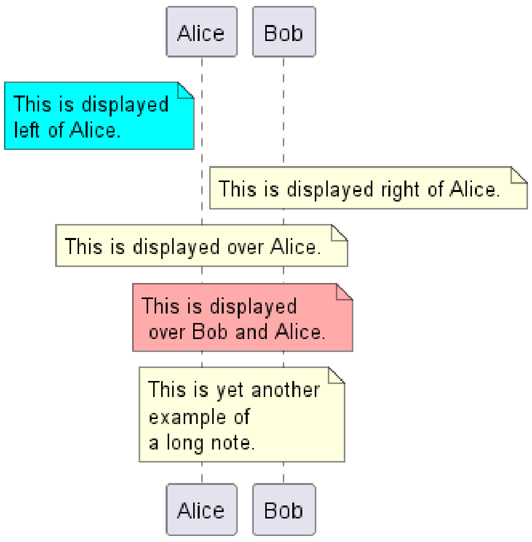
1.15 其他的注释信息方式

可以使用 `note left of`，`note right of` 或 `note over` 在节点 (participant) 的相对位置放置注释。

还可以通过修改背景色来高亮显示注释。

以及使用关键字 `end note` 来添加多行注释。

```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

note right of Alice: This is displayed right of Alice.

note over Alice: This is displayed over Alice.

note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

note over Bob, Alice
This is yet another
example of
a long note.
end note
@enduml
```

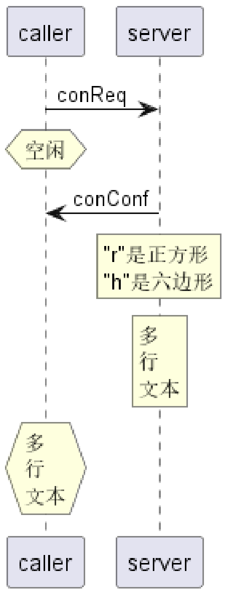

Alice
Bob
This is displayed left of Alice.
This is displayed right of Alice.
This is displayed over Alice.
This is displayed over Bob and Alice.
This is yet another example of a long note.
Alice
Bob

1.16 改变备注框的形状 [hnote 和 rnote]

你可以使用 `hnote` 和 `rnote` 这两个关键字来修改备注框的形状：

- `hnote` 代表六边形（hexagonal）的备注框；
- `rnote` 代表正方形（rectangle）的备注框。

```
@startuml
caller -> server : conReq
hnote over caller : 空闲
caller <- server : conConf
rnote over server
 "r"是正方形
 "h"是六边形
endrnote
rnote over server
 多
 行
 文本
endrnote
hnote over caller
 多
 行
 文本
endhnote
@enduml
```



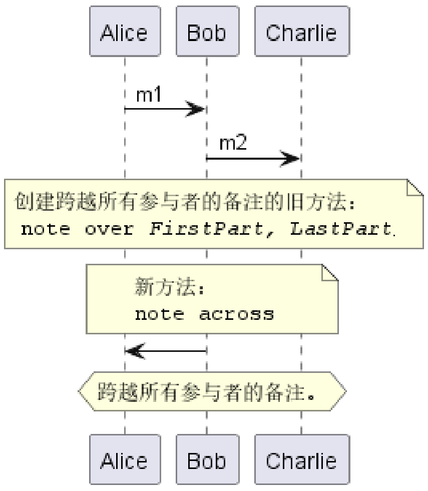
[参见 QA-1765]

1.17 在多个参与者添加备注 [across]

你可以之直接在所有参与者之间添加备注，格式是：

- `note across: 备注描述`

```
@startuml
Alice->Bob:m1
Bob->Charlie:m2
note over Alice, Charlie: 创建跨越所有参与者的备注的旧方法：\n ""note over //FirstPart, LastPart//""
note across: 新方法：\n""note across""
Bob->Alice
hnote across: 跨越所有参与者的备注。
@enduml
```



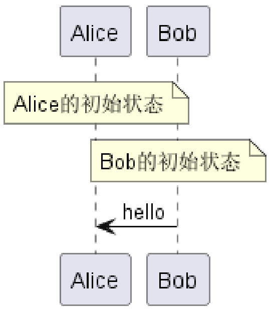
[参见 QA-9738]

1.18 在同一级对齐多个备注 [/]

使用/可以在同一级对齐多个备注：

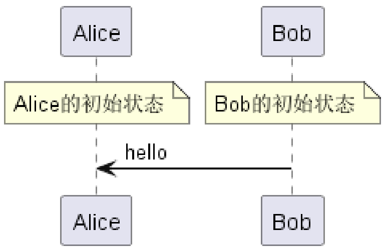
- 没有/ （默认情况下，备注不是对齐的。）

```
@startuml
note over Alice : Alice的初始状态
note over Bob : Bob的初始状态
Bob -> Alice : hello
@enduml
```



- with / *(the notes are aligned)*

```
@startuml
note over Alice : Alice的初始状态
/ note over Bob : Bob的初始状态
Bob -> Alice : hello
@enduml
```



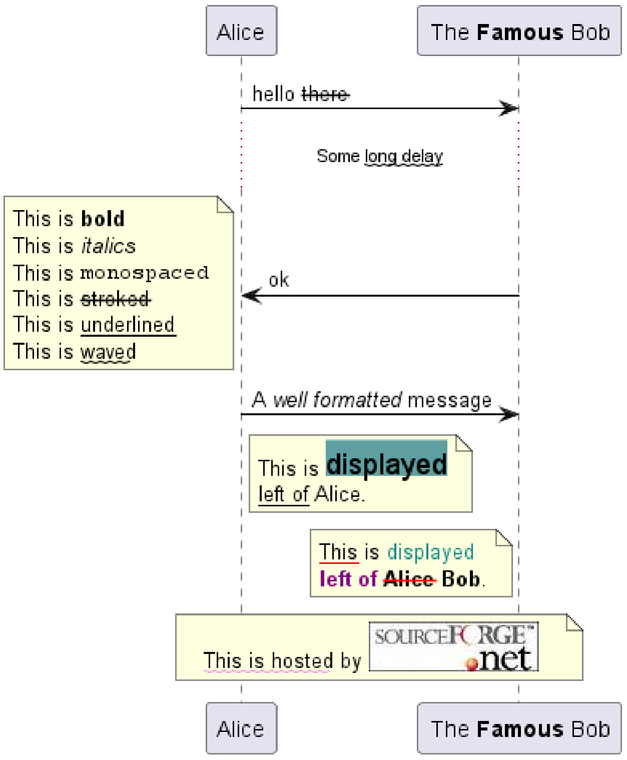
[参见 QA-354]

1.19 Creole 和 HTML

可以使用 creole 格式。

```
@startuml
participant Alice
participant "The **Famous** Bob" as Bob

Alice -> Bob : hello --there--
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stroked--
  This is __underlined__
  This is ~~waved~~
end note

Alice -> Bob : A //well formatted// message
note right of Alice
 This is <back:cadetblue><size:18>displayed</size></back>
 __left of__ Alice.
end note
```

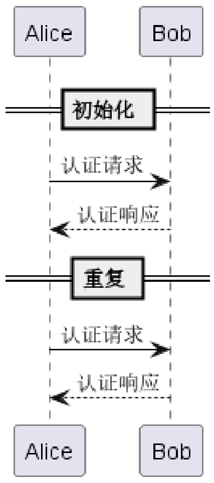
```
note left of Bob
 <u:red>This</u> is <color #118888>displayed</color>
 **<color purple>left of</color> <s:red>Alice</strike> Bob**.
end note
note over Alice, Bob
 <w:#FF33FF>This is hosted</w> by <img sourceforge.jpg>
end note
@enduml
```



1.20 分隔符

你可以通过使用 == 关键词来将你的图表分割成多个逻辑步骤。

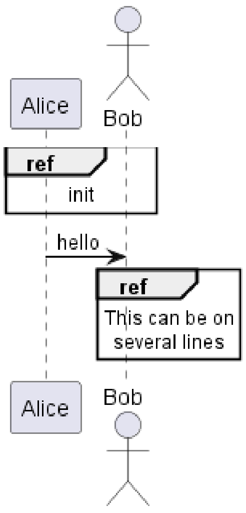
```
@startuml

== 初始化 ==

Alice -> Bob: 认证请求
Bob --> Alice: 认证响应

== 重复 ==

Alice -> Bob: 认证请求
Alice <-- Bob: 认证响应

@enduml
```



1.21 引用

你可以在图中通过使用 `ref over` 关键词来实现引用

```
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

ref over Bob
  This can be on
  several lines
end ref
@enduml
```

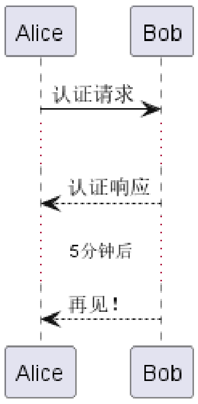


1.22 延迟

你可以使用 `...` 来表示延迟，并且还可以给延迟添加注释。

```
@startuml

Alice -> Bob: 认证请求
...
```

```
Bob --> Alice: 认证响应
...5分钟后...
Bob --> Alice: 再见!

@enduml
```

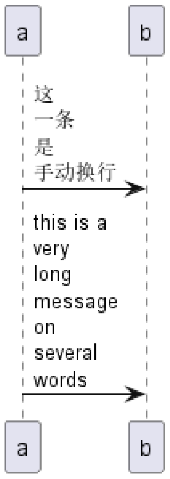


1.23 文本换行

你可以通过手动在文本中添加使长文本换行。

或者使用 maxMessageSize 设置（此方式暂不支持中文换行）：

```
@startuml
skinparam maxMessageSize 50
participant a
participant b
a -> b :这\n一条\n是\n手动换行
a -> b :this is a very long message on several words
@enduml
```



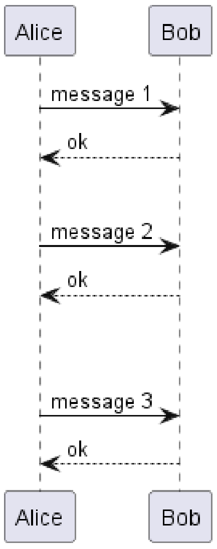
1.24 空间

你可以使用 ||| 来增加空间。

还可以使用数字指定增加的像素的数量。

```
@startuml

Alice -> Bob: message 1
Bob --> Alice: ok
|||
Alice -> Bob: message 2
```

```
Bob --> Alice: ok
||45||
Alice -> Bob: message 3
Bob --> Alice: ok

@enduml
```



1.25 生命线的激活与撤销

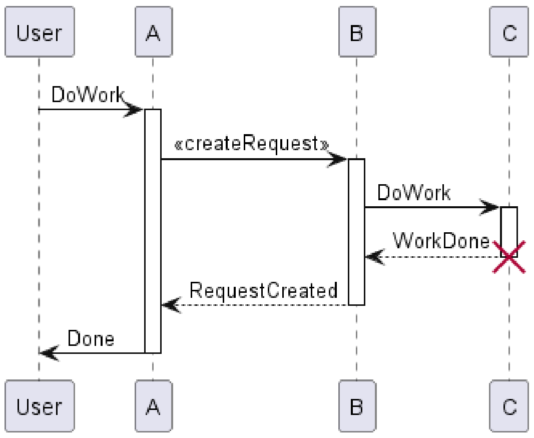
关键字 `activate` 和 `deactivate` 用来表示参与者的生命活动。

一旦参与者被激活，它的生命线就会显示出来。

`activate` 和 `deactivate` 适用于以上情形。

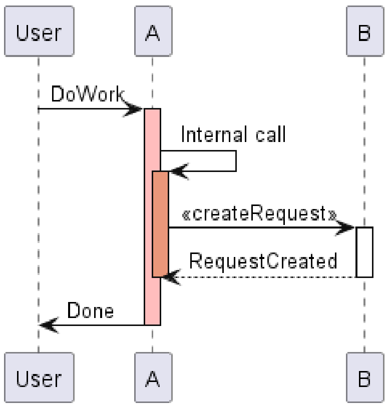
`destroy` 表示一个参与者的生命线的终结。

```
@startuml
participant User

User -> A: DoWork
activate A

A -> B: << createRequest >>
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: RequestCreated
deactivate B

A -> User: Done
deactivate A

@enduml
```



还可以使用嵌套的生命线，并且运行给生命线添加颜色。

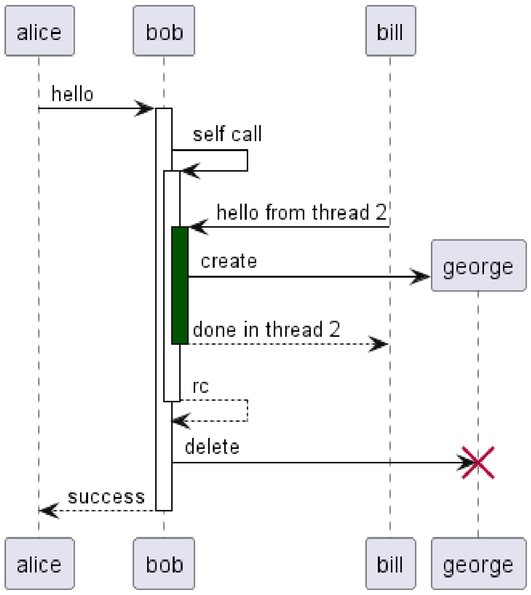
```
@startuml
participant User

User -> A: DoWork
activate A #FFBBBB

A -> A: Internal call
activate A #DarkSalmon

A -> B: << createRequest >>
activate B

B --> A: RequestCreated
deactivate B
deactivate A
A -> User: Done
deactivate A

@enduml
```



也可以使用自动激活关键字（autoactivate），这需要与 `return` 关键字配合：

```
@startuml
autoactivate on
alice -> bob : hello
bob -> bob : self call
bill -> bob #005500 : hello from thread 2
bob -> george ** : create
return done in thread 2
return rc
```

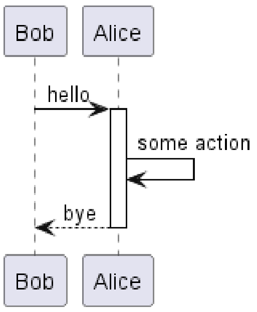
```
bob -> george !! : delete
return success

@enduml
```



1.26 返回

新命令 `return` 可以用于生成一个带有可选文本标签的返回信息。返回的点是导致最近一次激活生命线的点。语法是简单的返回标签，其中标签（如果提供）可以是传统信息中可以接受的任何字符串。

```
@startuml
Bob -> Alice : hello
activate Alice
Alice -> Alice : some action
return bye
@enduml
```

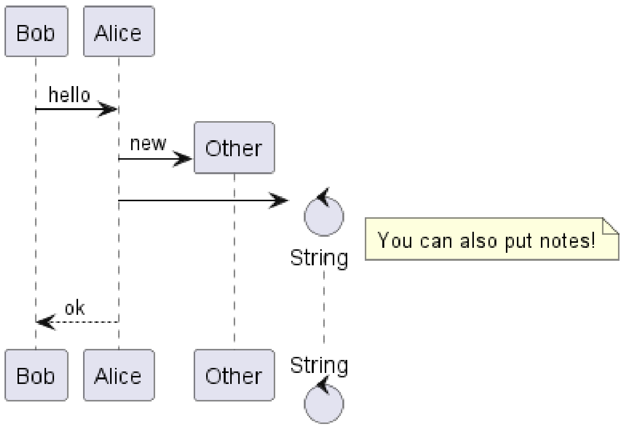


1.27 创建参与者

你可以把关键字 `create` 放在第一次接收到消息之前，以强调本次消息实际上是在创建新的对象。

```
@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

create control String
Alice -> String
note right : You can also put notes!
```

```
Alice --> Bob : ok

@enduml
```

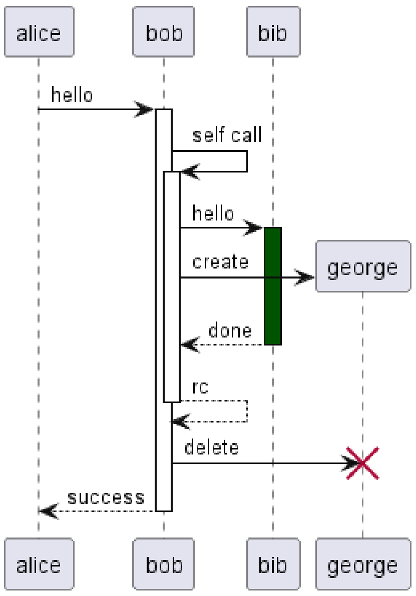


1.28 激活、撤销和创建的快捷语法。

在指定目标参与者后，可以立即使用以下语法：

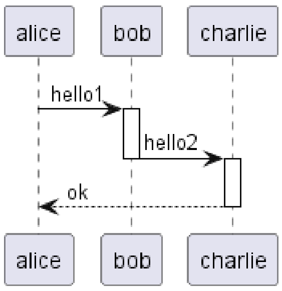
- `++` 激活目标（可选择在后面加上 #color）
- `--` 撤销激活源
- `**` 创建目标实例
- `!!` 摧毁目标实例

```
@startuml
alice -> bob ++ : hello
bob -> bob ++ : self call
bob -> bib ++  #005500 : hello
bob -> george ** : create
return done
return rc
bob -> george !! : delete
return success
@enduml
```

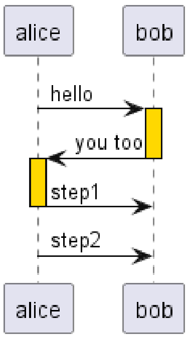


然后你就可以在一行上同时激活和撤销：

```
@startuml
alice   ->  bob     ++   : hello1
bob     ->  charlie --++ : hello2
charlie --> alice   --   : ok
@enduml
```



```
@startuml
@startuml
alice -> bob   --++ #gold: hello
bob   -> alice --++ #gold: you too
alice -> bob   --: step1
alice -> bob   : step2
@enduml
@enduml
```



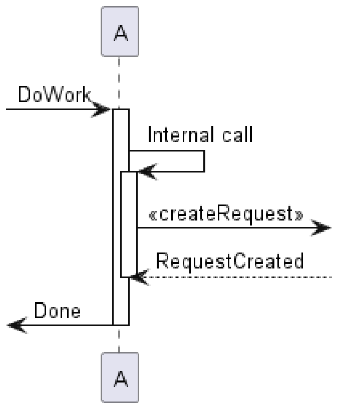
[参见 QA-4834, QA-9573 和 QA-13234]

1.29 进入和发出消息

如果只想关注部分图示，你可以使用进入和发出箭头。

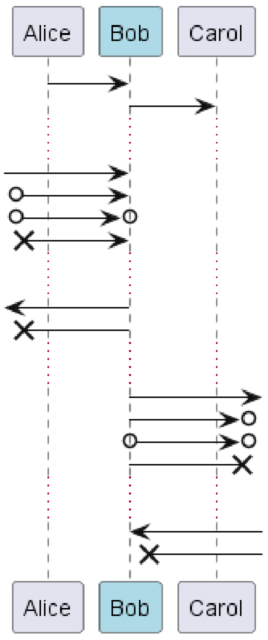
使用方括号 [和] 表示图示的左、右两侧。

```
@startuml
[-> A: DoWork

activate A

A -> A: Internal call
activate A

A ->] : << createRequest >>

A<--] : RequestCreated
deactivate A
[<- A: Done
deactivate A
@enduml
```



还可以使用下面的语法:

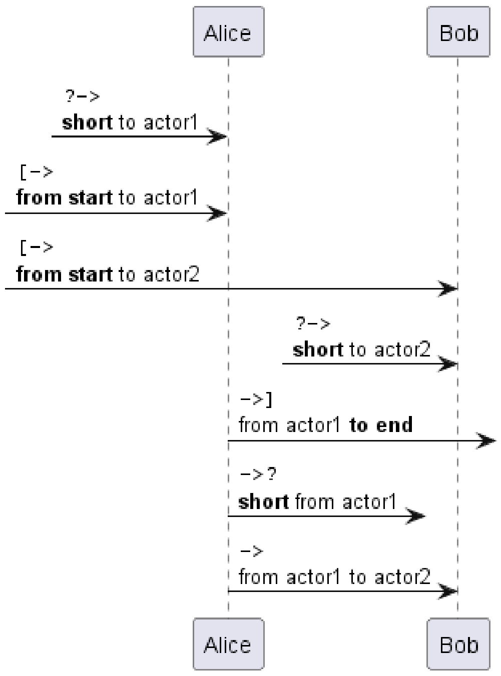
```
@startuml
participant Alice
participant Bob #lightblue
Alice -> Bob
Bob -> Carol
...
[-> Bob
[o-> Bob
[o->o Bob
[x-> Bob
...
[<- Bob
[x<- Bob
...
Bob ->]
Bob ->o]
Bob o->o]
Bob ->x]
...
Bob <-]
Bob x<-]

@enduml
```



1.30 缩短的进入信息与发出信息箭头

使用? 来显示缩短的箭头。

```
@startuml
?-> Alice     : ""?->""\n**short** to actor1
[-> Alice     : ""[->""\n**from start** to actor1
[-> Bob       : ""[->""\n**from start** to actor2
?-> Bob       : ""?->""\n**short** to actor2
Alice ->]     : ""->]""\nfrom actor1 **to end**
Alice ->?     : ""->?""\n**short** from actor1
Alice -> Bob : ""->"" \nfrom actor1 to actor2
@enduml
```

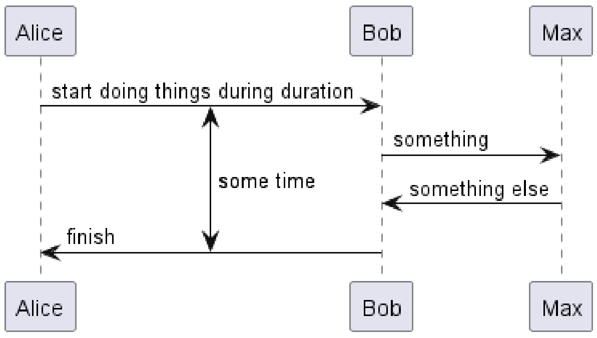


[参见 QA-310]

1.31 锚点和持续时间

使用 teoz 在图表中添加锚点，从而指定持续时间。

```
@startuml
!pragma teoz true

{start} Alice -> Bob : start doing things during duration
Bob -> Max : something
Max -> Bob : something else
{end} Bob -> Alice : finish

{start} <-> {end} : some time

@enduml
```



You can use the -P command-line option to specify the pragma:

```
java -jar plantuml.jar -Pteoz=true
```

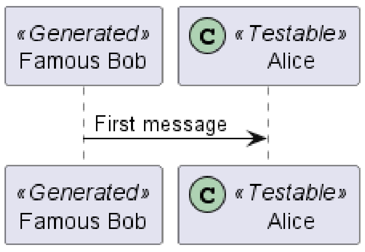
[Ref. issue-582]

1.32 构造类型和圈点

可以使用 << 和 >> 给参与者添加构造类型。

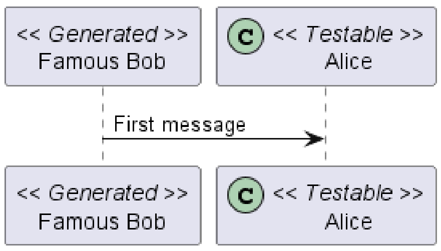
在构造类型中，你可以使用 (X,color) 格式的语法添加一个圆圈圈起来的字符。

```
@startuml

participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>

Bob->Alice: First message

@enduml
```

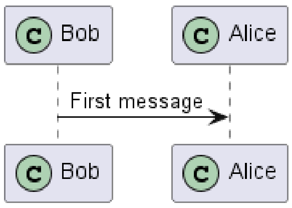


默认使用 *guillemet* 字符来显示构造类型。你可以使用外观参数 guillemet 来修改显示行为。

```
@startuml

skinparam guillemet false
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>

Bob->Alice: First message

@enduml
```



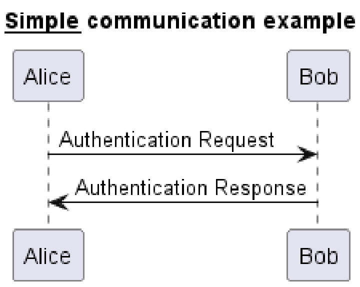
```
@startuml

participant Bob << (C,#ADD1B2) >>
participant Alice << (C,#ADD1B2) >>

Bob->Alice: First message

@enduml
```



1.33 更多标题信息

你可以在标题中使用 creole 格式。

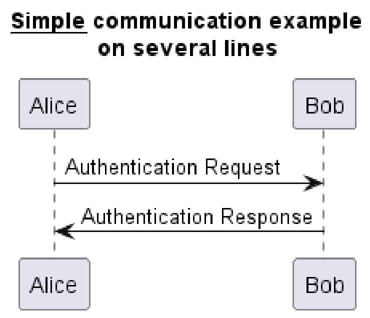
```
@startuml

title __Simple__ **communication** example

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```



在标题描述中使用表示换行。

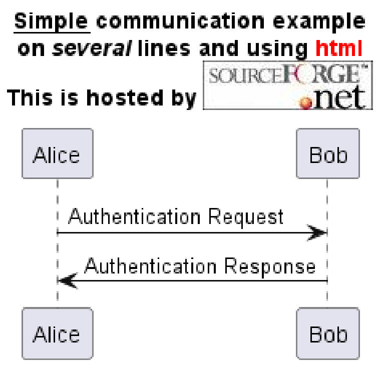
```
@startuml

title __Simple__ communication example\non several lines
```

```
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```



还可以使用关键字 `title` 和 `end title` 定义多行标题。

```
@startuml

title
 <u>Simple</u> communication example
 on <i>several</i> lines and using <font color=red>html</font>
 This is hosted by <img:sourceforge.jpg>
end title

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```



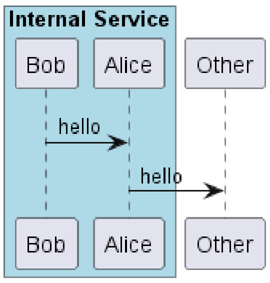
1.34 包裹参与者

可以使用 `box` 和 `end box` 画一个盒子将参与者包裹起来。

还可以在 `box` 关键字之后添加标题或者背景颜色。

```
@startuml

box "Internal Service" #LightBlue
participant Bob
participant Alice
end box
participant Other

Bob -> Alice : hello
Alice -> Other : hello
```

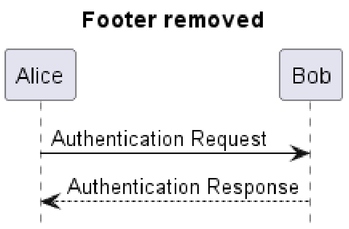
```
@enduml
```



1.35 移除脚注

使用 `hide footbox` 关键字移除脚注。

```
@startuml

hide footbox
title Footer removed

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

@enduml
```



1.36 外观参数 (skinparam)

用 skinparam 改变字体和颜色。

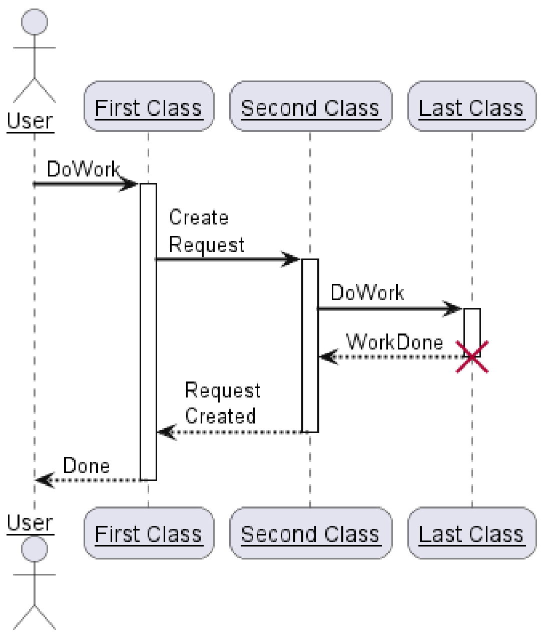
可以在如下场景中使用：

- 在图示的定义中，
- 在引入的文件中，
- 在命令行或者 ANT 任务提供的配置文件中。

你也可以修改其他渲染元素，如以下示例：

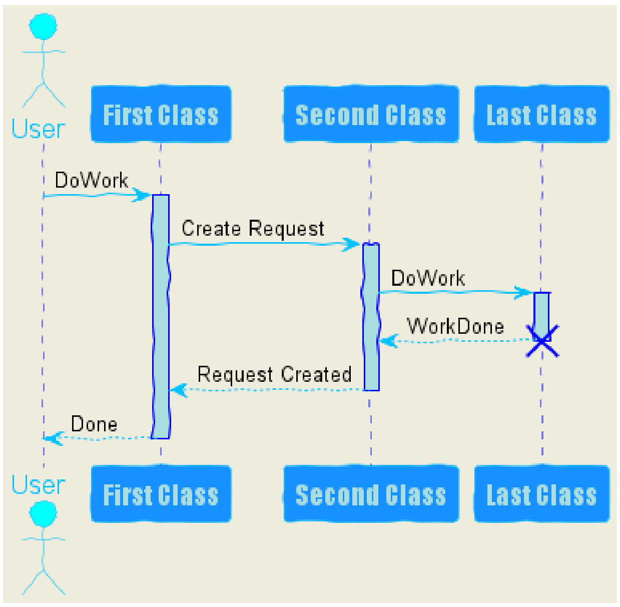
```
@startuml
skinparam sequenceArrowThickness 2
skinparam roundcorner 20
skinparam maxmessagesize 60
skinparam sequenceParticipant underline

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A
```

```
A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```



```
@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

skinparam sequence {
ArrowColor DeepSkyBlue
ActorBorderColor DeepSkyBlue
LifeLineBorderColor blue
LifeLineBackgroundColor #A9DCDF

ParticipantBorderColor DeepSkyBlue
ParticipantBackgroundColor DodgerBlue
ParticipantFontName Impact
ParticipantFontSize 17
ParticipantFontColor #A9DCDF

ActorBackgroundColor aqua
ActorFontColor DeepSkyBlue
ActorFontSize 17
ActorFontName Aapex
}
```

```
actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```

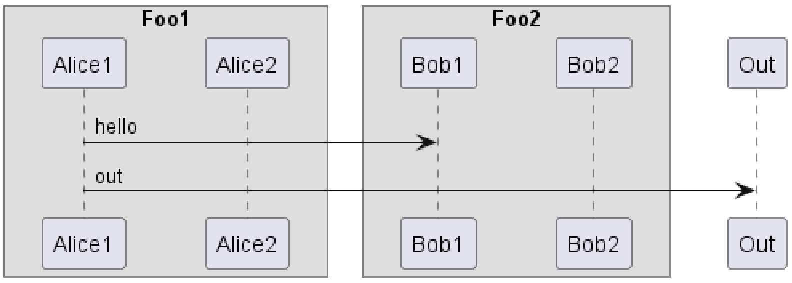


1.37 填充区设置

可以设定填充区的参数配置。

```
@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1"
participant Alice1
participant Alice2
end box
box "Foo2"
participant Bob1
participant Bob2
end box
```

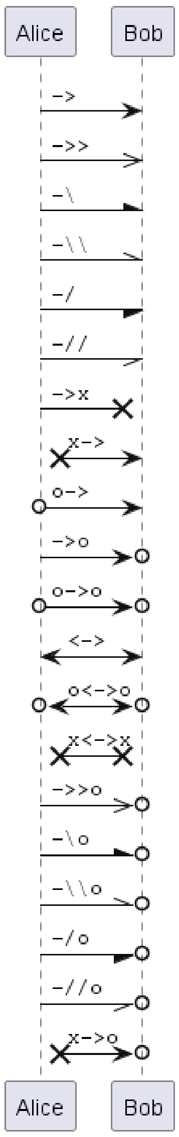
```
Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml
```



1.38 附录：箭头类型大全

1.38.1 普通箭头

```
@startuml
participant Alice as a
participant Bob   as b
a ->      b : ""->   ""
a ->>     b : ""->>  ""
a -\      b : ""-\   ""
a -\\     b : ""-\\\\""
a -/      b : ""-/   ""
a -//     b : ""-//  ""
a ->x     b : ""->x  ""
a x->     b : ""x->  ""
a o->     b : ""o->  ""
a ->o     b : ""->o  ""
a o->o    b : ""o->o ""
a <->     b : ""<->  ""
a o<->o   b : ""o<->o""
a x<->x   b : ""x<->x""
a ->>o    b : ""->>o ""
a -\o     b : ""-\o  ""
a -\\o    b : ""-\\\\o""
a -/o     b : ""-/o  ""
a -//o    b : ""-//o ""
a x->o    b : ""x->o ""
@enduml
```

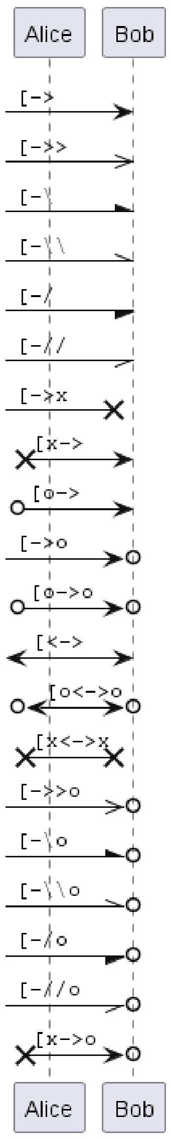


1.38.2 进入信息和发出信息（使用'[', ']'）

1.38.3 进入信息（使用'['）

```
@startuml
participant Alice as a
participant Bob   as b
[->      b : ""[->   ""
[->>     b : ""[->>  ""
[-\      b : ""[-\   ""
[-\\     b : ""[-\\\\""
[-/      b : ""[-/   ""
[-//     b : ""[-//  ""
[->x     b : ""[->x  ""
[x->     b : ""[x->  ""
[o->     b : ""[o->  ""
[->o     b : ""[->o  ""
[o->o    b : ""[o->o ""
[<->     b : ""[<->  ""
[o<->o   b : ""[o<->o""
[x<->x   b : ""[x<->x""
[->>o    b : ""[->>o ""
```

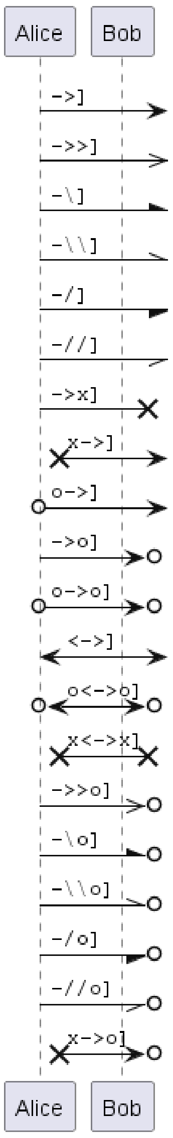
```
[-\o     b : ""[-\o   ""
[-\\o    b : ""[-\\\\o""
[-/o     b : ""[-/o   ""
[-//o    b : ""[-//o  ""
[x->o    b : ""[x->o  ""
@enduml
```



1.38.4 发出信息（使用’]’）

```
@startuml
participant Alice as a
participant Bob   as b
a ->]      : ""->]   ""
a ->>]     : ""->>]  ""
a -\]      : ""-\]   ""
a -\\]     : ""-\\\\]""
a -/]      : ""-/]   ""
a -//]     : ""-//]  ""
a ->x]     : ""->x]  ""
a x->]     : ""x->]  ""
a o->]     : ""o->]  ""
a ->o]     : ""->o]  ""
```

```
a o->o]     : ""o->o] ""
a <->]      : ""<->]  ""
a o<->o]    : ""o<->o]""
a x<->x]    : ""x<->x]""
a ->>o]     : ""->>o] ""
a -\o]      : ""-\o]  ""
a -\\o]     : ""-\\\\o]""
a -/o]      : ""-/o]  ""
a -//o]     : ""-//o] ""
a x->o]     : ""x->o] ""
@enduml
```

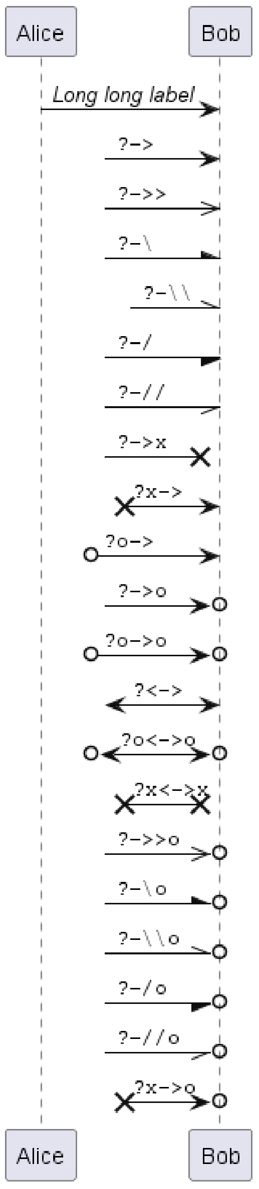


1.38.5 短进入信息和短发出信息（使用'?'）

1.38.6 短进入信息（使用'?'）

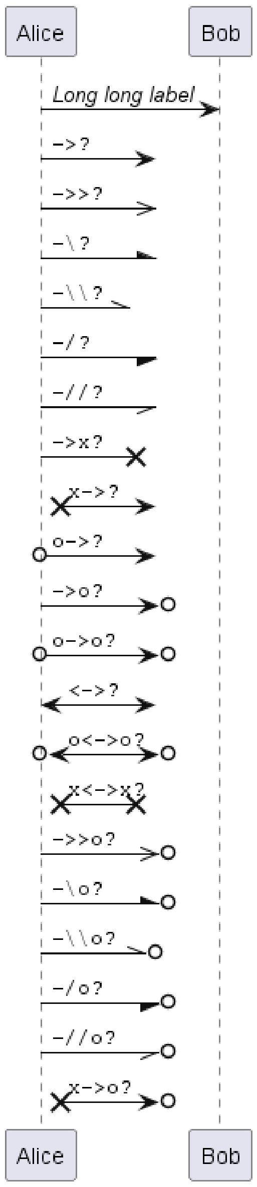
```
@startuml
participant Alice as a
participant Bob   as b
a ->     b : //Long long label//
?->      b : ""?->    ""
?->>     b : ""?->>   ""
?-\      b : ""?-\    ""
```

```
?-\\      b : ""?-\\\\""
?-/       b : ""?-/   ""
?-//      b : ""?-//  ""
?->x      b : ""?->x  ""
?x->      b : ""?x->  ""
?o->      b : ""?o->  ""
?->o      b : ""?->o  ""
?o->o     b : ""?o->o ""
?<->      b : ""?<->  ""
?o<->o    b : ""?o<->o""
?x<->x    b : ""?x<->x""
?->>o     b : ""?->>o ""
?-\o      b : ""?-\o  ""
?-\\o     b : ""?-\\\\o ""
?-/o      b : ""?-/o  ""
?-//o     b : ""?-//o ""
?x->o     b : ""?x->o ""
@enduml
```



1.38.7 短发出信息（使用'?'）

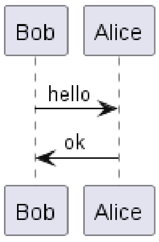
```
@startuml
participant Alice as a
participant Bob   as b
a ->     b : //Long long label//
a ->?      : ""->?   ""
a ->>?     : ""->>?  ""
a -\?      : ""-\?   ""
a -\\?     : ""-\\\\?""
a -/?      : ""-/?   ""
a -//?     : ""-//?  ""
a ->x?     : ""->x?  ""
a x->?     : ""x->?  ""
a o->?     : ""o->?  ""
a ->o?     : ""->o?  ""
a o->o?    : ""o->o? ""
a <->?     : ""<->?  ""
a o<->o?   : ""o<->o?""
a x<->x?   : ""x<->x?""
a ->>o?    : ""->>o? ""
a -\o?     : ""-\o?  ""
a -\\o?    : ""-\\\\o?""
a -/o?     : ""-/o?  ""
a -//o?    : ""-//o? ""
a x->o?    : ""x->o? ""
@enduml
```



1.39 特定外观参数

1.39.1 默认情况下

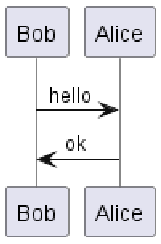
```
@startuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



1.39.2 生命线策略

- nosolid 虚线 （默认情况）

```
@startuml
skinparam lifelineStrategy nosolid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```

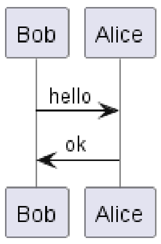


[参见 QA-9016]

- solid 实线

在时序图中使用实线生命线：`skinparam lifelineStrategy solid`

```
@startuml
skinparam lifelineStrategy solid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



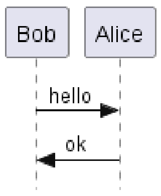
[参见 QA-2794]

1.39.3 style strictuml

为了符合严格 UML 的标准（线头的形状必须是三角形，而不能是箭头形），你可以使用：

- `skinparam style strictuml`

```
@startuml
skinparam style strictuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```

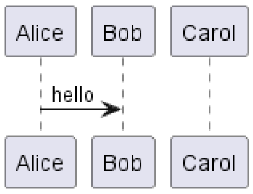


[参见 QA-1047]

1.40 隐藏孤立参与者

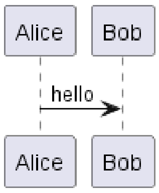
默认情况下会显示所有参与者。

```
@startuml
participant Alice
participant Bob
participant Carol

Alice -> Bob : hello
@enduml
```



可以使用 `hide unlinked` 命令来隐藏未被链接到的参与者。

```
@startuml
hide unlinked
participant Alice
participant Bob
participant Carol

Alice -> Bob : hello
@enduml
```

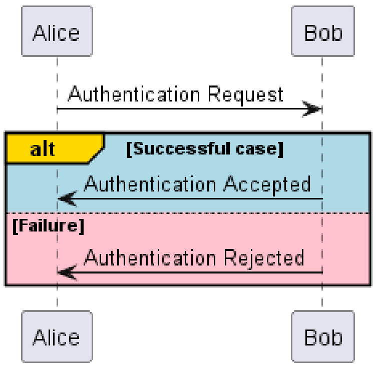


[Ref. QA-4247]

1.41 给分组信息着色

可以给分组信息 color[着色]。

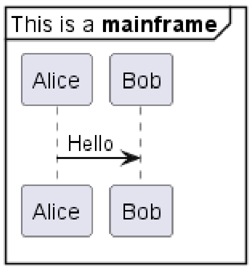
```
@startuml
Alice -> Bob: Authentication Request
alt#Gold #LightBlue Successful case
    Bob -> Alice: Authentication Accepted
else #Pink Failure
    Bob -> Alice: Authentication Rejected
end
@enduml
```



[Ref.QA-4750andQA-6410]

1.42 Mainframe

```
@startuml
mainframe This is a **mainframe**
Alice->Bob : Hello
@enduml
```



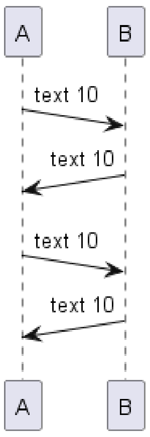
[Ref. QA-4019 and Issue#148]

1.43 Slanted or odd arrows

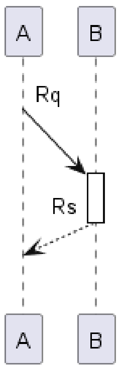
You can use the `(nn)` option (before or after arrow) to make the arrows slanted, where *nn* is the number of shift pixels.

[Available only after v1.2022.6beta+]

```
@startuml
A ->(10) B: text 10
B ->(10) A: text 10

A ->(10) B: text 10
A (10)<- B: text 10
@enduml
```

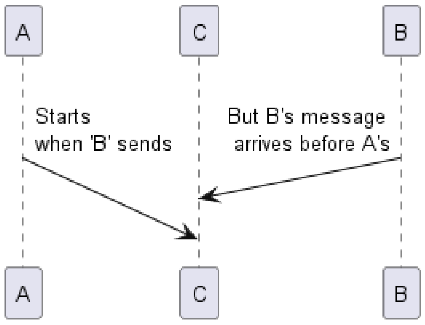


```
@startuml
A ->(40) B++: Rq
B -->(20) A--: Rs
@enduml
```



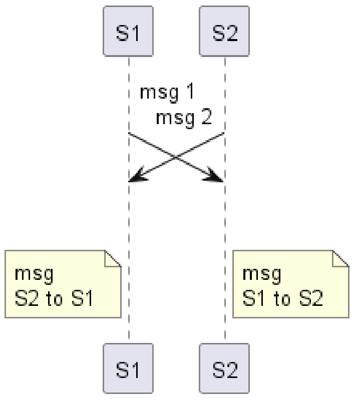
[Ref. QA-14145]

```
@startuml
!pragma teoz true
A ->(50) C: Starts\nwhen 'B' sends
& B ->(25) C: \nBut B's message\n arrives before A's
@enduml
```



[Ref. QA-6684]

```
@startuml
!pragma teoz true

S1 ->(30) S2: msg 1\n
& S2 ->(30) S1: msg 2

note left S1: msg\nS2 to S1
& note right S2: msg\nS1 to S2
@enduml
```



[Ref. QA-1072]

2 用例图

举几个例子：用例

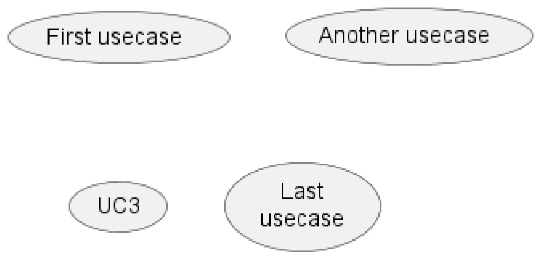
PlantUML offers a unique approach to creating use case diagrams through its text-based language. One of the primary advantages of using PlantUML is its **simplicity and efficiency**. Instead of manually drawing shapes and connections, users can define their diagrams using intuitive and concise textual descriptions. This not only speeds up the diagram creation process but also ensures **consistency and accuracy**. The ability to integrate with various documentation platforms and its wide range of supported output formats make PlantUML a versatile tool for both developers and non-developers. Lastly, being **open-source**, PlantUML boasts a strong community that continually contributes to its improvement and offers a wealth of resources for users at all levels.

2.1 用例

用例用圆括号括起来（两个圆括号看起来就像椭圆）。

也可以用关键字 `usecase` 来定义用例。还可以用关键字 `as` 定义一个别名，这个别名可以在以后定义关系的时候使用。

```
@startuml

(First usecase)
(Another usecase) as (UC2)
usecase UC3
usecase (Last\nusecase) as UC4

@enduml
```



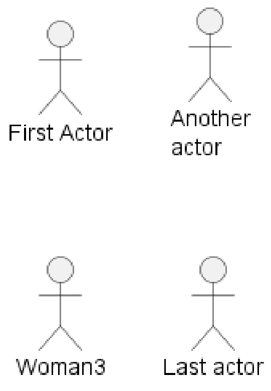
2.2 角色

角色用两个冒号包裹起来。

也可以用 `actor` 关键字来定义角色。还可以用关键字 `as` 来定义一个别名，这个别名可以在以后定义关系的时候使用。

在后面的例子中，我们会看到角色的定义是可选的。

```
@startuml

:First Actor:
:Another\nactor: as Man2
actor Woman3
actor :Last actor: as Person1

@enduml
```



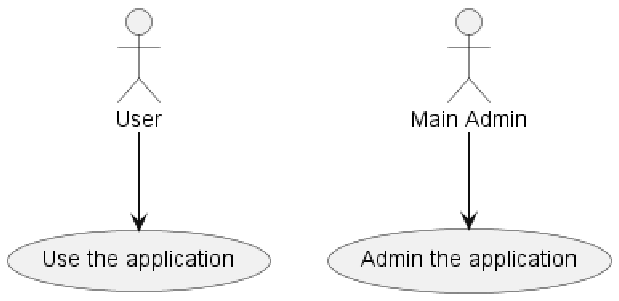
2.3 改变角色的样式

可以将角色的样式从默认的火柴人改成：

- 用户头像样式：`skinparam actorStyle awesome`
- 透明人样式：`skinparam actorStyle hollow`

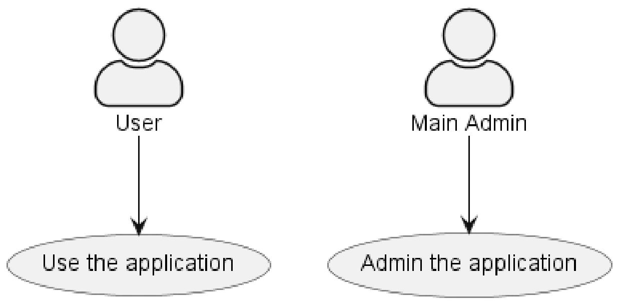
2.3.1 火柴人 默认

```
@startuml
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



2.3.2 用户头像

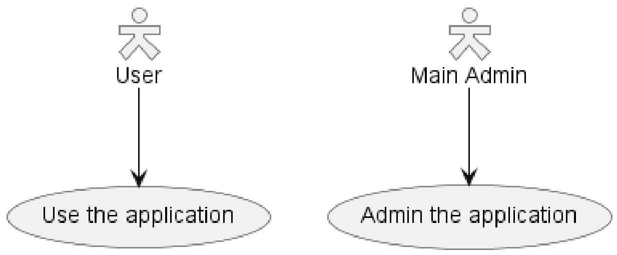
```
@startuml
skinparam actorStyle awesome
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



[参见 QA-10493]

2.3.3 透明人

```
@startuml
skinparam actorStyle Hollow
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



[参见 PR#396]

2.4 用例描述

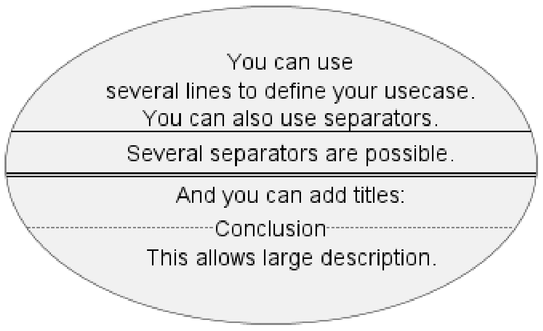
如果想定义跨越多行的用例描述，可以用双引号将其裹起来。

还可以使用这些分隔符：

- `--`（横线）
- `..`（虚线）
- `==`（双横线）
- `__`（下划线）

并且还可以在分隔符中间放置标题。

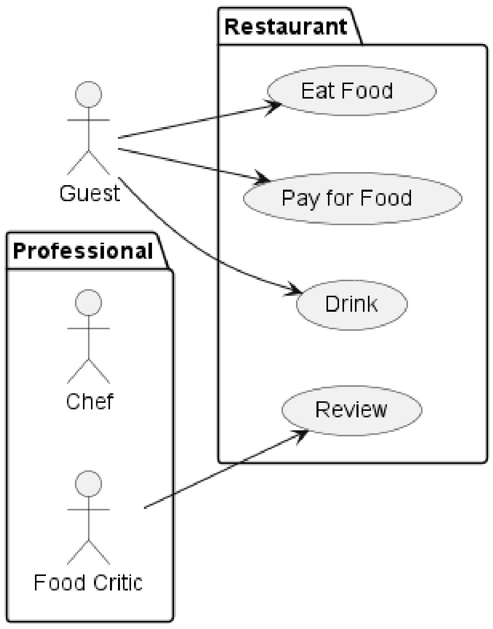
```
@startuml

usecase UC1 as "You can use
several lines to define your usecase.
You can also use separators.
--
Several separators are possible.
==
And you can add titles:
..Conclusion..
This allows large description."

@enduml
```


You can use
several lines to define your usecase.
You can also use separators.
Several separators are possible.
And you can add titles:
Conclusion
This allows large description.

2.5 使用包

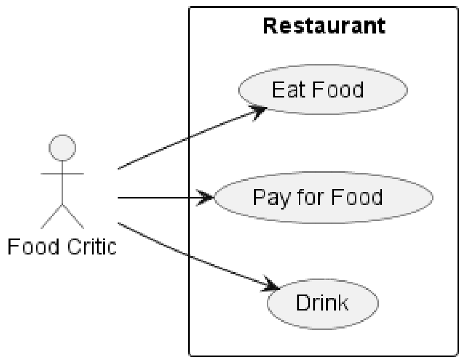
您可以一使用包来对角色或用例进行分组。

```
@startuml
left to right direction
actor Guest as g
package Professional {
  actor Chef as c
  actor "Food Critic" as fc
}
package Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
  usecase "Review" as UC4
}
fc --> UC4
g --> UC1
g --> UC2
g --> UC3
@enduml
```



您可以使用 `rectangle` 来改变包的外观。

```
@startuml
left to right direction
actor "Food Critic" as fc
rectangle Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
@enduml
```



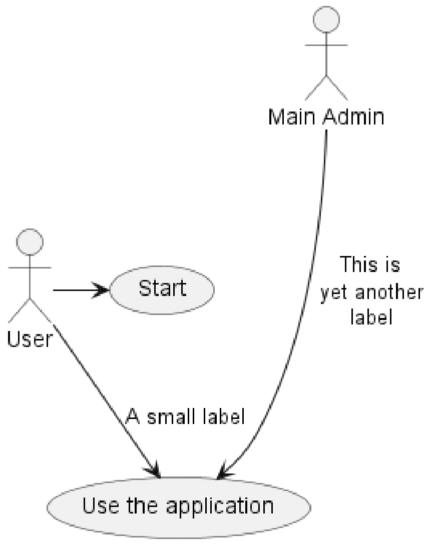
2.6 基础示例

用箭头--> 连接角色和用例。

横杠-越多，箭头越长。通过在箭头定义的后面加一个冒号及文字的方式来添加标签。

在这个例子中，*User* 并没有定义，而是直接拿来当做一个角色使用。

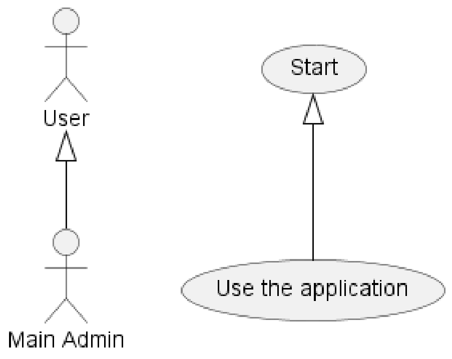
```
@startuml

User -> (Start)
User --> (Use the application) : A small label

:Main Admin: ---> (Use the application) : This is\nyet another\nlabel

@enduml
```



2.7 继承

如果一个角色或者用例继承于另一个，那么可以用 <|--符号表示。

```
@startuml
:Main Admin: as Admin
(Use the application) as (Use)

User <|-- Admin
(Start) <|-- (Use)

@enduml
```

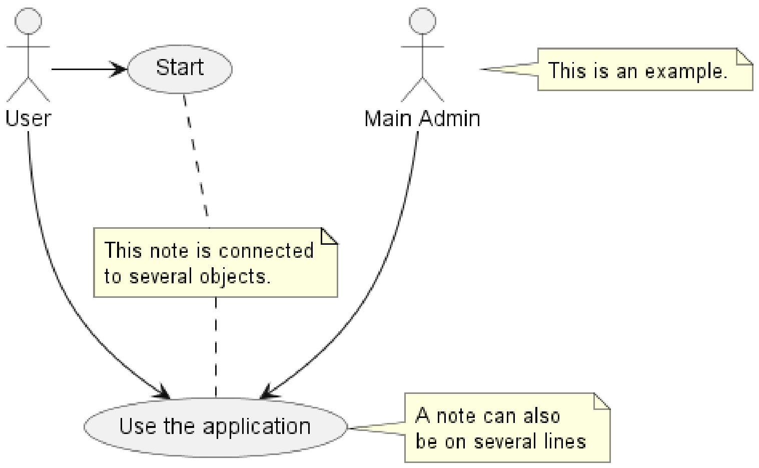


2.8 使用注释

可以用 `note left of` , `note right of` , `note top of` , `note bottom of` 等关键字给一个对象添加注释。

注释还可以通过 `note` 关键字来定义，然后用`..` 连接其他对象。

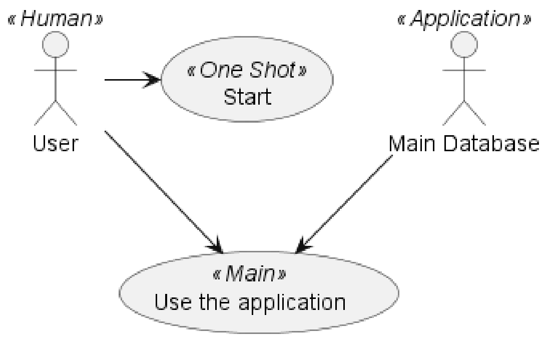
```
@startuml
:Main Admin: as Admin
(Use the application) as (Use)

User -> (Start)
User --> (Use)

Admin ---> (Use)

note right of Admin : This is an example.

note right of (Use)
  A note can also
  be on several lines
end note

note "This note is connected\nto several objects." as N2
(Start) .. N2
N2 .. (Use)
@enduml
```



2.9 构造类型

用 << 和 >> 来定义角色或者用例的构造类型。

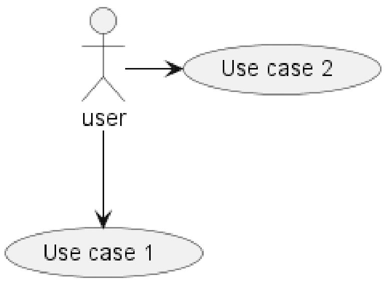
```
@startuml
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start)
User --> (Use)

MySql --> (Use)

@enduml
```



2.10 改变箭头方向

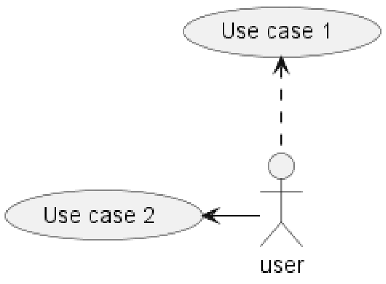
默认情况下，类之间的链接有两个破折号`--`，并且是垂直方向的。可以通过像这样放一个破折号（或点）来使用水平链接。

```
@startuml
:user: --> (Use case 1)
:user: -> (Use case 2)
@enduml
```



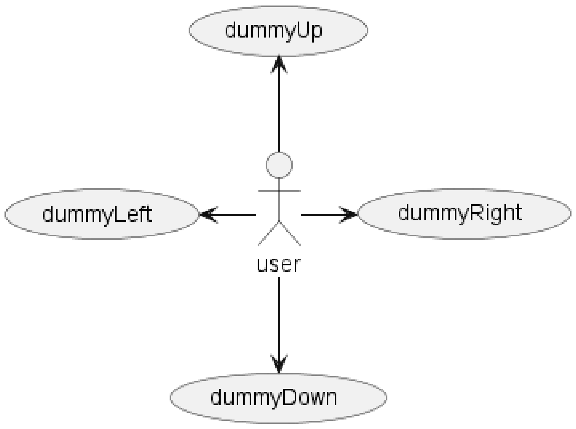
你也可以通过反转链接来改变方向。

```
@startuml
(Use case 1) <.. :user:
(Use case 2) <- :user:
@enduml
```



也可以通过在箭头内添加 `left`,`right`,`up` 或 `down` 关键字来改变箭头方向。

```
@startuml
:user: -left-> (dummyLeft)
:user: -right-> (dummyRight)
:user: -up-> (dummyUp)
:user: -down-> (dummyDown)
@enduml
```

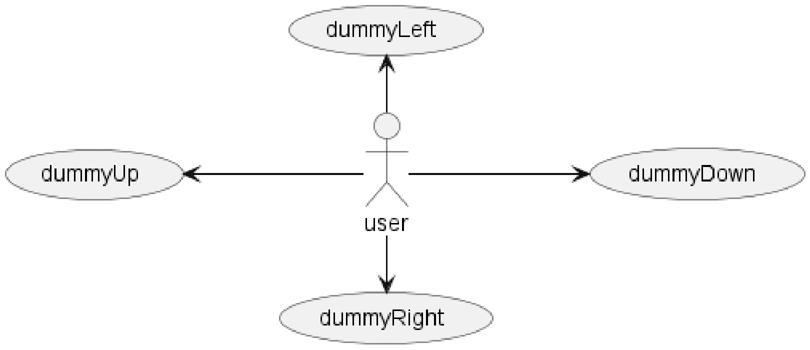


你可以通过只使用方向的第一个字符来缩短箭头（例如，`-d-` ，而不是 `-down-` ）或两个第一个字符 (`-do-`)。

请注意，你不应该滥用这个功能：*Graphviz* 通常在没有调整的情况下给出良好的结果。

并使用 `left to right direction` 参数。

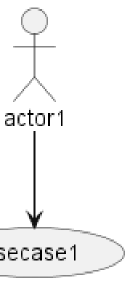
```
@startuml
left to right direction
:user: -left-> (dummyLeft)
:user: -right-> (dummyRight)
:user: -up-> (dummyUp)
:user: -down-> (dummyDown)
@enduml
```



2.11 分割图示

用 `newpage` 关键字将图示分解为多个页面。

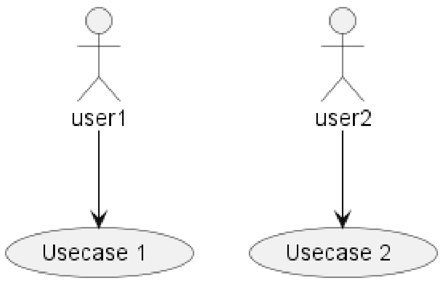
```
@startuml
:actor1: --> (Usecase1)
newpage
:actor2: --> (Usecase2)
@enduml
```



2.12 从左向右方向

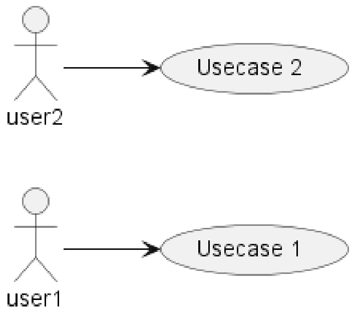
默认从上往下构建图示。

```
@startuml
'default
top to bottom direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)

@enduml
```



你可以用 `left to right direction` 命令改变图示方向。

```
@startuml

left to right direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)

@enduml
```



2.13 显示参数

用 skinparam 改变字体和颜色。

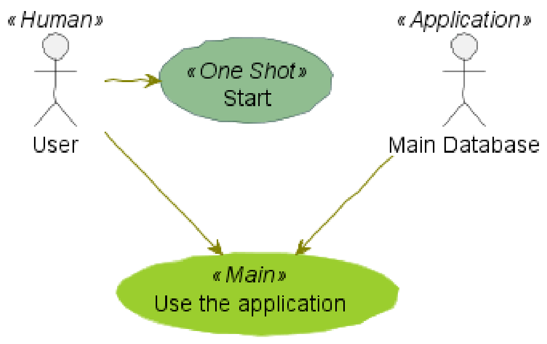
可以在如下场景中使用：

- 在图示的定义中，
- 在引入的文件中，

- 在命令行或者 ANT 任务提供的配置文件中。

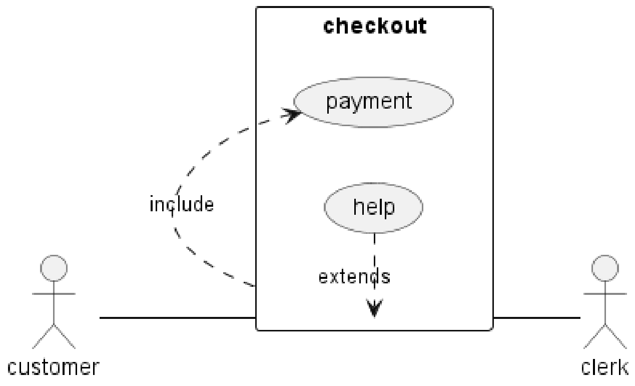
你也可以给构造的角色和用例指定特殊颜色和字体。

```
@startuml
skinparam handwritten true

skinparam usecase {
BackgroundColor DarkSeaGreen
BorderColor DarkSlateGray

BackgroundColor<< Main >> YellowGreen
BorderColor<< Main >> YellowGreen

ArrowColor Olive
ActorBorderColor black
ActorFontName Courier

ActorBackgroundColor<< Human >> Gold
}

User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start)
User --> (Use)

MySql --> (Use)

@enduml
```



2.14 完整样例

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor customer
actor clerk
rectangle checkout {
  customer -- (checkout)
  (checkout) .> (payment) : include
  (help) .> (checkout) : extends
  (checkout) -- clerk
}
@enduml
```

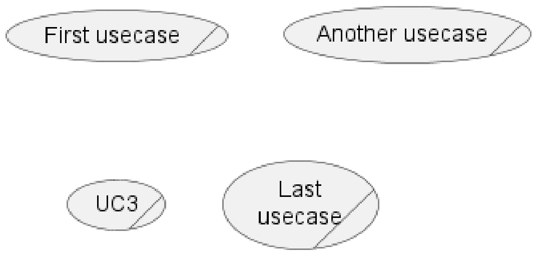


2.15 业务用例

你可以添加/ 来制作业务用例。

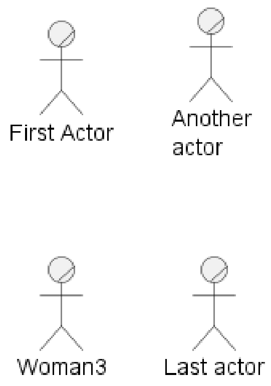
2.15.1 业务用例

```
@startuml

(First usecase)/
(Another usecase)/ as (UC2)
usecase/ UC3
usecase/ (Last\nusecase) as UC4

@enduml
```



2.15.2 商业行为者

```
@startuml

:First Actor:/
:Another\nactor:/ as Man2
actor/ Woman3
actor/ :Last actor: as Person1

@enduml
```



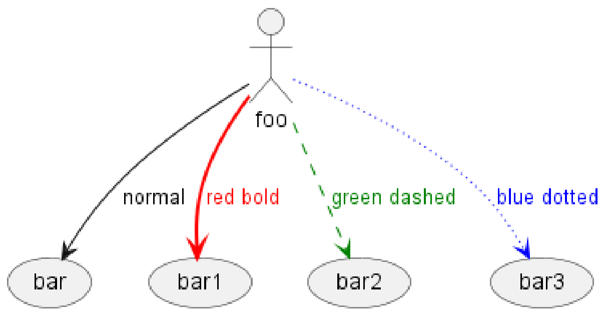
[Ref. QA-12179]

2.16 改变箭头的颜色和样式（内联样式）

你可以使用以下的内联式符号改变单个箭头的颜色或样式。

- `#color;line.[bold|dashed|dotted];text:color`

```
@startuml
actor foo
foo --> (bar) : normal
foo --> (bar1) #line:red;line.bold;text:red  : red bold
foo --> (bar2) #green;line.dashed;text:green : green dashed
foo --> (bar3) #blue;line.dotted;text:blue   : blue dotted
@enduml
```



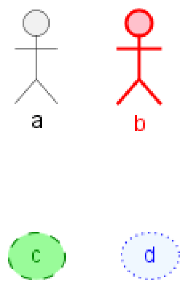
[参考 QA-3770 和 QA-3816] [参见部署图或类图的类似功能]

2.17 改变元素的颜色和样式（内联样式）

你可以用以下符号改变单个元素的颜色或样式。

- `#[color|back:color];line:color;line.[bold|dashed|dotted];text:color`

```
@startuml
actor a
actor b #pink;line:red;line.bold;text:red
usecase c #palegreen;line:green;line.dashed;text:green
usecase d #aliceblue;line:blue;line.dotted;text:blue
@enduml
```

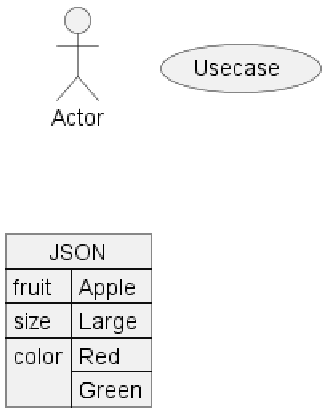


[参考 QA-5340 和改编自 QA-6852]

2.18 Display JSON Data on Usecase diagram

2.18.1 Simple example

```
@startuml
allowmixing

actor     Actor
usecase   Usecase

json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-15481]

For another example, see on JSON page.

3 类图

类图的设计语法与编程语言的传统语法相似。这种相似性为开发人员提供了一个熟悉的环境，从而使创建图表的过程更简单、更直观。

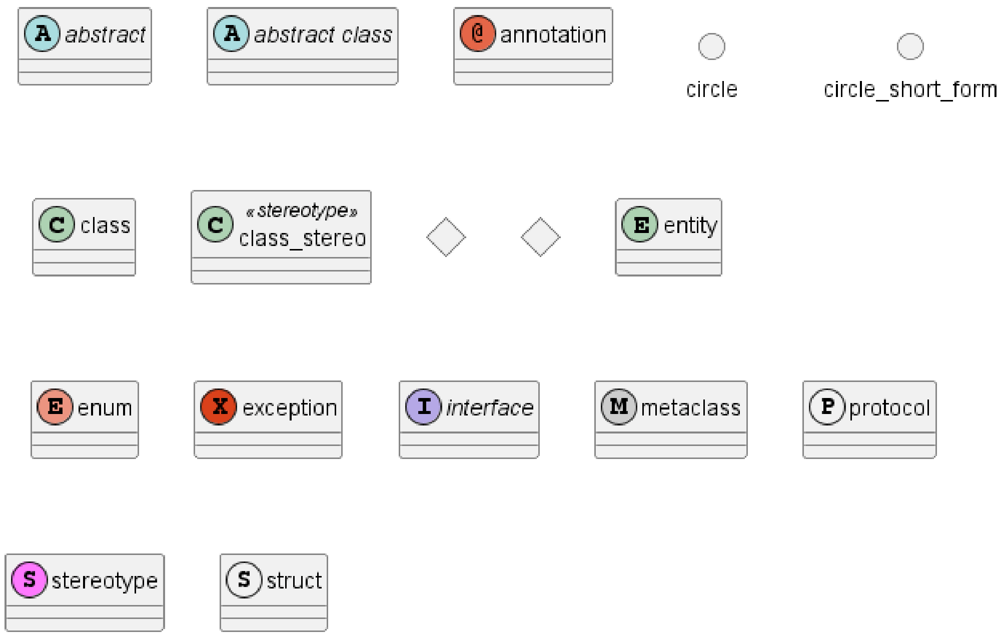
这种设计方法不仅简洁，而且还能创建既简洁又富有表现力的表述。此外，它还允许通过与序列图相呼应的语法来描绘类之间的关系，为流畅而深刻地描绘类之间的交互铺平了道路。

除了结构和关系表示法，类图语法还支持进一步的丰富，如包含注释和应用颜色，使用户能够创建信息丰富且视觉上吸引人的图表。

您可以了解更多有关 PlantUML 中一些常用命令的信息，以增强您的图表创建体验。

3.1 元素声明

```
@startuml
abstract        abstract
abstract class  "abstract class"
annotation      annotation
circle          circle
()              circle_short_form
class           class
class           class_stereo  <<stereotype>>
diamond         diamond
<>              diamond_short_form
entity          entity
enum            enum
exception       exception
interface       interface
metaclass       metaclass
protocol        protocol
stereotype      stereotype
struct          struct
@enduml
```



[参考 `protocol` 和 `struct`:GH-1028, 对于 `exception`:QA-16258]

3.2 类之间的关系

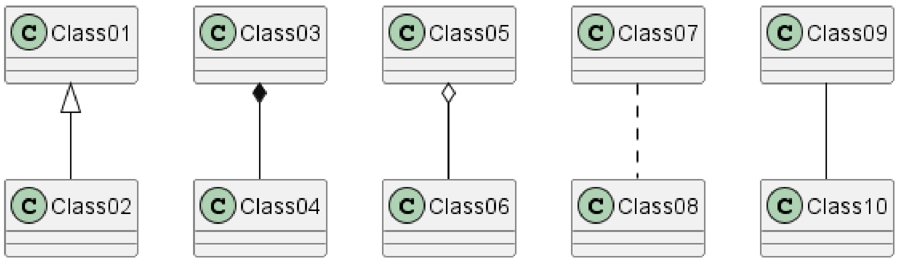
类之间的关系是用以下符号定义的。

关系类型	符号	绘图
泛化关系	<|--	◁—
组合关系	*--	◆—
聚合关系	o--	◇—

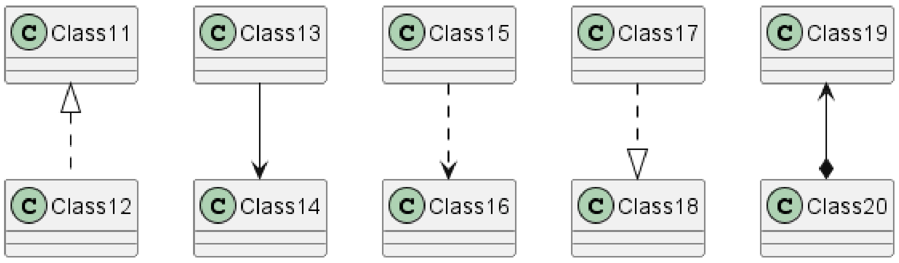
可以用.. 来代替-- ，会显示为虚线。

示例：

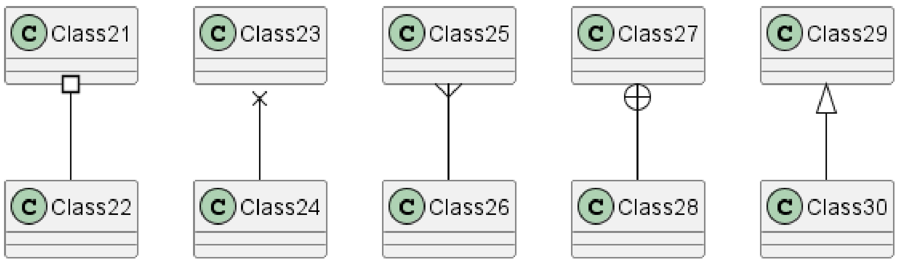
```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```



```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```



```
@startuml
Class21 #-- Class22
Class23 x-- Class24
Class25 }-- Class26
Class27 +-- Class28
Class29 ^-- Class30
@enduml
```

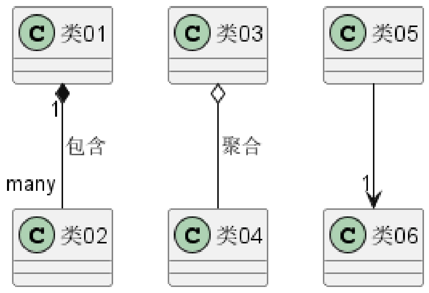


3.3 关系上的标签

在关系之间使用标签来说明时, 使用 : 后接标签文字。

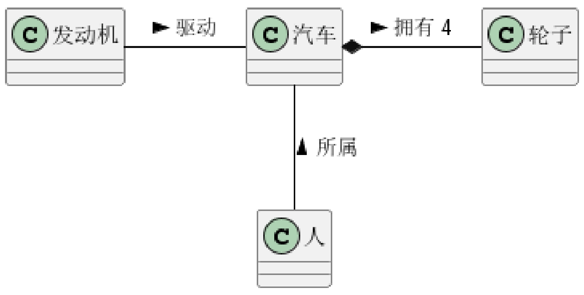
对元素的说明，你可以在每一边使用 "" 来说明.

```
@startuml

类01 "1" *-- "many" 类02 : 包含

类03 o-- 类04 : 聚合

类05 --> "1" 类06

@enduml
```



在标签的开始或结束位置添加 < 或 > 以表明是哪个对象作用到哪个对象上。

```
@startuml
class 汽车

发动机 - 汽车 : 驱动 >
汽车 *- 轮子 : 拥有 4 >
汽车 -- 人 : < 所属

@enduml
```

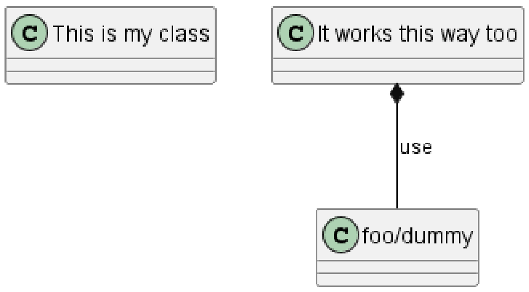


3.4 在元素名称和关系标签中使用非字母

如果你想在类 (或枚举...) 的显示名称中使用非字母，你可以：

- 在类定义中使用 as 关键字来指定一个别名
- 在类名称周围加上引号""

```
@startuml
class "This is my class" as class1
class class2 as "It works this way too"

class2 *-- "foo/dummy" : use
@enduml
```

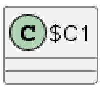


如果一个别名被分配给一个元素，文件的其余部分必须用别名而不是名称来指代该元素。

3.4.1 以 $

3.4.2 开始的名称注意，以 $ 开始的名称以后不能被隐藏或删除，因为 **hide** 和 **remove** 命令会认为该名称是 **$tag** 而不是一个组件名称。要想以后删除这些元素，它们必须有一个别名或必须被标记。

```
@startuml
class $C1
class $C2 $C2
class "$C2" as dollarC2
remove $C1
remove $C2
remove dollarC2
@enduml
```



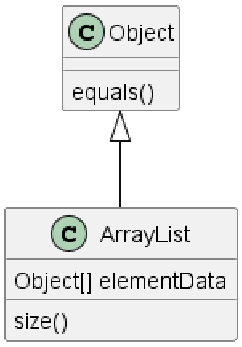
还要注意的是，以 $ 开始的名字是有效的，但是要给这样的元素分配一个别名，必须把名字放在引号"" 之间。

3.5 添加方法

要声明属性和方法，你可以使用符号:，后面跟字段或方法的名称。

编译器会通过检查括号来选择方法和字段。

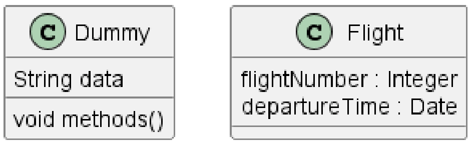
```
@startuml
Object <|-- ArrayList

Object : equals()
ArrayList : Object[] elementData
ArrayList : size()

@enduml
```



可用花括号 {} 为所有属性和方法分组。

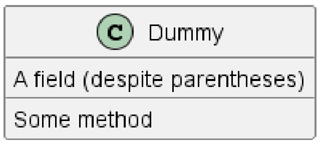
注意，语法对类型/名称的顺序有很大的灵活性。

```
@startuml
class Dummy {
  String data
  void methods()
}

class Flight {
   flightNumber : Integer
   departureTime : Date
}
@enduml
```



你可以使用 `{field}` 和 `{method}` 修饰符来覆盖编译器对属性和方法的默认识别。

```
@startuml
class Dummy {
  {field} A field (despite parentheses)
  {method} Some method
}

@enduml
```

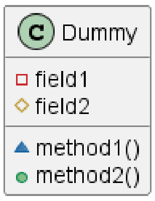


3.6 定义能见度 (可访问性)

当你定义属性或者方法时，你可以使用特殊符号定义相应条目的可访问性质。

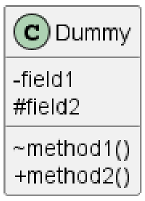
字符	图标 (属性)	图标 (方法)	可访问性
-	□	■	private 私有
#	◇	◆	protected 受保护
~	△	▲	package private 包内可见
+	○	●	public 公有

```
@startuml

class Dummy {
 -field1
 #field2
 ~method1()
 +method2()
}

@enduml
```



你可以采用命令 `skinparam classAttributeIconSize 0` 来展示特殊符号本身：

```
@startuml
skinparam classAttributeIconSize 0
class Dummy {
 -field1
 #field2
 ~method1()
 +method2()
}

@enduml
```

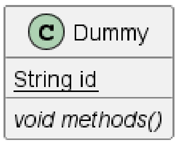


3.7 抽象与静态

通过修饰符 `{static}` 或者 `{abstract}`，可以定义静态或者抽象的方法或者属性。

这些修饰符可以写在行的开始或者结束。也可以使用 `{classifier}` 这个修饰符来代替 `{static}`.

```
@startuml
class Dummy {
  {static} String id
  {abstract} void methods()
}
@enduml
```

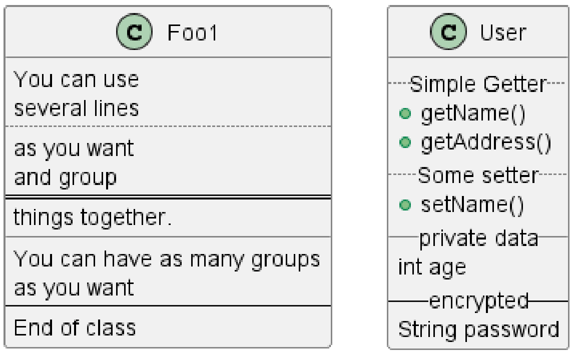


3.8 高级类体

PlantUML 默认自动将方法和属性重新分组，你可以自己定义分隔符来重排方法和属性，下面的分隔符都是可用的：`--` `..` `==` `__`.

还可以在分隔符中添加标题：

```
@startuml
class Foo1 {
  You can use
  several lines
  ..
  as you want
  and group
  ==
  things together.
  --
  You can have as many groups
  as you want
  --
  End of class
}

class User {
  .. Simple Getter ..
  + getName()
  + getAddress()
  .. Some setter ..
  + setName()
  __ private data __
  int age
  -- encrypted --
  String password
}

@enduml
```



3.9 备注和版型

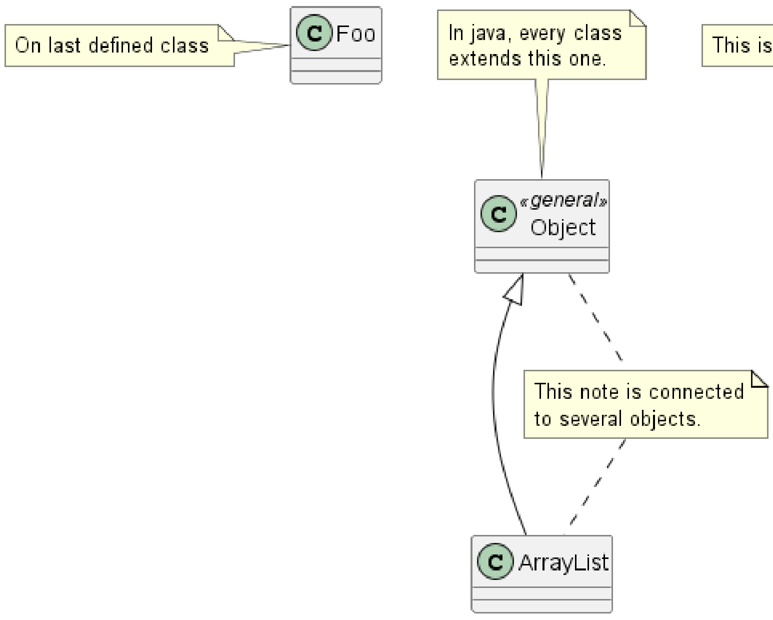
版型通过类关键字 (”<<” 和”>>”) 来定义

你可以使用 `note left of` , `note right of` , `note top of` , `note bottom of` 这些关键字来添加备注。

你还可以在类的声明末尾使用 `note left`, `note right`,`note top`, `note bottom` 来添加。

此外，单独用 `note` 这个关键字也是可以的，使用 `..` 符号可以作出一条连接它与其它对象的虚线。

```
@startuml
class Object << general >>
Object <|--- ArrayList

note top of Object : In java, every class\nextends this one.

note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2
Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class

@enduml
```


On last defined class
Foo
In java, every class extends this one.
This is a floating note
«general» Object
This note is connected to several objects.
ArrayList

3.10 备注中的更多功能

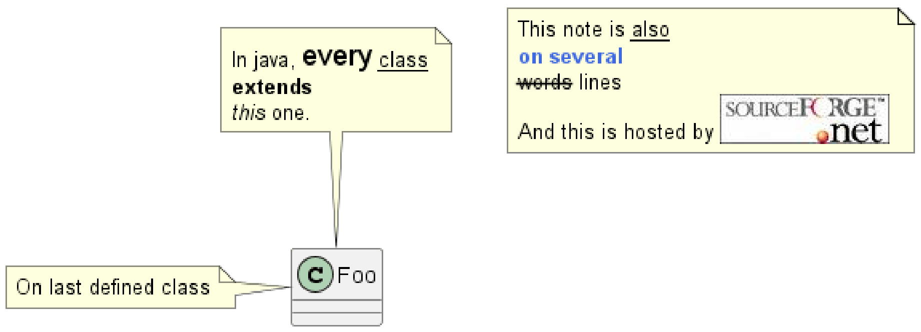
可以在注释中使用部分 html 标签 (See Creole expression):

- `<b>`
- `<u>`
- `<i>`
- `<s>`, `<del>`, `<strike>`
- `<font color="#AAAAAA">` or `<font color="colorName">`
- `<color:#AAAAAA>` or `<color:colorName>`
- `<size:nn>` to change font size
- `<img src="file">` or `<img:file>`: the file must be accessible by the filesystem

你也可以在注释中展示多行。

你也可以在定义的 class 之后直接使用 `note left`, `note right`, `note top`, `note bottom` 来定义注释。

```
@startuml

class Foo
note left: On last defined class

note top of Foo
  In java, <size:18>every</size> <u>class</u>
  <b>extends</b>
  <i>this</i> one.
end note

note as N1
  This note is <u>also</u>
  <b><color:royalBlue>on several</color>
  <s>words</s> lines
  And this is hosted by <img:sourceforge.jpg>
end note

@enduml
```



3.11 注释属性 (field, attribute, member) 或方法

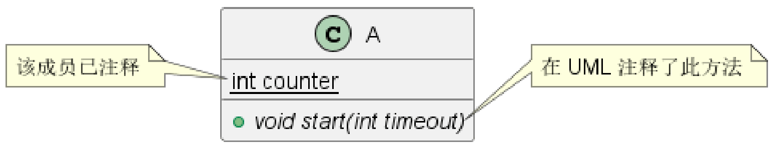
可以在属性（field、attribute、member）或方法上添加注释。

3.11.1 注意

- 不能与 top 和 bottom 同时使用 (只支持 *left* 和 *right*)
- 不能与表示命名空间的分隔符:: 同时使用

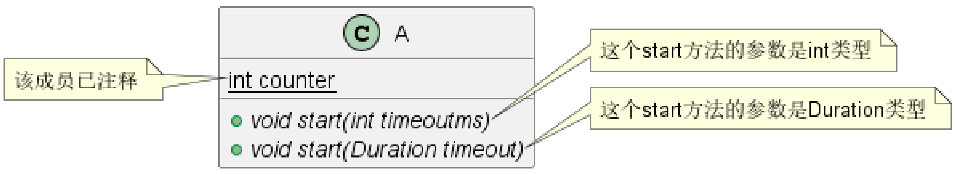
3.11.2 注释属性或方法

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeout)
}
note left of A::counter
  该成员已注释
end note
note right of A::start
  在 UML 注释了此方法
end note
@enduml
```



3.11.3 给同名方法注释

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeoutms)
+void {abstract} start(Duration timeout)
}
note left of A::counter
  该成员已注释
end note
note right of A::"start(int timeoutms)"
  这个start方法的参数是int类型
end note
note right of A::"start(Duration timeout)"
  这个start方法的参数是Duration类型
end note
@enduml
```


A
int counter
void start(int timeoutms)
void start(Duration timeout)
该成员已注释
这个start方法的参数是int类型
这个start方法的参数是Duration类型

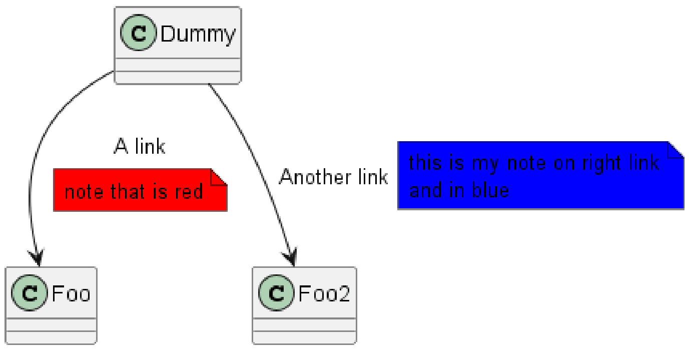
[Ref. QA-3474 and QA-5835]

3.12 链接的注释

在定义链接之后，你可以用 `note on link` 给链接添加注释

如果想要改变注释相对于标签的位置，你也可以用 `note left on link`，`note right on link`，`note bottom on link`。(对应位置分别在 label 的左边，右边，下边)

```
@startuml

class Dummy
Dummy --> Foo : A link
note on link #red: note that is red

Dummy --> Foo2 : Another link
note right on link #blue
this is my note on right link
and in blue
end note

@enduml
```



3.13 抽象类和接口

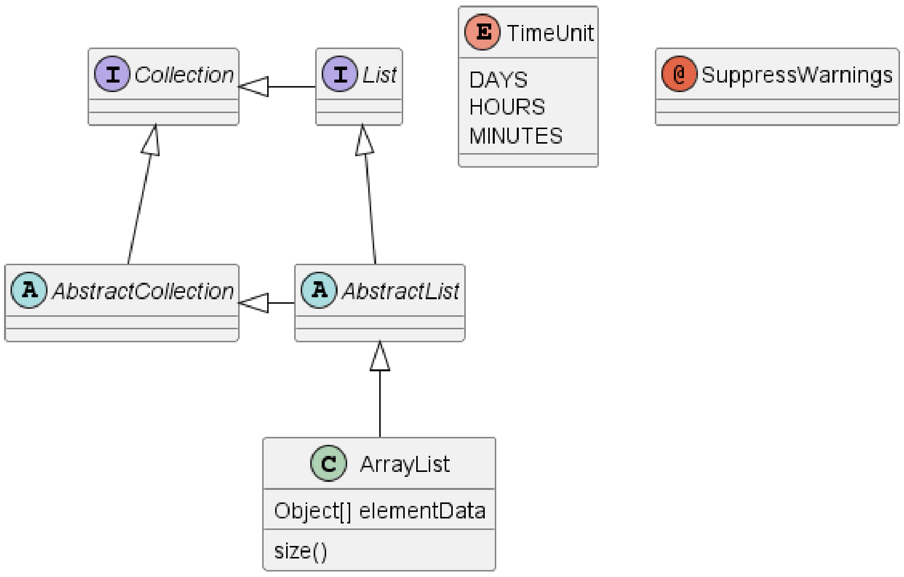
用关键字 `abstract` 或 `abstract class` 来定义抽象类。

抽象类用斜体显示。

也可以使用 `interface`, `annotation` 和 `enum` 等关键字。

```
@startuml

abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

List <|-- AbstractList
Collection <|-- AbstractCollection

Collection <|- List
AbstractCollection <|- AbstractList
```

```
AbstractList <|-- ArrayList

class ArrayList {
  Object[] elementData
  size()
}

enum TimeUnit {
  DAYS
  HOURS
  MINUTES
}

annotation SuppressWarnings

@enduml
```



[参考: ' 标注和成员' Issue#458]

3.14 隐藏属性、函数等

通过使用命令 “`hide/show`”，你可以用参数表示类的显示方式。

基础命令是: `hide empty members`. 这个命令会隐藏空白的方法和属性。

除 `empty members` 外，你可以用:

- `empty fields` 或者 `empty attributes` 空属性,
- `empty methods` 空函数，
- `fields` 或 `attributes` 隐藏字段或属性，即使是被定义了
- `methods` 隐藏方法，即使是被定义了
- `members` 隐藏字段 和 方法，即使是被定义了
- `circle` 类名前带圈的，
- `stereotype` 原型。

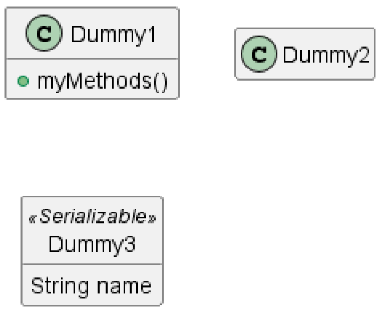
同样可以使用 `hide` 或 `show` 关键词，对以下内容进行设置:

- `class` 所有类，
- `interface` 所有接口，

- enum 所有枚举，
- <<foo1>> 实现 *foo1* 的类，
- 一个既定的类名。

你可以使用 `show/hide` 命令来定义相关规则和例外。

```
@startuml

class Dummy1 {
  +myMethods()
}

class Dummy2 {
  +hiddenMethod()
}

class Dummy3 <<Serializable>> {
String name
}

hide members
hide <<Serializable>> circle
show Dummy1 methods
show <<Serializable>> fields

@enduml
```

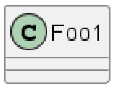


3.15 隐藏类

你也可以使用 `show/hide` 命令来隐藏类

如果你定义了一个大的!included 文件，且想在文件包含之后隐藏部分类，该功能会很有帮助。

```
@startuml

class Foo1
class Foo2

Foo2 *-- Foo1

hide Foo2

@enduml
```

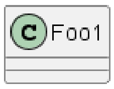


3.16 删除类

您还可以使用 `remove` 命令来删除类。

如果您定义了一个大的 [!included file]（预处理），并且如果您想在 [file contains]（预处理）之后删除一些类，这可能很有用。

```
@startuml

class Foo1
class Foo2

Foo2 *-- Foo1

remove Foo2

@enduml
```

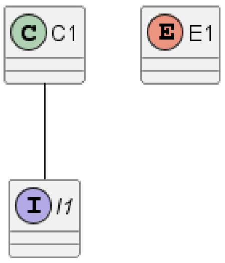


3.17 Hide, Remove or Restore tagged element or wildcard

You can put `$tags` (using `$`) on elements, then remove, hide or restore components either individually or by tags.

By default, all components are displayed:

```
@startuml
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1
@enduml
```

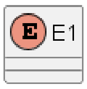


But you can:

- `hide $tag13` components:

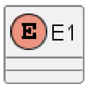
```
@startuml
class C1 $tag13
enum E1
interface I1 $tag13
```

```
C1 -- I1

hide $tag13
@enduml
```



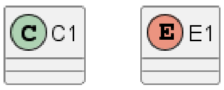
- or `remove $tag13` components:

```
@startuml
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1

remove $tag13
@enduml
```



- or `remove $tag13 and restore $tag1` components:

```
@startuml
class C1 $tag13 $tag1
enum E1
interface I1 $tag13
C1 -- I1

remove $tag13
restore $tag1
@enduml
```



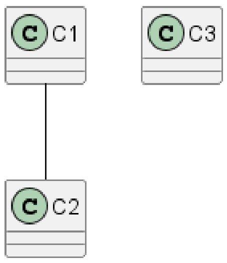
- or `remove * and restore $tag1` components:

```
@startuml
class C1 $tag13 $tag1
enum E1
interface I1 $tag13
C1 -- I1

remove *
restore $tag1
@enduml
```

3.18 隐藏或删除未关联的类

默认情况下, 所有的类都将会展示:

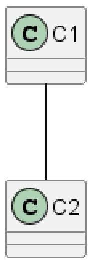
```
@startuml
class C1
class C2
class C3
C1 -- C2
@enduml
```



不过你可以使用:

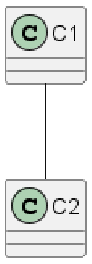
- `hide @unlinked` 来隐藏未关联的类:

```
@startuml
class C1
class C2
class C3
C1 -- C2

hide @unlinked
@enduml
```



- 或者使用 `remove @unlinked` 来删除未关联的类:

```
@startuml
class C1
class C2
class C3
C1 -- C2

remove @unlinked
@enduml
```

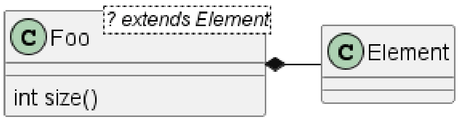


[Adapted from QA-11052]

3.19 泛型（generics）

你可以用 `<` 和 `>` 来定义类的泛型。

```
@startuml

class Foo<? extends Element> {
  int size()
}
Foo *- Element

@enduml
```



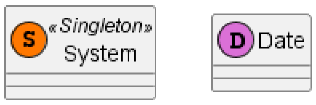
It is possible to disable this drawing using `skinparam genericDisplay old` command.

3.20 指定标记（Spot）

通常标记字符 (C, I, E or A) 用于标记类 (classes), 接口（interface）, 枚举（enum）和抽象类（abstract classes）.

但是当你想定义原型时，可以增加对应的单个字符及颜色，来定义自己的标记（spot），就像下面一样：

```
@startuml

class System << (S,#FF7700) Singleton >>
class Date << (D,orchid) >>
@enduml
```

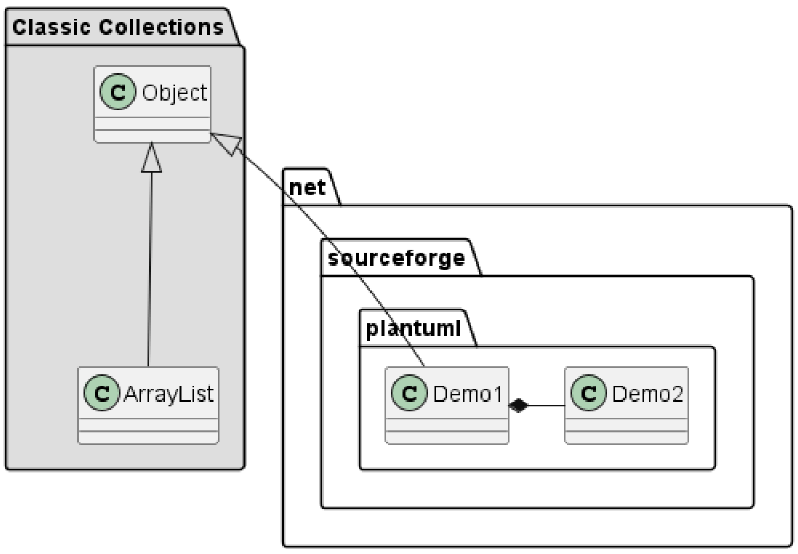


3.21 包

你可以通过关键词 `package` 声明包，同时可选的来声明对应的背景色（通过使用 html 色彩代码或名称）。

注意：包可以被定义为嵌套。

```
@startuml

package "Classic Collections" #DDDDDD {
  Object <|-- ArrayList
}

package net.sourceforge.plantuml {
  Object <|-- Demo1
  Demo1 *- Demo2
}

@enduml
```

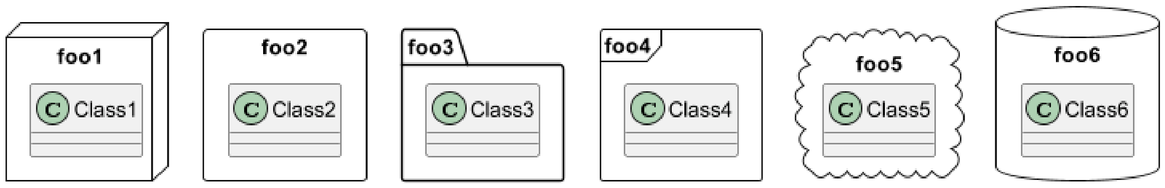


3.22 包样式

包可以定义不同的样式。

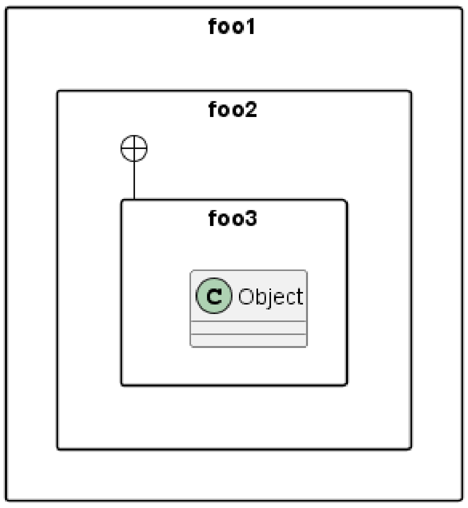
你可以通过以下的命令来设置默认样式: `skinparam packageStyle`, 或者对包使用对应的模板:

```
@startuml
scale 750 width
package foo1 <<Node>> {
  class Class1
}

package foo2 <<Rectangle>> {
  class Class2
}

package foo3 <<Folder>> {
  class Class3
}

package foo4 <<Frame>> {
  class Class4
}

package foo5 <<Cloud>> {
  class Class5
}

package foo6 <<Database>> {
  class Class6
}

@enduml
```



你也可以参考下面的示例来定义包之间的连线:

```
@startuml

skinparam packageStyle rectangle

package foo1.foo2 {
}

package foo1.foo2.foo3 {
  class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```



3.23 命名空间（Namespaces）

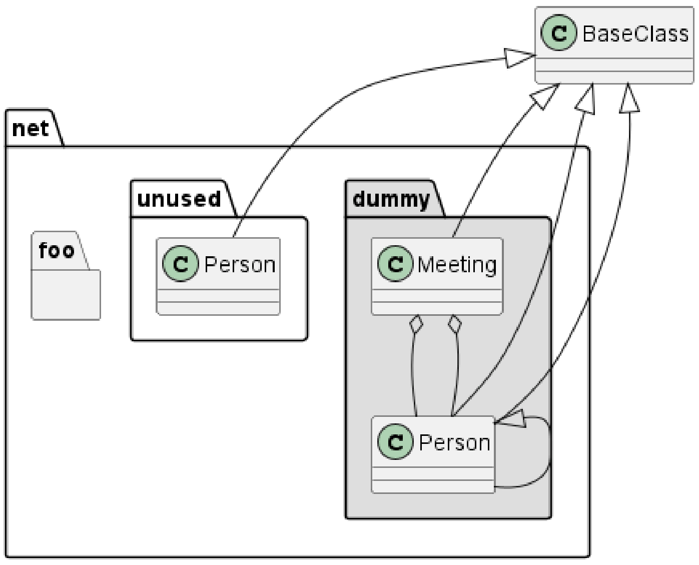
在使用包（package）时（区别于命名空间），类名是类的唯一标识。也就意味着，在不同的包（package）中的类，不能使用相同的类名。

在那种情况下（译注：同名、不同全限定名类），你应该使用命名空间来取而代之。

你可以从其他命名空间，使用全限定名来引用类，默认命名空间（译注：无名的命名空间）下的类，以一个“.”开头（的类名）来引用（译注：示例中的 `BaseClass`).

注意：你不用显示地创建命名空间：一个使用全限定名的类会自动被放置到对应的命名空间。

```
@startuml

class BaseClass

namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

    .BaseClass <|- Meeting
}

namespace net.foo {
  net.dummy.Person  <|- Person
  .BaseClass <|-- Person
  net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

@enduml
```

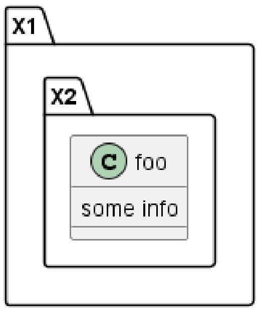


There won't be any difference between namespaces and packages anymore: both keywords are now synonymous.

3.24 自动创建命名空间

使用命令 `set namespaceSeparator ???` 你可以自定义命名空间分隔符（为 "." 以外的字符）.

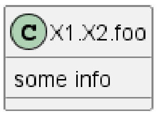
```
@startuml

set namespaceSeparator ::
class X1::X2::foo {
  some info
}

@enduml
```



禁止自动创建包则可以使用 `set namespaceSeparator none`.

```
@startuml

set namespaceSeparator none
class X1.X2.foo {
  some info
}
```

```
@enduml
```

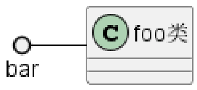


3.25 棒棒糖接口

需要定义棒棒糖样式的接口时可以遵循以下语法:

- `bar ()- foo`
- `bar ()-- foo`
- `foo -() bar`

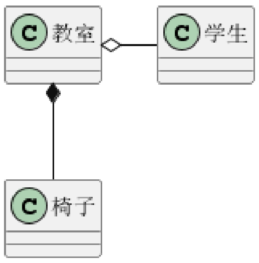
```
@startuml
class foo类
bar ()- foo类
@enduml
```



3.26 改变箭头方向

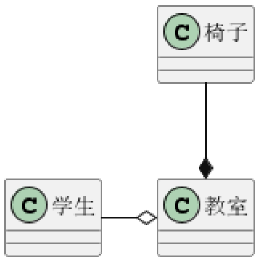
类之间默认采用两个破折号 `--` 显示出垂直方向的线. 要得到水平方向的可以像这样使用单破折号 (或者点):

```
@startuml
教室 o- 学生
教室 *-- 椅子
@enduml
```



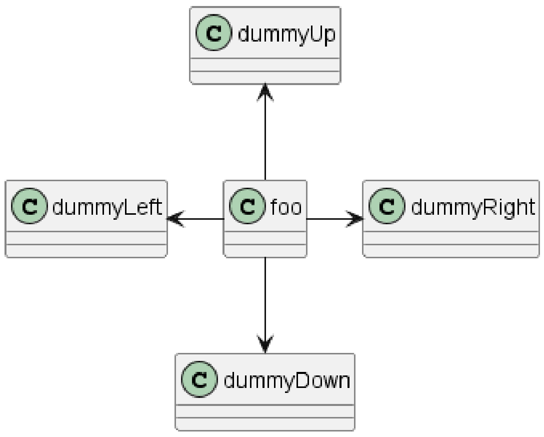
你也可以通过改变倒置链接来改变方向

```
@startuml
学生 -o 教室
椅子 --* 教室
@enduml
```



也可通过在箭头内部使用关键字，例如 `left`, `right`, `up` 或者 `down`，来改变方向

```
@startuml
foo -left-> dummyLeft
foo -right-> dummyRight
foo -up-> dummyUp
foo -down-> dummyDown
@enduml
```

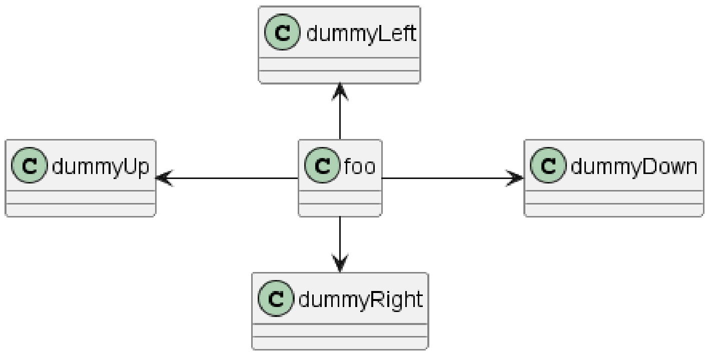


您可以使用缩写形式来表示方向，第一个字符（例如，`-d-` 而不是 `-down-`) 或前两个字符 (`-do-`)。

请注意，您不应滥用此功能：*Graphviz* 通常无需调整即可提供良好的结果。

同时也支持 `left to right direction` 参数:

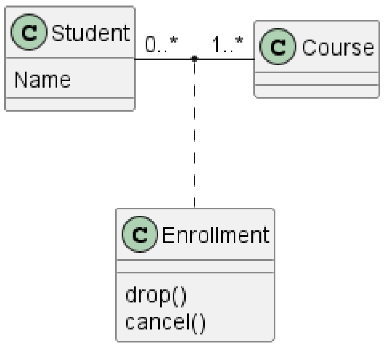
```
@startuml
left to right direction
foo -left-> dummyLeft
foo -right-> dummyRight
foo -up-> dummyUp
foo -down-> dummyDown
@enduml
```



3.27 “关系”类

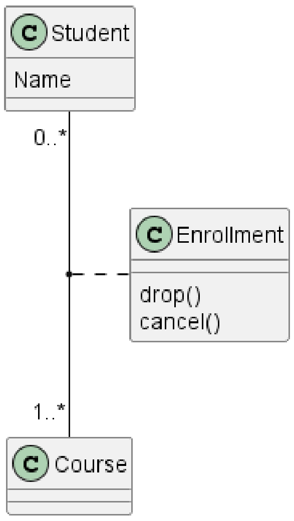
你可以在定义了两个类之间的关系后定义一个 关系类 *association class* 例如:

```
@startuml
class Student {
  Name
}
Student "0..*" - "1..*" Course
(Student, Course) .. Enrollment

class Enrollment {
  drop()
  cancel()
}
@enduml
```



也可以用另一种方式:

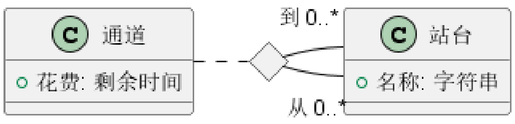
```
@startuml
class Student {
  Name
}
Student "0..*" -- "1..*" Course
(Student, Course) . Enrollment

class Enrollment {
  drop()
  cancel()
}
@enduml
```



3.28 同级关联 (Association on same classe)

```
@startuml
class 站台 {
    +名称: 字符串
}

class 通道 {
    +花费: 剩余时间
}

<> 结点
```

```
通道 . 结点
结点 - "从 0..*" 站台
结点 - "到 0..* " 站台
@enduml
```



[参考: Incubation: Associations]

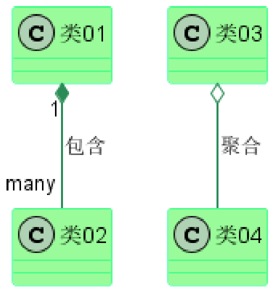
3.29 样式参数

用 skinparam 改变字体和颜色。

可以在如下场景中使用:

- 在图示的定义中,
- 在引入的文件中,
- 在命令行或者 ANT 任务提供的配置文件中。

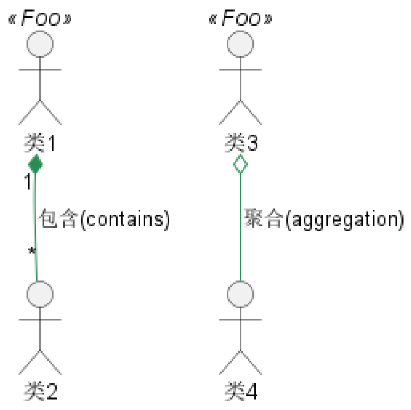
```
@startuml

skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

类01 "1" *-- "many" 类02 : 包含

类03 o-- 类04 : 聚合

@enduml
```



3.30 模板样式

你可以给模型自定义颜色和字体 (You can define specific color and fonts for stereotyped classes.)

```
@startuml

skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
BackgroundColor<<Foo>> Wheat
BorderColor<<Foo>> Tomato
}
skinparam stereotypeCBackgroundColor YellowGreen
skinparam stereotypeCBackgroundColor<< Foo >> DimGray

类1 <<Foo>>
类3 <<Foo>>
类1 "1" *-- "*" 类2 : 包含(contains)

类3 o-- 类4 : 聚合(aggregation)

@enduml
```



3.31 渐变颜色

你可以使用 # 号为类、注释等等自定义颜色。

在自定义颜色中你可以使用标准颜色的名称或 RGB 编码，参见: Colors.

你同样可以使用下面的语法为背景色声明为渐变的颜色:

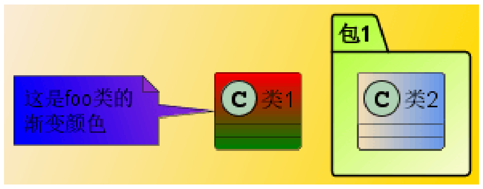
渐变的两个颜色可以使用下面的符号分割：

- |,
- /,
- \,
- -,

他们的区别在于颜色渐变的方向不同。

例如:

```
@startuml

skinparam backgroundcolor AntiqueWhite/Gold
skinparam classBackgroundColor Wheat|CornflowerBlue

class 类1 #red-green
note left of 类1 #blue\9932CC
  这是foo类的
  渐变颜色
end note

package 包1 #GreenYellow/LightGoldenRodYellow {
  class 类2
}

@enduml
```



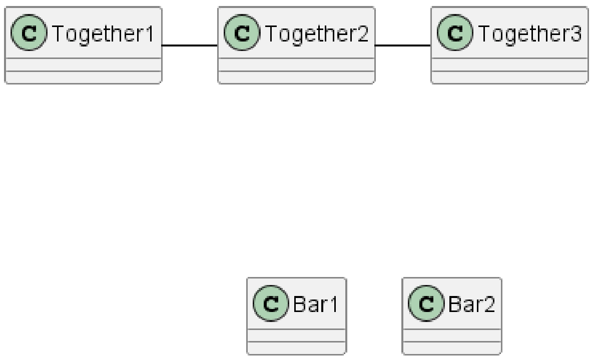
3.32 辅助布局

有时候，默认布局并不完美...

你可以使用 together 关键词将某些类进行分组：布局引擎会尝试将它们捆绑在一起（如同在一个包 (package) 内）

你也可以使用建立 hidden 链接的方式来强制布局

```
@startuml

class Bar1
class Bar2
together {
  class Together1
  class Together2
  class Together3
}
Together1 - Together2
Together2 - Together3
Together2 -[hidden]--> Bar1
Bar1 -[hidden]> Bar2


@enduml
```



3.33 拆分大文件

有些情况下，会有一些很大的图片文件。

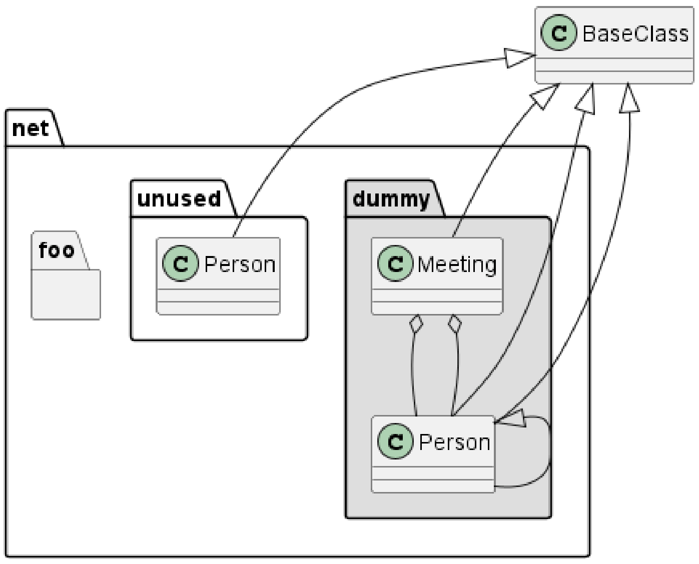
可以用 page (hpages)x(vpages) 这个命令把生成的图片文件拆分成若干个文件。

hpages 用来表示水平方向页面数，and vpages 用来表示垂直方面页面数。

你也可以使用特定的皮肤设定来给分页添加边框（见例子）

```
@startuml
' Split into 4 pages
page 2x2
skinparam pageMargin 10
skinparam pageExternalColor gray
skinparam pageBorderColor black
```

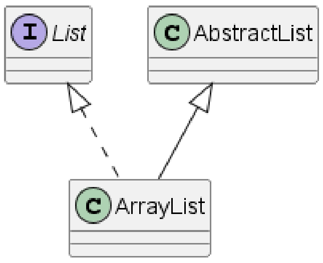
```
class BaseClass

namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

    .BaseClass <|- Meeting

}

namespace net.foo {
  net.dummy.Person  <|- Person
  .BaseClass <|-- Person

  net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person
@enduml
```



3.34 继承 (Extends) 和实现 (implements)

同样可使用 extends 和 implements 关键词.

```
@startuml
class ArrayList implements List
class ArrayList extends AbstractList
@enduml
```



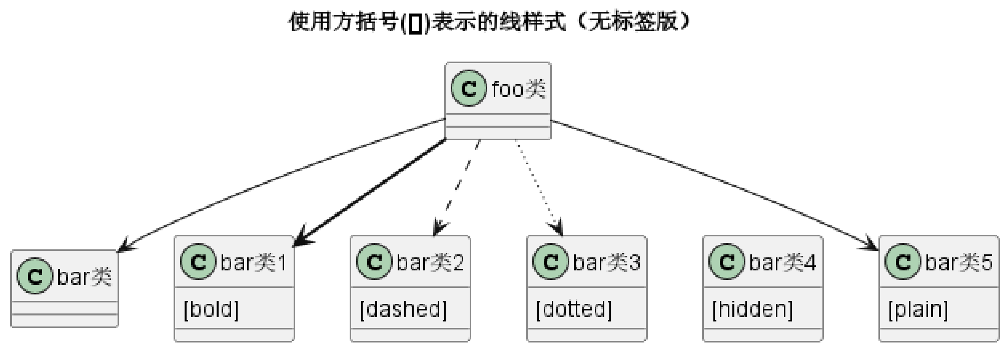
3.35 方括号表示关系 (连接或箭头) 的样式

3.35.1 线样式

可以明确的使用 `bold`, `dashed`, `dotted`, `hidden` 或 `plain` 来表示关系, 连接或箭头:

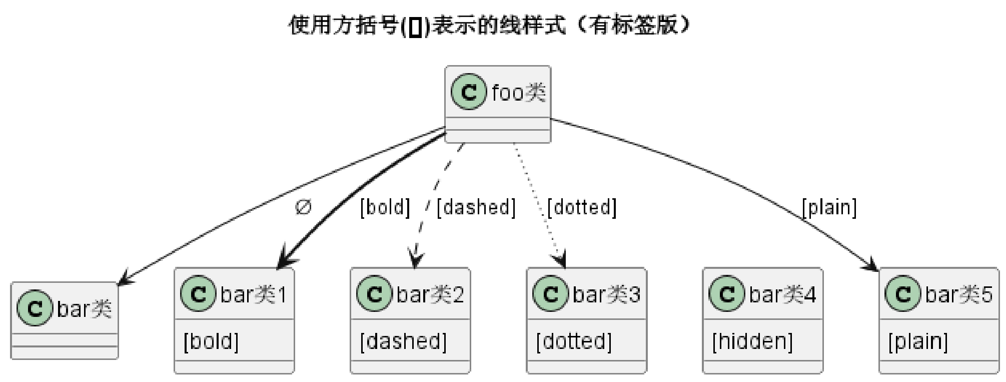
- 没有标签

```
@startuml
title 使用方括号([])表示的线样式（无标签版）

class foo类
class bar类
bar类1 : [bold]
bar类2 : [dashed]
bar类3 : [dotted]
bar类4 : [hidden]
bar类5 : [plain]

foo类 --> bar类
foo类 -[bold]-> bar类1
foo类 -[dashed]-> bar类2
foo类 -[dotted]-> bar类3
foo类 -[hidden]-> bar类4
foo类 -[plain]-> bar类5
@enduml
```



- 有标签

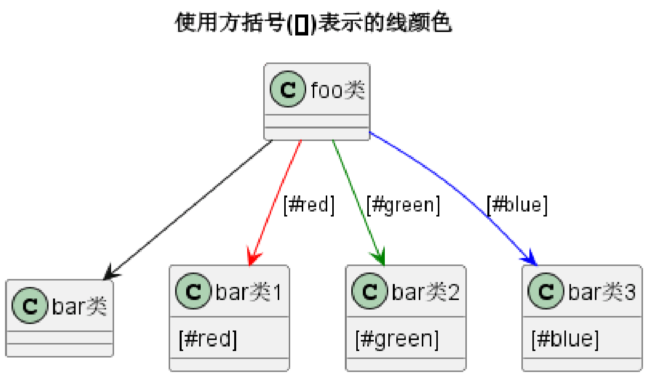
```
@startuml
title 使用方括号([])表示的线样式（有标签版）

class foo类
class bar类
bar类1 : [bold]
bar类2 : [dashed]
bar类3 : [dotted]
bar类4 : [hidden]
bar类5 : [plain]

foo类 --> bar类          :
foo类 -[bold]-> bar类1   : [bold]
foo类 -[dashed]-> bar类2 : [dashed]
foo类 -[dotted]-> bar类3 : [dotted]
foo类 -[hidden]-> bar类4 : [hidden]
foo类 -[plain]-> bar类5  : [plain]
@enduml
```



[改编自 QA-4181]

3.35.2 线颜色

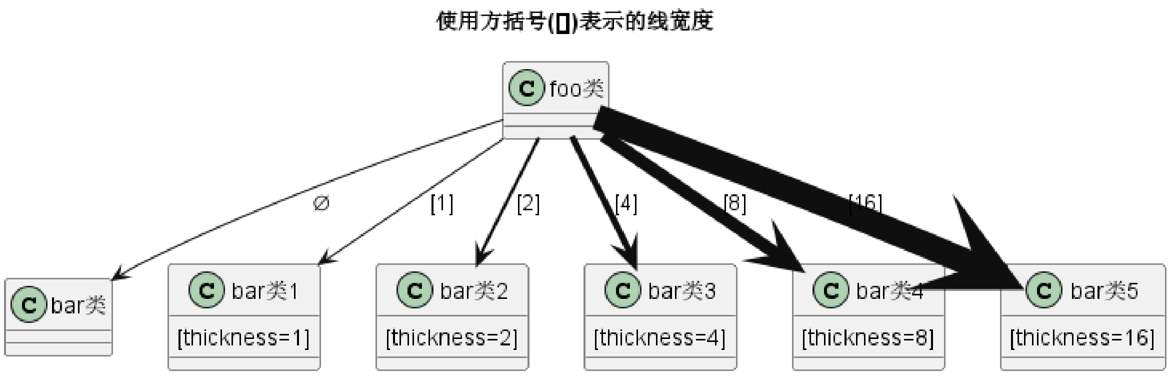
```
@startuml
title 使用方括号([])表示的线颜色
class foo类
class bar类
bar类1 : [#red]
bar类2 : [#green]
bar类3 : [#blue]

foo类 --> bar类
foo类 -[#red]-> bar类1     : [#red]
foo类 -[#green]-> bar类2   : [#green]
foo类 -[#blue]-> bar类3    : [#blue]
'foo类 -[#blue;#yellow;#green]-> bar类4
@enduml
```



3.35.3 线宽度

```
@startuml
title 使用方括号([])表示的线宽度
class foo类
class bar类
bar类1 : [thickness=1]
bar类2 : [thickness=2]
bar类3 : [thickness=4]
bar类4 : [thickness=8]
bar类5 : [thickness=16]

foo类 --> bar类                 :
foo类 -[thickness=1]-> bar类1   : [1]
```

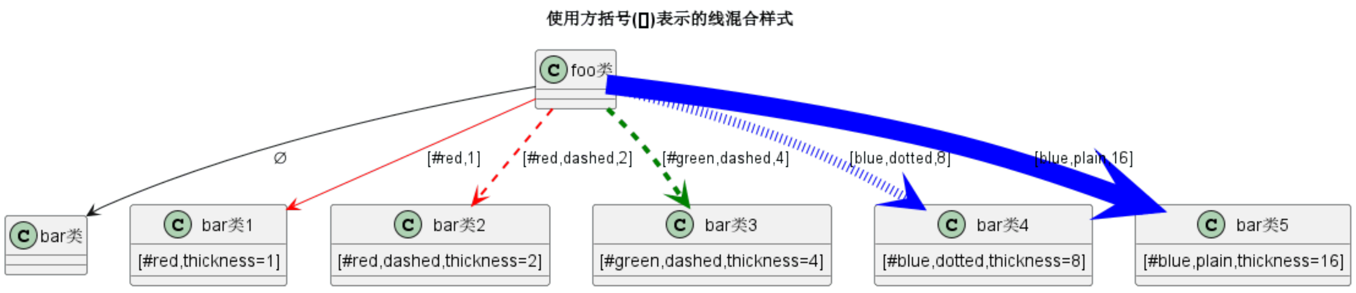
```
foo类 -[thickness=2]-> bar类2   : [2]
foo类 -[thickness=4]-> bar类3   : [4]
foo类 -[thickness=8]-> bar类4   : [8]
foo类 -[thickness=16]-> bar类5  : [16]

@enduml
```



[参考: QA-4949]

3.35.4 混合样式

```
@startuml
title 使用方括号([])表示的线混合样式
class foo类
class bar类
bar类1 : [#red,thickness=1]
bar类2 : [#red,dashed,thickness=2]
bar类3 : [#green,dashed,thickness=4]
bar类4 : [#blue,dotted,thickness=8]
bar类5 : [#blue,plain,thickness=16]

foo类 --> bar类                            :
foo类 -[#red,thickness=1]-> bar类1         : [#red,1]
foo类 -[#red,dashed,thickness=2]-> bar类2  : [#red,dashed,2]
foo类 -[#green,dashed,thickness=4]-> bar类3 : [#green,dashed,4]
foo类 -[#blue,dotted,thickness=8]-> bar类4  : [blue,dotted,8]
foo类 -[#blue,plain,thickness=16]-> bar类5  : [blue,plain,16]
@enduml
```


使用方括号([])表示的线混合样式
foo类
bar类
bar类1
[#red,thickness=1]
bar类2
[#red,dashed,thickness=2]
bar类3
[#green,dashed,thickness=4]
bar类4
[#blue,dotted,thickness=8]
bar类5
[#blue,plain,thickness=16]
∅
[#red,1]
[#red,dashed,2]
[#green,dashed,4]
[blue,dotted,8]
[blue,plain,16]

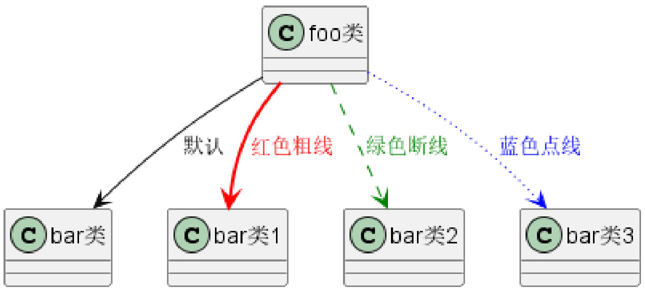
3.36 改变关系 (线和箭头) 的颜色和样式 (单行样式)

你可以改变表示关系的线和箭头的颜色或样式，使用下面的单行样式格式：

- `#color;line.[bold|dashed|dotted];text:color`

```
@startuml
class foo类
```

```
foo类 --> bar类 : 默认
foo类 --> bar类1 #line:red;line.bold;text:red  : 红色粗线
foo类 --> bar类2 #green;line.dashed;text:green : 绿色断线
foo类 --> bar类3 #blue;line.dotted;text:blue   : 蓝色点线
@enduml
```



[See similar feature on deployment]

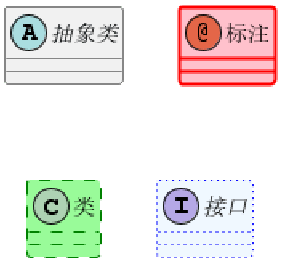
3.37 改变类颜色和样式 (单行样式)

你可以改变定义类的颜色或样式，通过下面两种指定格式:

- `#color ##[style]color`

第一个颜色 (`#color`) 表示背景色，然后第二个表示线的样式和颜色 (`##[style]color`)

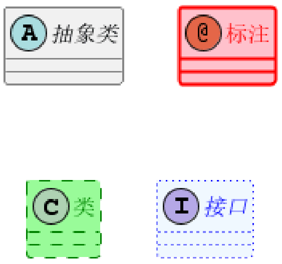
```
@startuml
abstract   抽象类
annotation 标注     #pink ##[bold]red
class      类       #palegreen ##[dashed]green
interface  接口     #aliceblue ##[dotted]blue
@enduml
```



[参考: QA-1487]

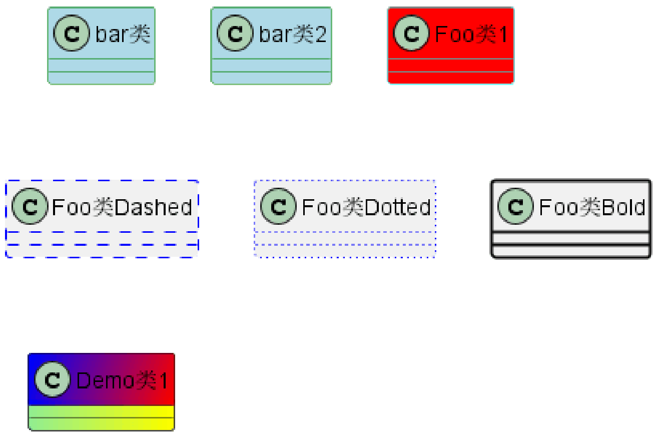
- `#[color|back:color];header:color;line:color;line.[bold|dashed|dotted];text:color`

```
@startuml
abstract   抽象类
annotation 标注     #pink;line:red;line.bold;text:red
class      类       #palegreen;line:green;line.dashed;text:green
interface  接口     #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



第一个原始示例:

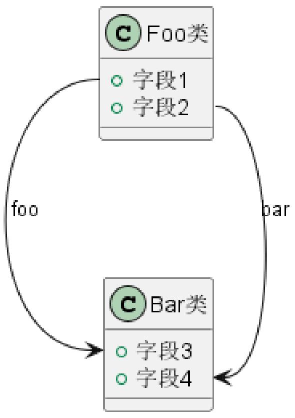
```
@startuml
class bar类 #line:green;back:lightblue
class bar类2 #lightblue;line:green

class Foo类1 #back:red;line:00FFFF
class Foo类Dashed #line.dashed:blue
class Foo类Dotted #line.dotted:blue
class Foo类Bold #line.bold
class Demo类1 #back:lightgreen|yellow;header:blue/red
@enduml
```



[参考. QA-3770]

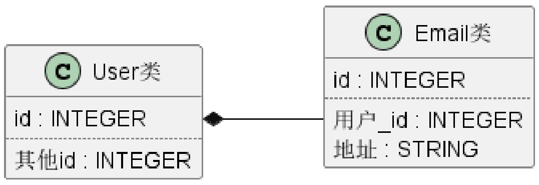
3.38 类成员的箭头方向 (Arrows from/to class members)

```
@startuml
class Foo类 {
+ 字段1
+ 字段2
}

class Bar类 {
+ 字段3
+ 字段4
}

Foo类::字段1 --> Bar类::字段3 : foo
Foo类::字段2 --> Bar类::字段4 : bar
@enduml
```



[参考: QA-3636]

```
@startuml
left to right direction

class User类 {
  id : INTEGER
  ..
  其他id : INTEGER
}

class Email类 {
  id : INTEGER
  ..
  用户_id : INTEGER
  地址 : STRING
}

User类::id *-- Email类::用户_id
@enduml
```



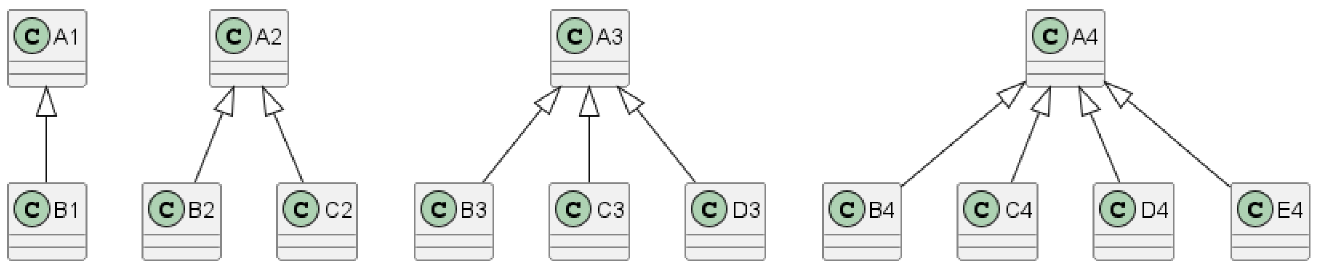
[参考: QA-5261]

3.39 分组继承关系 (Grouping inheritance arrow heads)

你可以用 `skinparam groupInheritance` 关键字合并泛化箭头, 后接参数合并阈值 (从几个继承类时开始合并)。

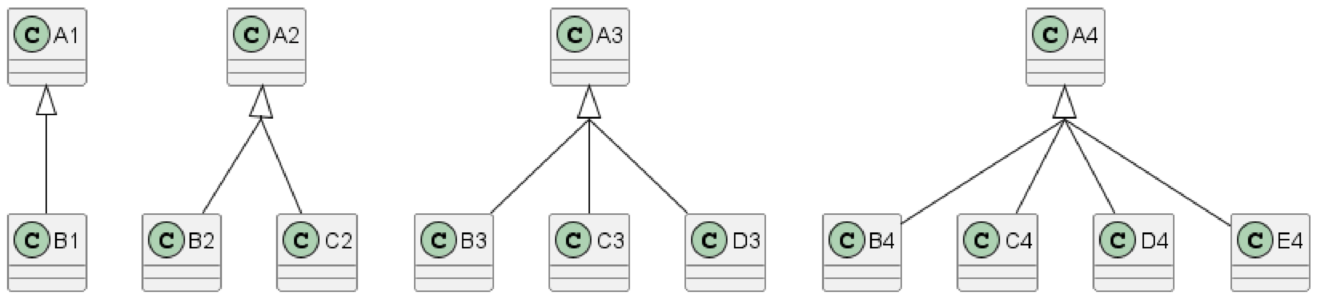
3.39.1 GroupInheritance 1 (不合并)

```
@startuml
skinparam groupInheritance 1

A1 <|-- B1

A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
@enduml
```



3.39.2 GroupInheritance 2 (从 2 个组开始合并)

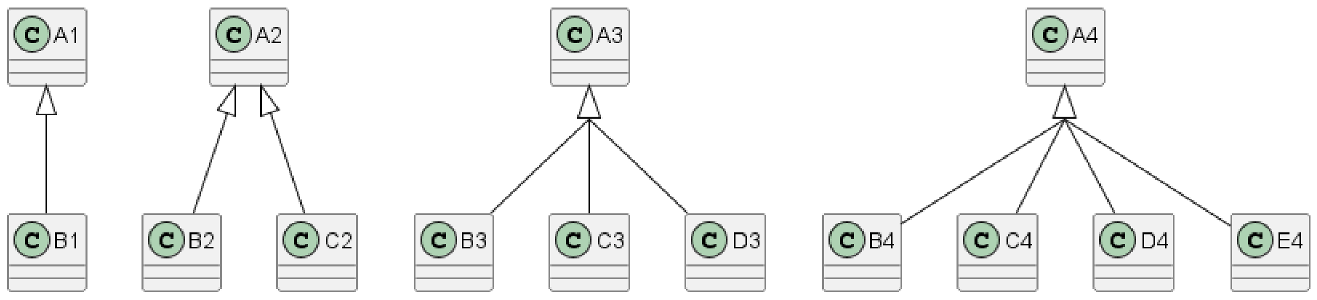
```
@startuml
skinparam groupInheritance 2

A1 <|-- B1

A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
@enduml
```



3.39.3 GroupInheritance 3 (从 3 个组开始合并)

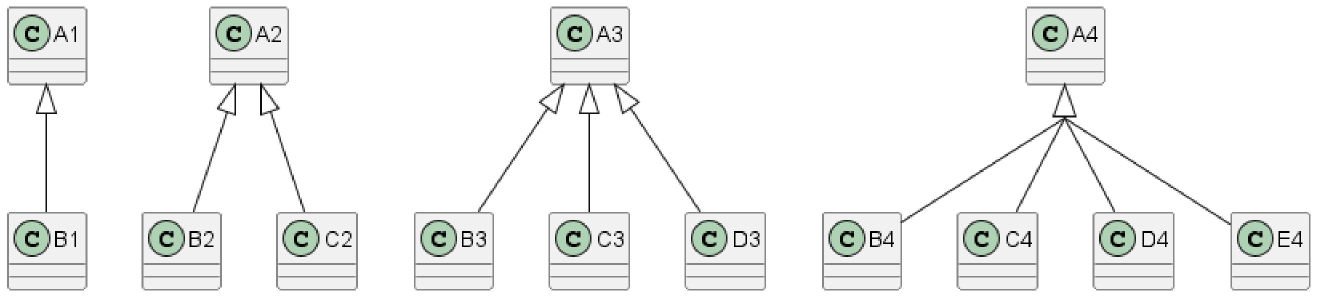
```
@startuml
skinparam groupInheritance 3

A1 <|-- B1

A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
```

```
@enduml
```



3.39.4 GroupInheritance 4 (从 4 个组开始合并)

```
@startuml
skinparam groupInheritance 4

A1 <|-- B1

A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
@enduml
```

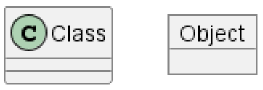


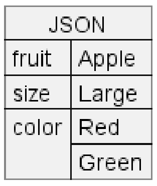
[参考: QA-3193, 和 QA-13532]

3.40 Display JSON Data on Class or Object diagram

3.40.1 Simple example

```
@startuml
class Class
object Object
json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```





JSON	
fruit	Apple
size	Large
color	Red
	Green

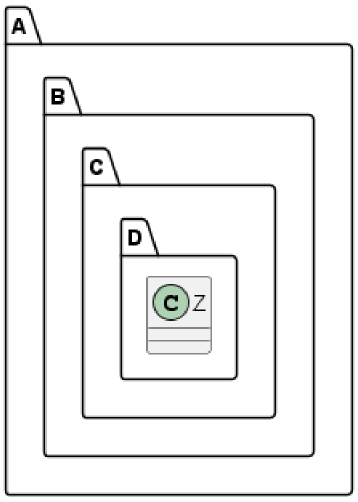
[Ref. QA-15481]

For another example, see on JSON page.

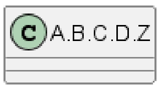
3.41 Packages and Namespaces Enhancement

[From V1.2023.2+, and V1.2023.5]

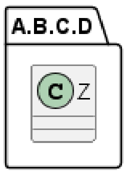
```
@startuml
class A.B.C.D.Z {
}
@enduml
```



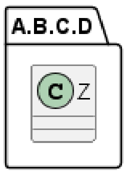
```
@startuml
set separator none
class A.B.C.D.Z {
}
@enduml
```



```
@startuml
!pragma useIntermediatePackages false
class A.B.C.D.Z {
}
@enduml
```



```
@startuml
set separator none
package A.B.C.D {
  class Z {
  }
}
@enduml
```



[Ref. GH-1352]

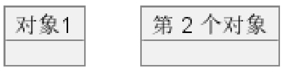
4 对象图

PlantUML offers a simple and intuitive way to create object diagrams using plain text. Its user-friendly syntax allows for quick diagram creation without the need for complex GUI tools. Moreover, the [PlantUML forum](https://forum.plantuml.net/) provides a platform for users to discuss, share, and seek assistance, fostering a collaborative community. By choosing PlantUML, users benefit from both the efficiency of markdown-based diagramming and the support of an active community.

4.1 对象的定义

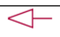
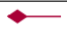
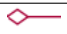
使用关键字 object 定义实例。

```
@startuml
object 对象1
object "第 2 个对象" as o2
@enduml
```



4.2 对象之间的关系

对象之间的关系是用以下符号定义的。

类型	符号	图像
延伸	<|--	
组成	*--	
聚合	o--	

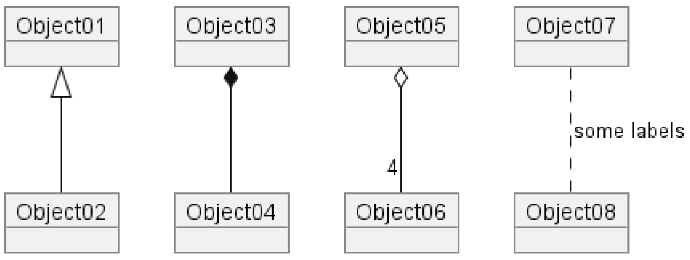
可以用.. 替换-- ，以获得虚线。

知道了这些规则，就有可能画出下面的图画。

可以在关系上添加一个标签，使用：，后面是标签的文字。

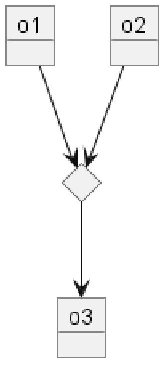
对于 cardinality，你可以在关系的每一侧使用双引号"" ，。

```
@startuml
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 .. Object08 : some labels
@enduml
```



4.3 关联对象

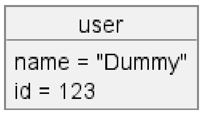
```
@startuml
object o1
object o2
diamond dia
object o3

o1  --> dia
o2  --> dia
dia --> o3
@enduml
```



4.4 添加属性

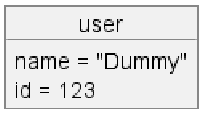
用冒号加属性名的形式声明属性。

```
@startuml

object user

user : name = "Dummy"
user : id = 123

@enduml
```



也可以用大括号批量声明属性。

```
@startuml

object user {
  name = "Dummy"
  id = 123
}

@enduml
```



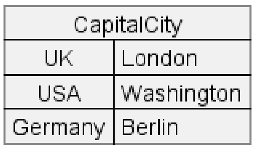
4.5 类图中的通用特性

- 可见性
- 定义注释
- 使用包
- 美化输出内容

4.6 Map table or associative array

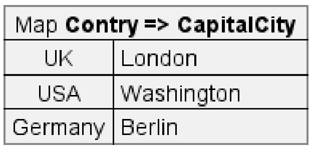
You can define a map table or associative array, with `map` keyword and `=>` separator.

```
@startuml
map CapitalCity {
 UK => London
 USA => Washington
 Germany => Berlin
}
@enduml
```



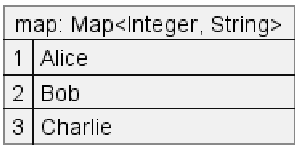
CapitalCity	
UK	London
USA	Washington
Germany	Berlin

```
@startuml
map "Map **Contry => CapitalCity**" as CC {
 UK => London
 USA => Washington
 Germany => Berlin
}
@enduml
```



Map **Contry => CapitalCity**	
UK	London
USA	Washington
Germany	Berlin

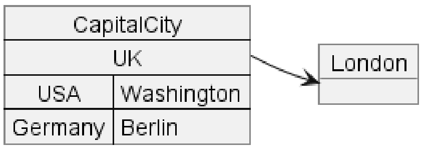
```
@startuml
map "map: Map<Integer, String>" as users {
 1 => Alice
 2 => Bob
 3 => Charlie
}
@enduml
```



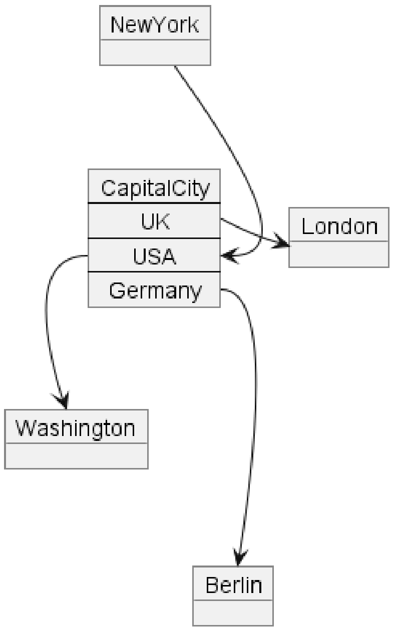
map: Map<Integer, String>	
1	Alice
2	Bob
3	Charlie

And add link with object.

```
@startuml
object London

map CapitalCity {
 UK *-> London
 USA => Washington
 Germany => Berlin
}
@enduml
```



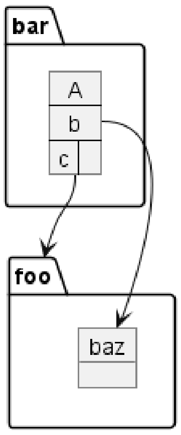
```
@startuml
object London
object Washington
object Berlin
object NewYork

map CapitalCity {
 UK *-> London
 USA *--> Washington
 Germany *---> Berlin
}

NewYork --> CapitalCity::USA
@enduml
```



[Ref. #307]

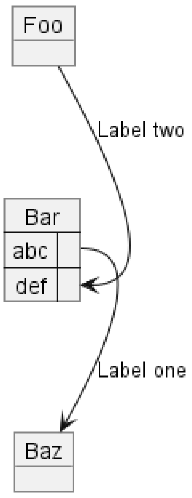
```
@startuml
package foo {
    object baz
}

package bar {
    map A {
        b *-> foo.baz
        c =>
    }
}
```

```
A::c --> foo
@enduml
```



[Ref. QA-12934]

```
@startuml
object Foo
map Bar {
  abc=>
  def=>
}
object Baz

Bar::abc --> Baz : Label one
Foo --> Bar::def : Label two
@enduml
```

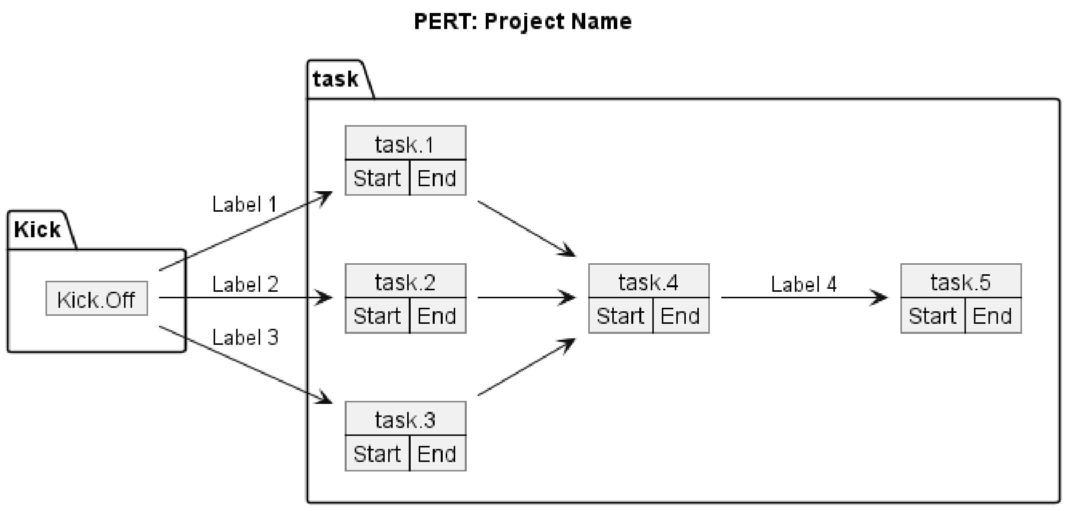


[Ref. #307]

4.7 程序（或项目）评估和审查技术（PERT）与地图

你可以使用 `map table`，以制作程序（或项目）评估和审查技术（PERT）图。

```
@startuml PERT
left to right direction
' Horizontal lines: -->, <--, <-->
' Vertical lines: ->, <-, <->
title PERT: Project Name

map Kick.Off {
}
map task.1 {
    Start => End
}
map task.2 {
    Start => End
}
map task.3 {
    Start => End
}
map task.4 {
    Start => End
}
map task.5 {
    Start => End
}
Kick.Off --> task.1 : Label 1
Kick.Off --> task.2 : Label 2
Kick.Off --> task.3 : Label 3
task.1 --> task.4
task.2 --> task.4
task.3 --> task.4
task.4 --> task.5 : Label 4
@enduml
```

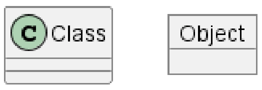


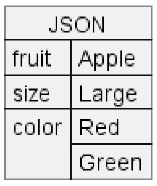
[Ref. QA-12337]

4.8 Display JSON Data on Class or Object diagram

4.8.1 Simple example

```
@startuml
class Class
object Object
json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```





JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.

5 活动图

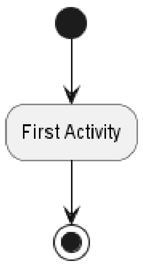
5.1 简单活动

使用 (*) 作为活动图的开始点和结束点。

有时，你可能想用 (*top) 强制开始点位于图示的顶端。

使用--> 绘制箭头。

```
@startuml

(*) --> "First Activity"
"First Activity" --> (*)

@enduml
```

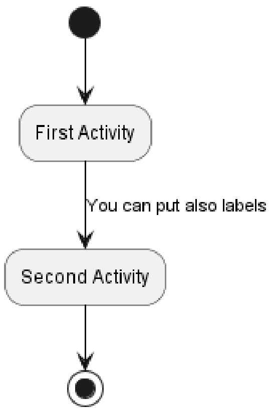


5.2 箭头上的标签

默认情况下，箭头开始于最接近的活动。

可以用 [和] 放在箭头定义的后面来添加标签。

```
@startuml

(*) --> "First Activity"
-->[You can put also labels] "Second Activity"
--> (*)

@enduml
```



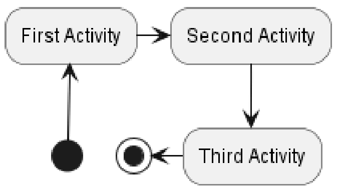
5.3 改变箭头方向

你可以使用-> 定义水平方向箭头，还可以使用下列语法强制指定箭头的方向：

- `-down->` (default arrow)
- `-right->` or `->`
- `-left->`

- -up->

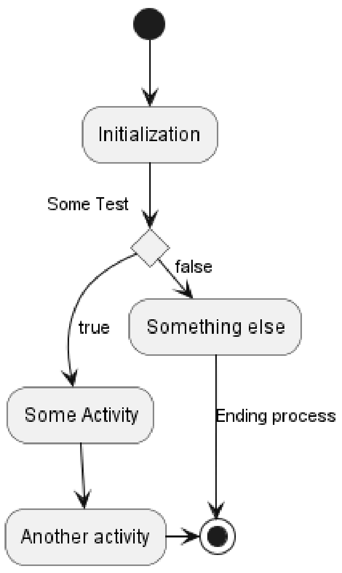
```
@startuml

(*) -up-> "First Activity"
-right-> "Second Activity"
--> "Third Activity"
-left-> (*)

@enduml
```



5.4 分支

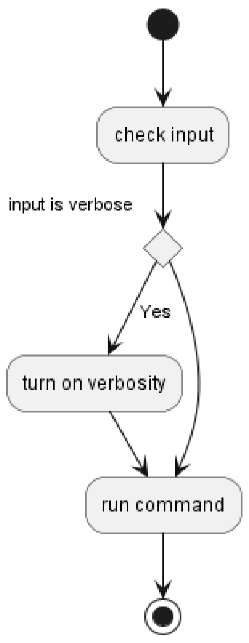
你可以使用关键字 `if/then/else` 创建分支。

```
@startuml
(*) --> "Initialization"

if "Some Test" then
  -->[true] "Some Activity"
  --> "Another activity"
  -right-> (*)
else
  ->[false] "Something else"
  -->[Ending process] (*)
endif

@enduml
```



不过，有时你可能需要重复定义同一个活动：

```
@startuml
(*)  --> "check input"
If "input is verbose" then
--> [Yes] "turn on verbosity"
--> "run command"
else
--> "run command"
Endif
-->(*)
@enduml
```



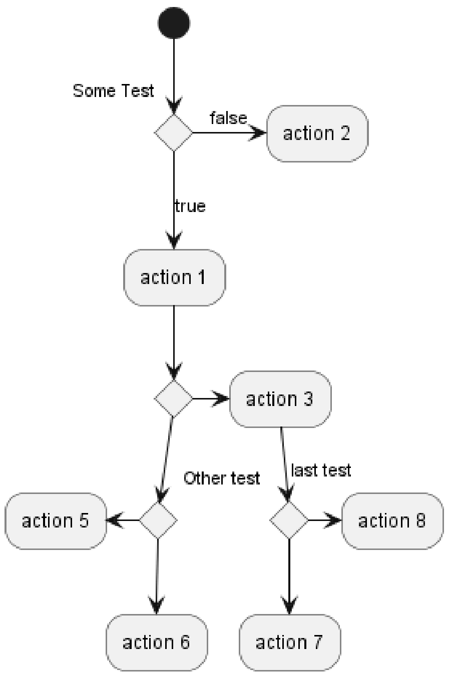
5.5 更多分支

默认情况下，一个分支连接上一个最新的活动，但是也可以使用 `if` 关键字进行连接。

还可以嵌套定义分支。

```
@startuml

(*) --> if "Some Test" then

  -->[true] "action 1"

  if "" then
    -> "action 3" as a3
  else
    if "Other test" then
      -left-> "action 5"
    else
      --> "action 6"
    endif
  endif

else

  ->[false] "action 2"

endif

a3 --> if "last test" then
  --> "action 7"
else
  -> "action 8"
```

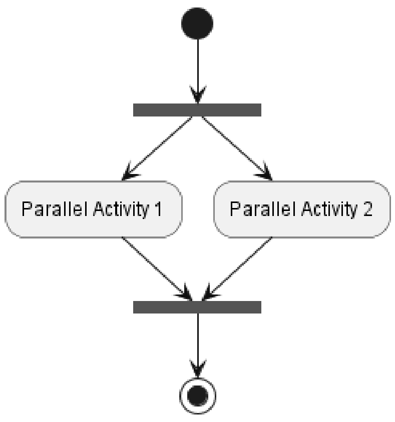
```
endif

@enduml
```



5.6 同步

你可以使用 `=== code ===` 来显示同步条。

```
@startuml

(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

===B1=== --> "Parallel Activity 2"
--> ===B2===

--> (*)

@enduml
```

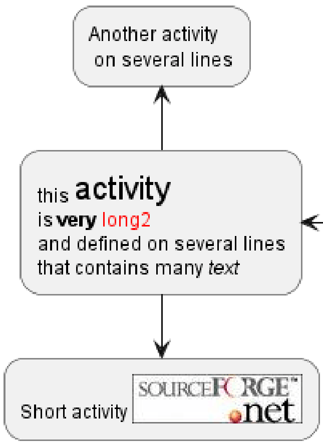


5.7 长的活动描述

定义活动时可以用来定义跨越多行的描述。

还可以用 as 关键字给活动起一个短的别名。这个别名可以在接下来的图示定义中使用。

```
@startuml
(*) -left-> "this <size:20>activity</size>
is <b>very</b> <color:red>long2</color>
and defined on several lines
that contains many <i>text</i>" as A1

-up-> "Another activity\n on several lines"

A1 --> "Short activity <img:sourceforge.jpg>"
@enduml
```



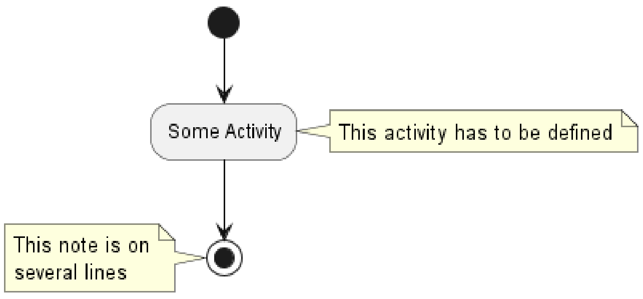
5.8 注释

你可以在活动定义之后用 note left, note right, note top or note bottom, 命令给活动添加注释。

如果想给开始点添加注释，只需把注释的定义放在活动图最开始的地方即可。

也可以用关键字 endnote 定义多行注释。

```
@startuml

(*) --> "Some Activity"
note right: This activity has to be defined
"Some Activity" --> (*)
note left
 This note is on
 several lines
end note

@enduml
```



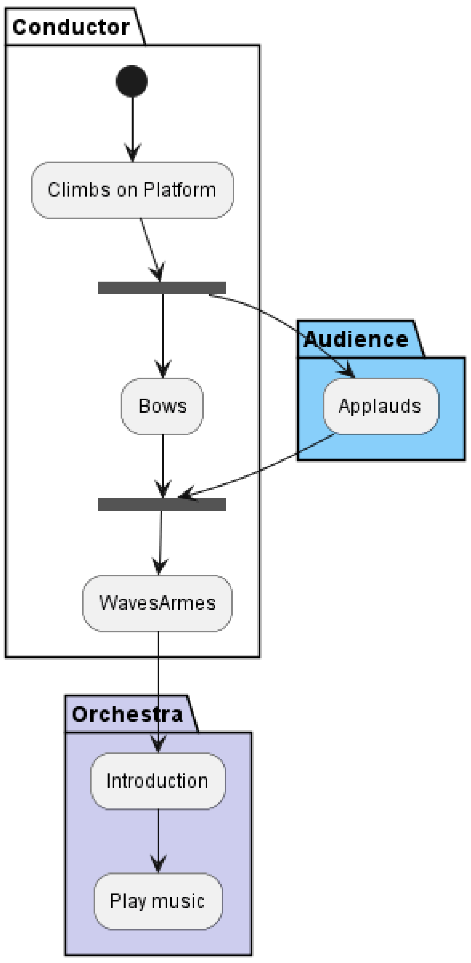
5.9 分区

用关键字 `partition` 定义分区，还可以设置背景色 (用颜色名或者颜色值)。

定义活动的时候，它自动被放置到最新的分区中。

用} 结束分区的定义。

```
@startuml

partition Conductor {
  (*) --> "Climbs on Platform"
  --> === S1 ===
  --> Bows
}

partition Audience #LightSkyBlue {
  === S1 === --> Applauds
}

partition Conductor {
  Bows --> === S2 ===
  --> WavesArmes
  Applauds --> === S2 ===
}

partition Orchestra #CCCCEE {
  WavesArmes --> Introduction
  --> "Play music"
}

@enduml
```



5.10 显示参数

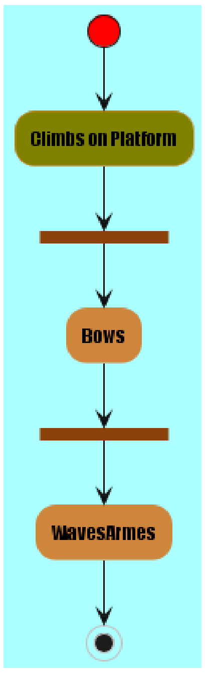
用 skinparam 命令修改字体和颜色。

如下场景可用：

- 在图示定义中
- 在引入的文件中
- 在命令行或 ANT 任务提供的配置文件中。

还可以为构造类型指定特殊颜色和字体。

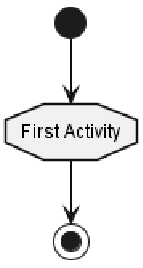
```
@startuml

skinparam backgroundColor #AAFFFF
skinparam activity {
  StartColor red
  BarColor SaddleBrown
  EndColor Silver
  BackgroundColor Peru
  BackgroundColor<< Begin >> Olive
  BorderColor Peru
  FontName Impact
}

(*) --> "Climbs on Platform" << Begin >>
--> === S1 ===
--> Bows
--> === S2 ===
--> WavesArmes
--> (*)

@enduml
```



5.11 八边形活动

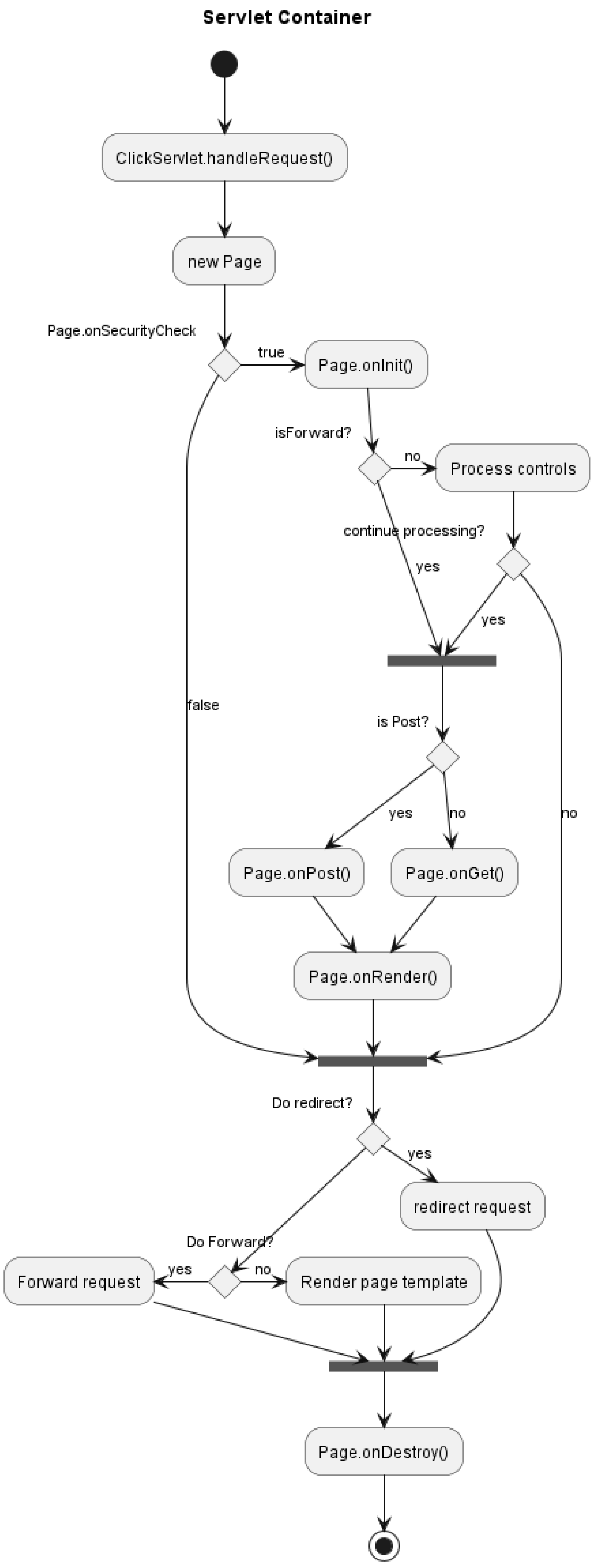
可用用 skinparam activityShape octagon 命令将活动的外形改为八边形。

```
@startuml
'Default is skinparam activityShape roundBox
skinparam activityShape octagon

(*) --> "First Activity"
"First Activity" --> (*)

@enduml
```



5.12 一个完整的例子

```
@startuml
title Servlet Container

(*) --> "ClickServlet.handleRequest()"
--> "new Page"

if "Page.onSecurityCheck" then
  ->[true] "Page.onInit()"
  if "isForward?" then
   ->[no] "Process controls"

   if "continue processing?" then
     -->[yes] ===RENDERING===
   else
     -->[no] ===REDIRECT_CHECK===
   endif

  else
   -->[yes] ===RENDERING===
  endif

  if "is Post?" then
    -->[yes] "Page.onPost()"
    --> "Page.onRender()" as render
    --> ===REDIRECT_CHECK===
  else
    -->[no] "Page.onGet()"
    --> render
  endif

else
  -->[false] ===REDIRECT_CHECK===
endif

if "Do redirect?" then
 ->[yes] "redirect request"
 --> ==BEFORE_DESTROY===
else
 if "Do Forward?" then
  -left->[yes] "Forward request"
  --> ==BEFORE_DESTROY===
 else
  -right->[no] "Render page template"
  --> ==BEFORE_DESTROY===
 endif
endif

--> "Page.onDestroy()"
-->(*)

@enduml
```


Servlet Container
ClickServlet.handleRequest()
new Page
Page.onSecurityCheck
true
Page.onInit()
isForward?
no
Process controls
continue processing?
yes
yes
false
is Post?
yes
no
no
Page.onPost()
Page.onGet()
Page.onRender()
Do redirect?
yes
redirect request
Do Forward?
yes
no
Forward request
Render page template
Page.onDestroy()

6 活动图（新语法）

以前用于活动图的语法存在一些局限性和可维护性问题。认识到这些弊端后，我们推出了全新的语法和实现方式，不仅用户友好，而且更加稳定。

6.0.1 新语法的优势

- 无需依赖 Graphviz：与序列图一样，新语法无需安装 Graphviz，从而简化了设置过程。
- 易于 * 维护：

6.0.2 过渡到新 * 语法

虽然我们将继续支持旧语法以保持兼容性，但我们强烈建议用户迁移到新语法，以充分利用它所提供的增强功能和优势。

现在就进行迁移，使用新的活动图语法体验更简化、更高效的图表制作流程。

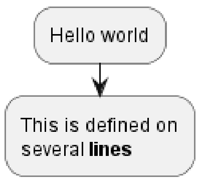
6.1 简单活动图

活动标签 (activity label) 以冒号开始，以分号结束。

文本格式支持 creole wiki 语法。

活动默认按照它们定义的顺序进行自动连接。

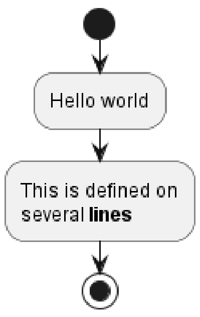
```
@startuml
:Hello world;
:This is defined on
several **lines**;
@enduml
```



6.2 开始/停止/结束

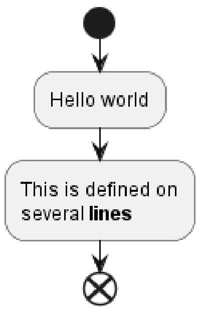
你可以使用 `start` 和 `stop` 关键字来表示一个图的开始和结束。

```
@startuml
start
:Hello world;
:This is defined on
several **lines**;
stop
@enduml
```



你也可以使用 `end` 关键字。

```
@startuml
start
:Hello world;
:This is defined on
several **lines**;
end
@enduml
```



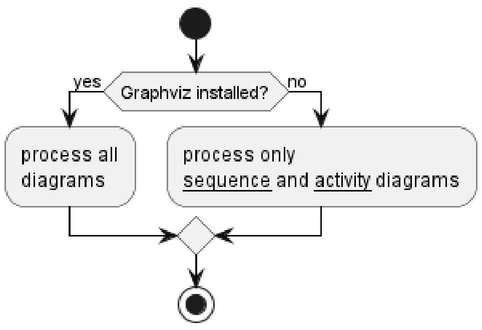
6.3 条件

你可以使用 `if`,`then`, `break` 和 `else` 关键词来在你的图表中放入测试。标签可以用圆括号提供。

有 3 种语法可供选择。

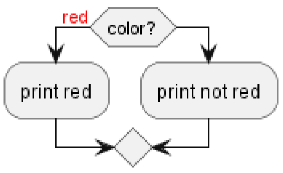
- `if (...) then (...)`

```
@startuml

start

if (Graphviz installed?) then (yes)
  :process all\ndiagrams;
else (no)
  :process only
  __sequence__ and __activity__ diagrams;
endif

stop

@enduml
```



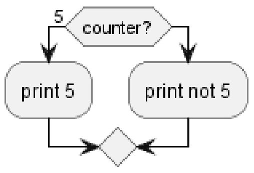
- `if (...) is (...) then`

```
@startuml
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
@enduml
```



- `if (...) equals (...) then`

```
@startuml
if (counter?) equals (5) then
:print 5;
else
:print not 5;
@enduml
```

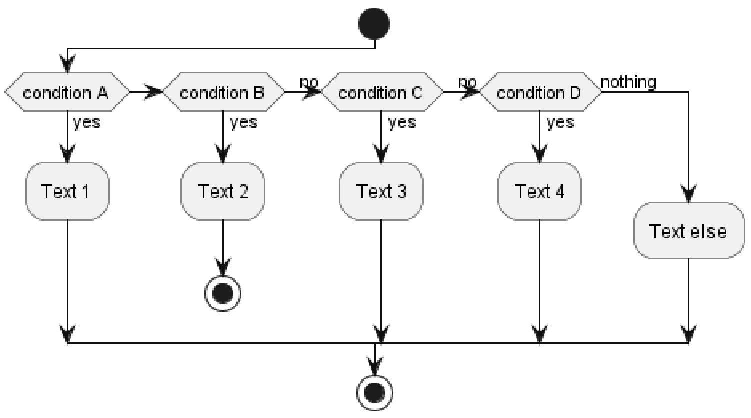


[Ref. QA-301]

6.3.1 几个测试（水平模式）

你可以使用 `elseif` 关键字来拥有几个测试（默认是水平模式）。

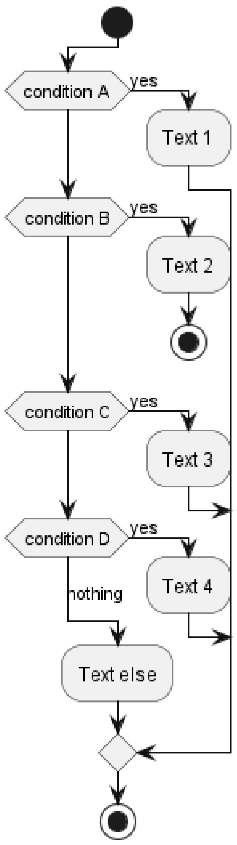
```
@startuml
start
if (condition A) then (yes)
  :Text 1;
elseif (condition B) then (yes)
  :Text 2;
  stop
(no) elseif (condition C) then (yes)
  :Text 3;
(no) elseif (condition D) then (yes)
  :Text 4;
else (nothing)
  :Text else;
endif
stop
@enduml
```



6.3.2 几个测试（垂直模式）

你可以使用`!pragma useVerticalIf on` 命令，让测试处于垂直模式。

```
@startuml
!pragma useVerticalIf on
start
if (condition A) then (yes)
  :Text 1;
elseif (condition B) then (yes)
  :Text 2;
  stop
elseif (condition C) then (yes)
  :Text 3;
elseif (condition D) then (yes)
  :Text 4;
else (nothing)
  :Text else;
endif
stop
@enduml
```



你可以使用`-P` command-line[命令行] 选项来指定 pragma。

```
java -jar plantuml.jar -PuseVerticalIf=on
```

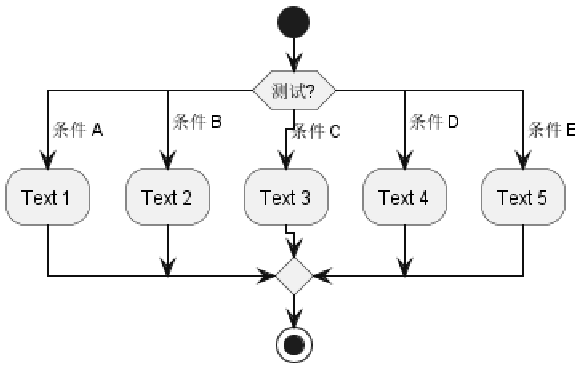
[参考文献：QA-3931,issue-582] 。

6.4 Switch 判断 [switch, case, endswitch]

你可以使用 `switch`, `case` 和 `endswitch` 关键词在图表中绘制 Switch 判断.

使用括号表示标注.

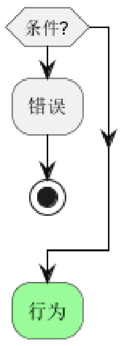
```
@startuml
start
switch (测试?)
case ( 条件 A )
  :Text 1;
case ( 条件 B )
  :Text 2;
case ( 条件 C )
  :Text 3;
case ( 条件 D )
  :Text 4;
case ( 条件 E )
  :Text 5;
endswitch
stop
@enduml
```



6.5 条件判断和终止 [kill, detach]

你可以在 if 判断中终止一个行为.

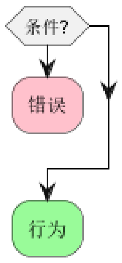
```
@startuml
if (条件?) then
  :错误;
  stop
endif
#palegreen:行为;
@enduml
```



但如果你想在特定行为上停止，你可以使用 `kill` 或 `detach` 关键字:

- `kill`

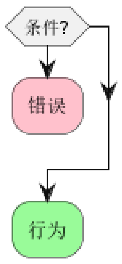
```
@startuml
if (条件?) then
  #pink:错误;
  kill
endif
#palegreen:行为;
@enduml
```



[参考. QA-265]

- detach

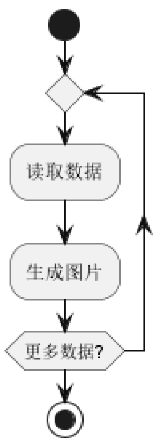
```
@startuml
if (条件?) then
  #pink:错误;
  detach
endif
#palegreen:行为;
@enduml
```



6.6 重复循环

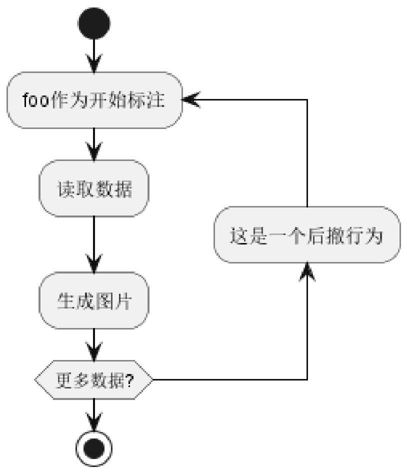
你可以使用关键字 `repeat` 和 `repeatwhile` 进行重复循环。

```
@startuml

start

repeat
  :读取数据;
  :生成图片;
repeat while (更多数据?)

stop

@enduml
```



你同样可以使用一个全局行为作为 repeat 目标，在返回循环开始时使用 backward 关键字插入一个全局行为。

```
@startuml

start

repeat :foo作为开始标注;
  :读取数据;
  :生成图片;
backward:这是一个后撤行为;
repeat while (更多数据?)

stop

@enduml
```

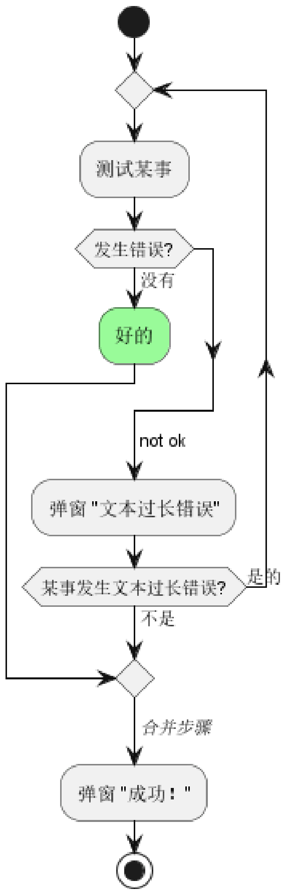


[Ref. QA-5826]

6.7 打断循环 [break]

你可以使用 break 关键字跟在循环中的某个行为后面打断循环.

```
@startuml
start
repeat
  :测试某事;
    if (发生错误?) then (没有)
      #palegreen:好的;
      break
    endif
    ->not ok;
    :弹窗 "文本过长错误";
repeat while (某事发生文本过长错误?) is (是的) not (不是)
->//合并步骤//;
:弹窗 "成功！";
stop
@enduml
```



[参考：QA-6105]

6.8 Goto 和标签处理 [label, goto]

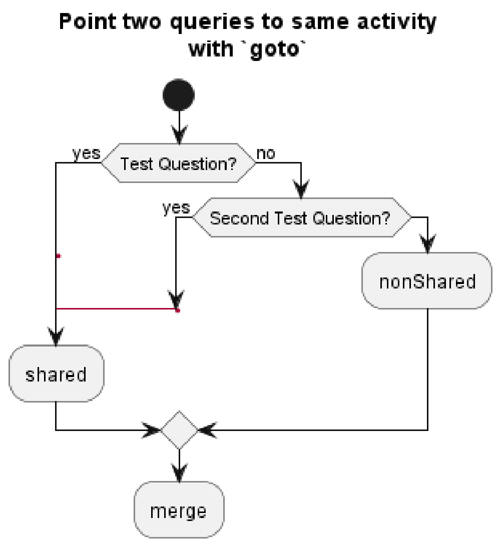
目前只是实验性的

你可以使用 `label` 和 `goto` 关键词来表示 Goto 处理，其中：

- `label <label_name>`
- `goto <label_name>`

```
@startuml
title Point two queries to same activity\nwith `goto`
start
if (Test Question?) then (yes)
'space label only for alignment
label sp_lab0
label sp_lab1
'real label
label lab
:shared;
```

```
else (no)
if (Second Test Question?) then (yes)
label sp_lab2
goto sp_lab1
else
:nonShared;
endif
endif
:merge;
@enduml
```

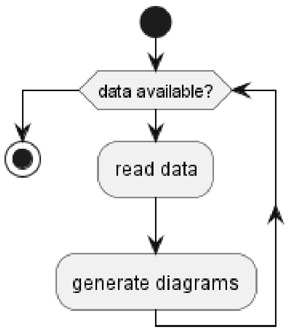


[Ref.QA-15026,QA-12526and initiallyQA-1626] 。

6.9 while 循环

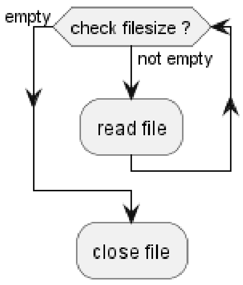
可以使用关键字 `while` 和 `end while` 进行 while 循环。

```
@startuml

start

while (data available?)
  :read data;
  :generate diagrams;
endwhile

stop

@enduml
```



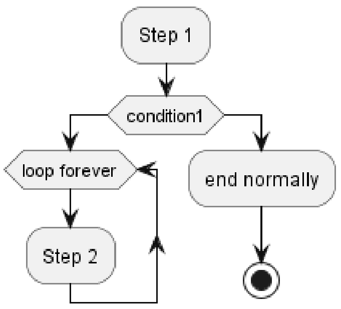
还可以在关键字 endwhile 后添加标注，还有一种方式是使用关键字 is。

```
@startuml
while (check filesize ?) is (not empty)
  :read file;
endwhile (empty)
:close file;
@enduml
```



如果你使用 +detach+ 来形成一个无限循环, 那么你可能需要使用 +-[hidden]->+ 来隐藏一些不完整的箭头。

```
@startuml
:Step 1;
if (condition1) then
  while (loop forever)
   :Step 2;
  endwhile
  -[hidden]->
  detach
else
  :end normally;
  stop
endif
@enduml
```

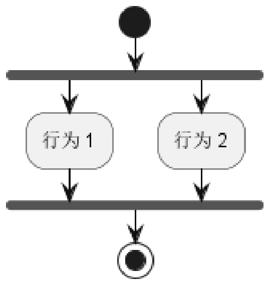


6.10 并行处理 [fork, fork again, end fork, end merge]

你可以使用 `fork`，`fork again` 和 `end fork` 或者 `end merge` 等关键字表示并行处理。

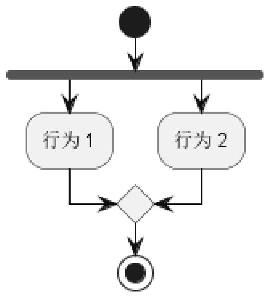
6.10.1 `fork` 示例

```
@startuml
start
fork
  :行为 1;
fork again
  :行为 2;
end fork
stop
@enduml
```



6.10.2 `fork` 和合并示例

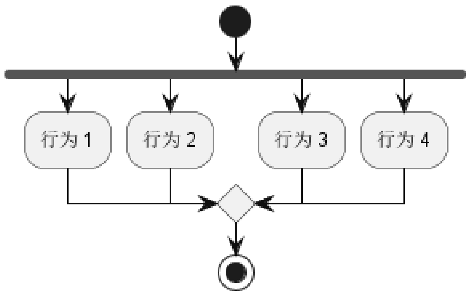
```
@startuml
start
fork
  :行为 1;
fork again
  :行为 2;
end merge
stop
@enduml
```



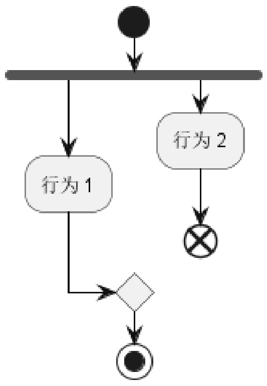
[参考：QA-5320]

```
@startuml
start
fork
  :行为 1;
fork again
  :行为 2;
fork again
  :行为 3;
fork again
  :行为 4;
```

```
end merge
stop
@enduml
```



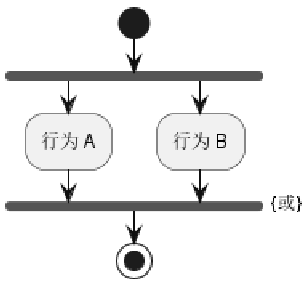
```
@startuml
start
fork
  :行为 1;
fork again
  :行为 2;
  end
end merge
stop
@enduml
```



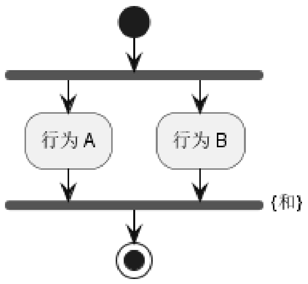
[参考：QA-13731]

6.10.3 `end fork` 标注 (或 UML 连接规范):

```
@startuml
start
fork
  :行为 A;
fork again
  :行为 B;
end fork {或}
stop
@enduml
```



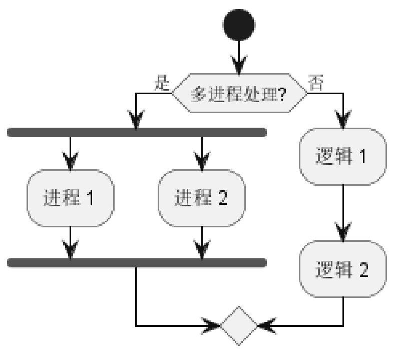
```
@startuml
start
fork
  :行为 A;
fork again
  :行为 B;
end fork {和}
stop
@enduml
```



[参考：QA-5346]

6.10.4 其他示例

```
@startuml

start

if (多进程处理?) then (是)
  fork
    :进程 1;
  fork again
    :进程 2;
  end fork
else (否)
  :逻辑 1;
  :逻辑 2;
endif

@enduml
```

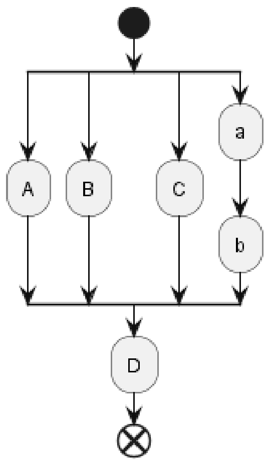


6.11 分割处理

6.11.1 分割

你可以使用 `split`, `split again` 和 `end split` 关键字去表达分割处理

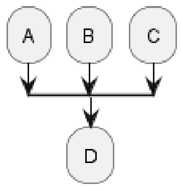
```
@startuml
start
split
   :A;
split again
   :B;
split again
   :C;
split again
   :a;
   :b;
end split
:D;
end
@enduml
```



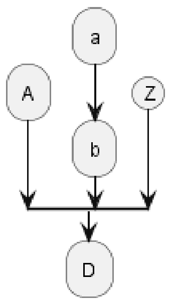
6.11.2 输入分割 (多个入口)

你可以使用包含 `hidden` 指令的箭头去制造一个输入分割 (多入口):

```
@startuml
split
   -[hidden]->
   :A;
split again
   -[hidden]->
   :B;
split again
   -[hidden]->
   :C;
end split
:D;
@enduml
```



```
@startuml
split
   -[hidden]->
   :A;
split again
   -[hidden]->
   :a;
   :b;
split again
   -[hidden]->
   (Z)
end split
:D;
@enduml
```



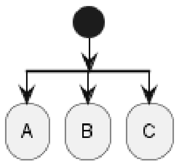
[参考：QA-8662]

6.11.3 输出分割 (多个结束点)

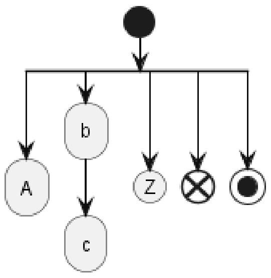
你可以使用 `kill` 或 `detach` 去制造一个输出分割 (多个结束点):

```
@startuml
start
split
   :A;
   kill
split again
   :B;
   detach
split again
   :C;
   kill
```

```
end split
@enduml
```



```
@startuml
start
split
   :A;
   kill
split again
   :b;
   :c;
   detach
split again
   (Z)
   detach
split again
   end
split again
   stop
end split
@enduml
```

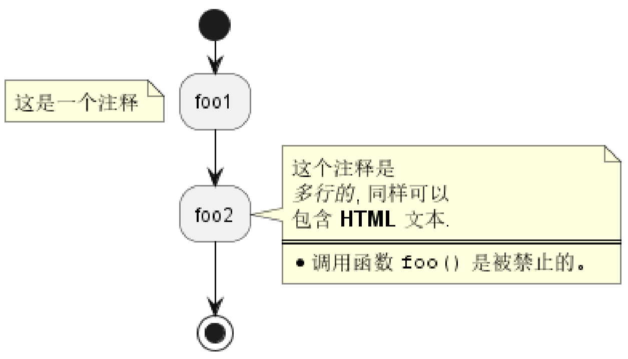


6.12 注释

文本格式支持 creole wiki 语法。

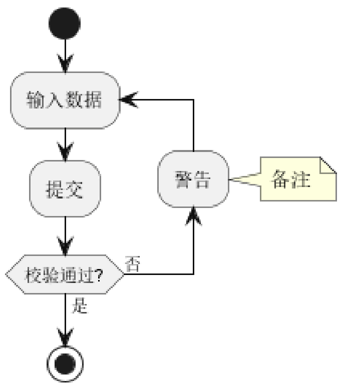
一个注释可以是浮动的，使用 `floating` 关键词。

```
@startuml

start
:foo1;
floating note left: 这是一个注释
:foo2;
note right
  这个注释是
  //多行的//, 同样可以
  包含 <b>HTML</b> 文本.
  ====
  * 调用函数 ""foo()"" 是被禁止的。
end note
stop

@enduml
```



您可以添加返回行为的注释：

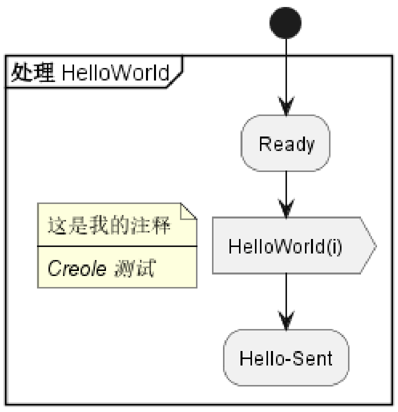
```
@startuml
start
repeat :输入数据;
:提交;
backward :警告;
note right: 备注
repeat while (校验通过?) is (否) not (是)
stop
@enduml
```



[参考：QA-11788]

您可以添加分区活动的注释：

```
@startuml
start
partition "**处理** HelloWorld" {
    note
        这是我的注释
        ----
        //Creole 测试//
    end note
    :Ready;
    :HelloWorld(i)>
    :Hello-Sent;
}
@enduml
```

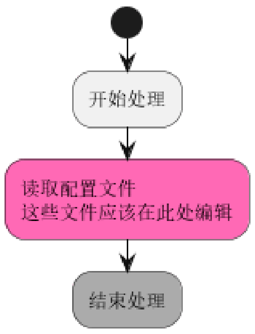


[参考：QA-2398]

6.13 改变颜色

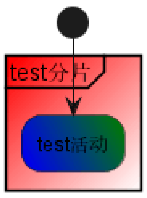
你可以为一些活动指定颜色

```
@startuml

start
:开始处理;
#HotPink:读取配置文件
这些文件应该在此处编辑;
#AAAAAA:结束处理;

@enduml
```



你通用可以使用渐变色.

```
@startuml
start
partition #red/white test分片 {
        #blue\green:test活动;
}
@enduml
```

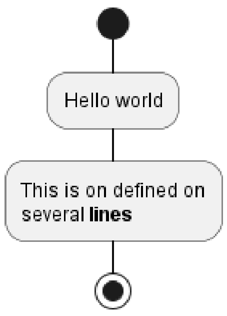


[参考：QA-4906]

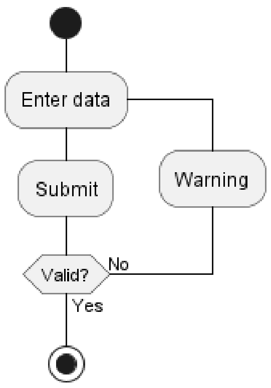
6.14 无箭头连接线

您可以使用 `skinparam ArrowHeadColor none` 参数来表示仅使用线条连接活动，而不带箭头。

```
@startuml
skinparam ArrowHeadColor none
start
:Hello world;
:This is on defined on
several **lines**;
stop
@enduml
```



```
@startuml
skinparam ArrowHeadColor none
start
repeat :Enter data;
:Submit;
backward :Warning;
repeat while (Valid?) is (No) not (Yes)
stop
@enduml
```

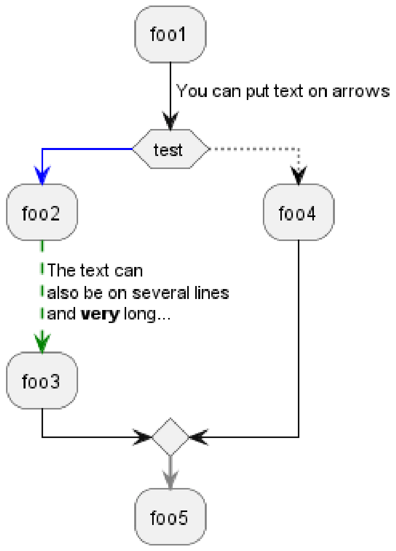


6.15 箭头

使用-> 标记，你可以给箭头添加文字或者修改箭头颜色。

同时，你也可以选择点状 (dotted)，条状 (dashed)，加粗或者是隐式箭头

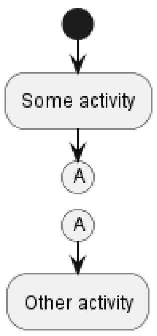
```
@startuml
:foo1;
-> You can put text on arrows;
if (test) then
  -[#blue]->
  :foo2;
  -[#green,dashed]-> The text can
  also be on several lines
  and **very** long...;
  :foo3;
else
  -[#black,dotted]->
  :foo4;
endif
-[#gray,bold]->
:foo5;
@enduml
```



6.16 连接器 (Connector)

你可以使用括号定义连接器。

```
@startuml
start
:Some activity;
(A)
detach
(A)
:Other activity;
@enduml
```

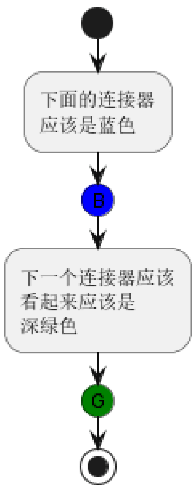


6.17 连接器颜色

你可以在连接器上增加颜色

```
@startuml
start
```

```
:下面的连接器
应该是蓝色;
#blue:(B)
:下一个连接器应该
看起来应该是
深绿色;
#green:(G)
stop
@enduml
```

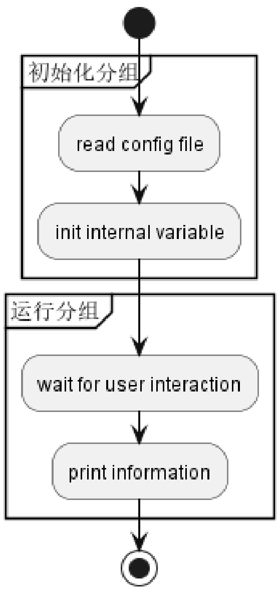


[参考. QA-10077]

6.18 组合 (grouping)

通过定义分组 (group)，你可以把多个活动分组。

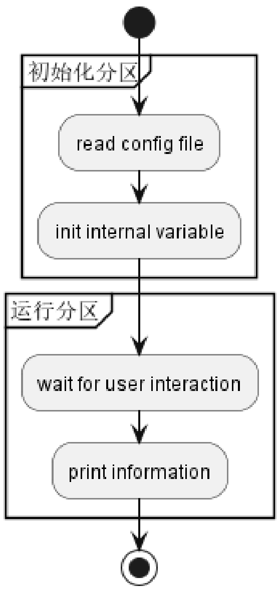
```
@startuml
start
group 初始化分组
    :read config file;
    :init internal variable;
end group
group 运行分组
    :wait for user interaction;
    :print information;
end group

stop
@enduml
```



6.18.1 分区

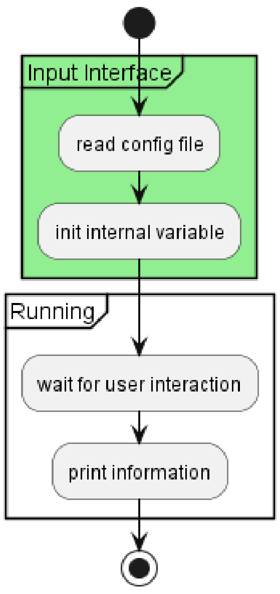
通过定义分区 (partition)，你可以把多个活动组合 (group) 在一起:

```
@startuml
start
partition 初始化分区 {
    :read config file;
    :init internal variable;
}
partition 运行分区 {
    :wait for user interaction;
    :print information;
}

stop
@enduml
```



这里同样可以改变分区颜色 color:

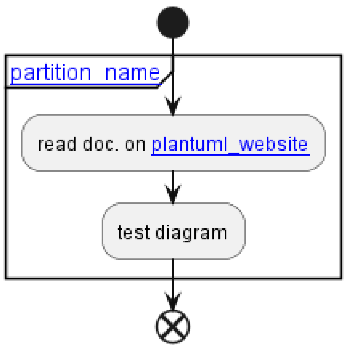
```
@startuml
start
partition #lightGreen "Input Interface" {
    :read config file;
    :init internal variable;
}
partition Running {
    :wait for user interaction;
    :print information;
}
stop
@enduml
```



[参考：QA-2793]

同样可以添加一个链接到分区：

```
@startuml
start
partition "[[http://plantuml.com partition_name]]" {
    :read doc. on [[http://plantuml.com plantuml_website]];
    :test diagram;
}
end
@enduml
```



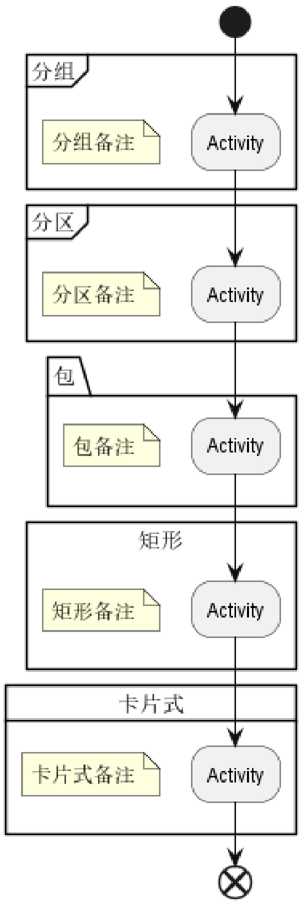
[参考：QA-542]

6.18.2 分组, 分区, 包, 矩形或卡片式

你可以分组活动通过定义:

- group;
- partition;
- package;
- rectangle;
- card.

```
@startuml
start
group 分组
  :Activity;
end group
floating note: 分组备注

partition 分区 {
  :Activity;
}
floating note: 分区备注

package 包 {
  :Activity;
}
floating note: 包备注

rectangle 矩形 {
  :Activity;
}
floating note: 矩形备注

card 卡片式 {
  :Activity;
}
floating note: 卡片式备注
end
@enduml
```

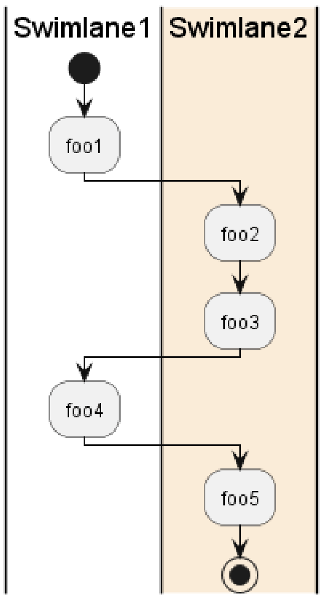


6.19 泳道 (Swimlanes)

你可以使用管道符 | 来定义泳道。

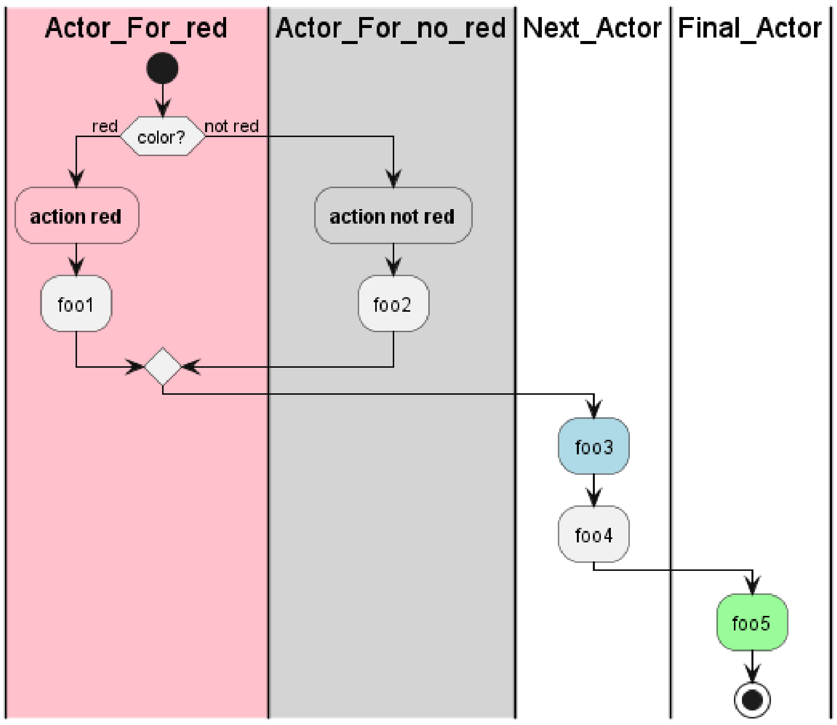
还可以改变泳道的颜色。

```
@startuml
|Swimlane1|
start
:foo1;
|#AntiqueWhite|Swimlane2|
:foo2;
:foo3;
|Swimlane1|
:foo4;
|Swimlane2|
:foo5;
stop
@enduml
```



你可以在泳道中增加 if 判断或 repeat 或 while 循环.

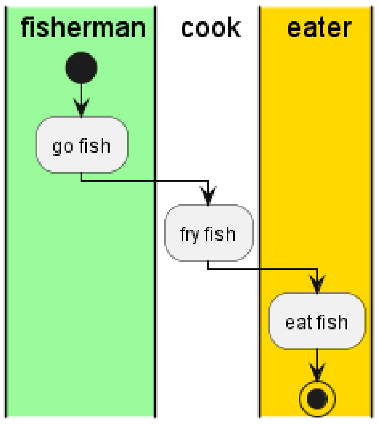
```
@startuml
|#pink|Actor_For_red|
start
if (color?) is (red) then
#pink:**action red**;
:foo1;
else (not red)
|#lightgray|Actor_For_no_red|
#lightgray:**action not red**;
:foo2;
endif
|Next_Actor|
#lightblue:foo3;
:foo4;
|Final_Actor|
#palegreen:foo5;
stop
@enduml
```



你同样可以在泳道中增加别名，使用 `alias` 语法:

- `|[#<color>|]<swimlane_alias>| <swimlane_title>`

```
@startuml
|#palegreen|f| fisherman
|c| cook
|#gold|e| eater
|f|
start
:go fish;
|c|
:fry fish;
|e|
:eat fish;
stop
@enduml
```



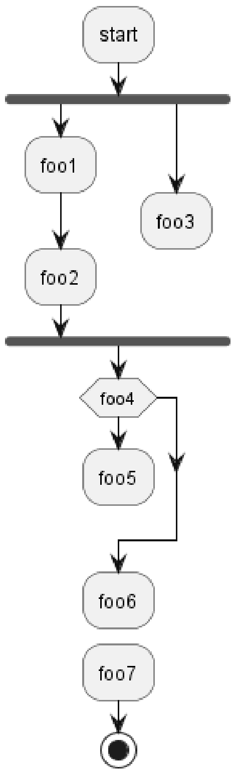
[参考：QA-2681]

6.20 分离 (detach)

可以使用关键字 detach 或 kill 移除箭头。

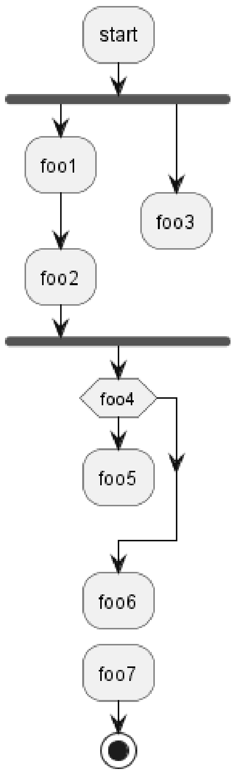
- detach

```
@startuml
 :start;
 fork
   :foo1;
   :foo2;
 fork again
   :foo3;
   detach
 endfork
 if (foo4) then
   :foo5;
   detach
 endif
 :foo6;
 detach
 :foo7;
 stop
@enduml
```



- kill

```
@startuml
 :start;
 fork
   :foo1;
   :foo2;
 fork again
   :foo3;
   kill
 endfork
 if (foo4) then
   :foo5;
   kill
 endif
 :foo6;
 kill
 :foo7;
 stop
@enduml
```



6.21 SDL（规范和描述语言）

6.21.1 SDL 形状名称表

名称	旧语法	定型语法
输入	<	<<input>>
输出	>	<<output>>
程序	|	<<procedure>>
加载	\	<<load>>
保存	/	<<save>>
连续	}	<<continuous>>
任务	]	<<task>>

[Ref.QA-11518,GH-1270]

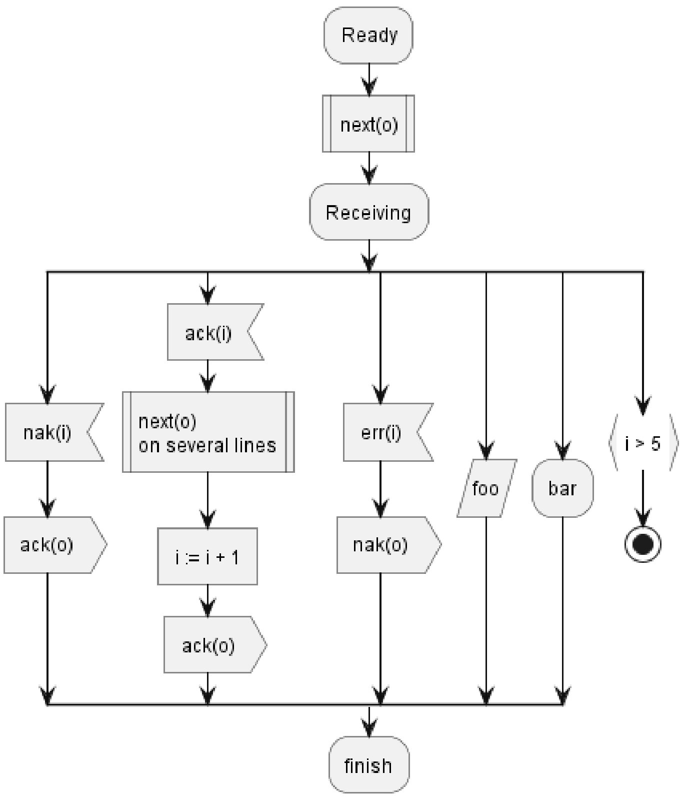
6.21.2 SDL using final separator (Deprecated form)

通过更改最终；separator，可以为活动设置不同的渲染：

- |
- <
- >

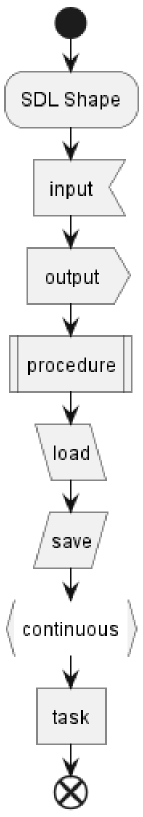
- /
- \\
-]
- }

```
@startuml
:Ready;
:next(o)|
:Receiving;
split
 :nak(i)<
 :ack(o)>
split again
 :ack(i)<
 :next(o)
 on several lines|
 :i := i + 1]
 :ack(o)>
split again
 :err(i)<
 :nak(o)>
split again
 :foo/
split again
 :bar\\
split again
 :i > 5}
stop
end split
:finish;
@enduml
```



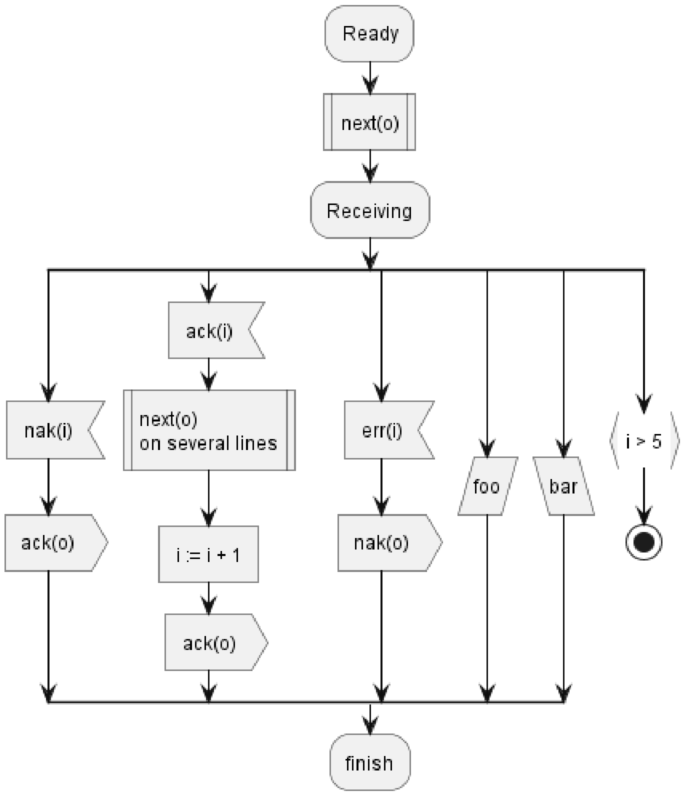
6.21.3 使用正态分隔符和立体原型的 SDL（当前正式形式）

```
@startuml
start
:SDL Shape;
:input; <<input>>
:output; <<output>>
:procedure; <<procedure>>
:load; <<load>>
:save; <<save>>
:continuous; <<continuous>>
:task; <<task>>
end
@enduml
```



```
@startuml
:Ready;
:next(o); <<procedure>>
:Receiving;
split
 :nak(i); <<input>>
 :ack(o); <<output>>
split again
 :ack(i); <<input>>
 :next(o)
 on several lines; <<procedure>>
 :i := i + 1; <<task>>
 :ack(o); <<output>>
split again
 :err(i); <<input>>
 :nak(o); <<output>>
split again
 :foo; <<save>>
```

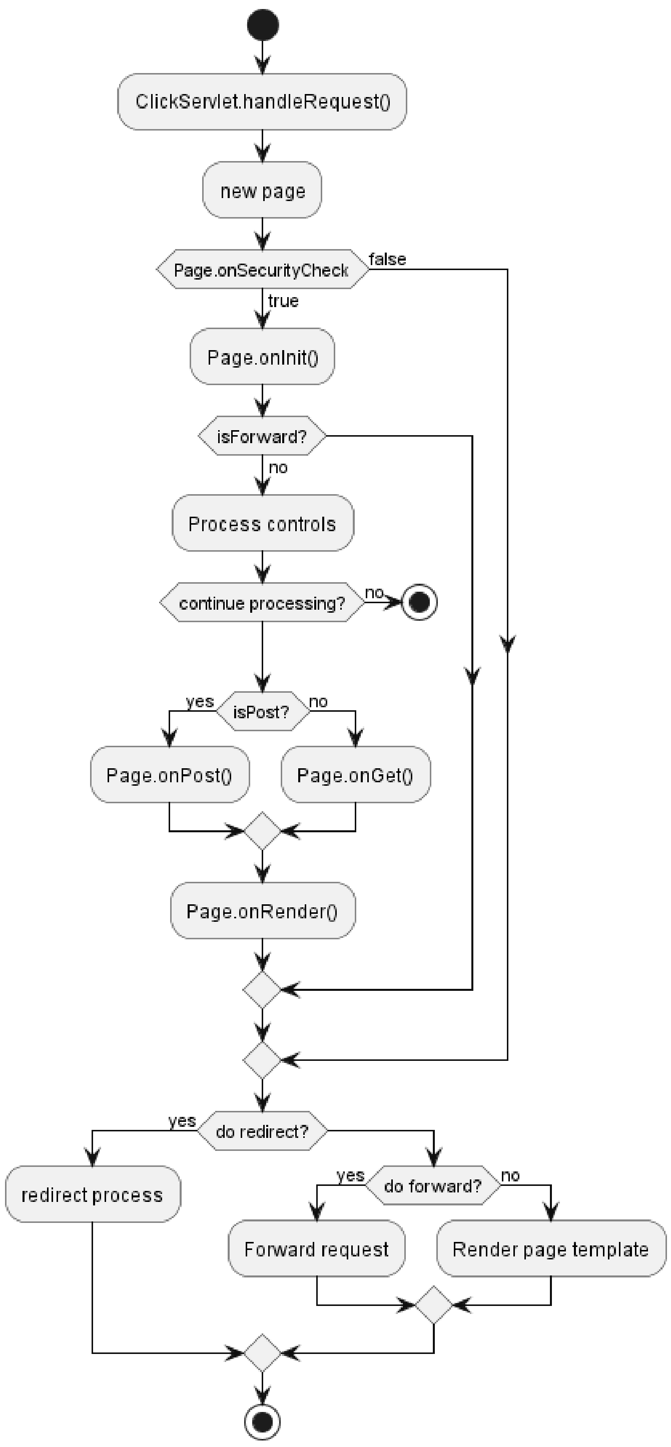
```
split again
 :bar; <<load>>
split again
 :i > 5; <<continuous>>
stop
end split
:finish;
@enduml
```



6.22 一个完整的例子

```
@startuml

start
:ClickServlet.handleRequest();
:new page;
if (Page.onSecurityCheck) then (true)
  :Page.onInit();
  if (isForward?) then (no)
    :Process controls;
    if (continue processing?) then (no)
      stop
    endif

    if (isPost?) then (yes)
      :Page.onPost();
    else (no)
      :Page.onGet();
    endif
    :Page.onRender();
  endif
else (false)
```

```
endif

if (do redirect?) then (yes)
  :redirect process;
else
  if (do forward?) then (yes)
    :Forward request;
  else (no)
    :Render page template;
  endif
endif

stop

@enduml
```

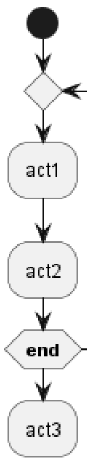


6.23 判断的样式

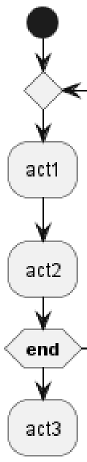
6.23.1 inside 样式 (默认)

```
@startuml
skinparam conditionStyle inside
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
```

```
@enduml
```

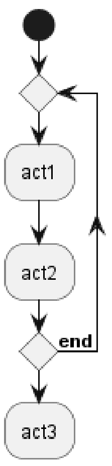


```
@startuml
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```



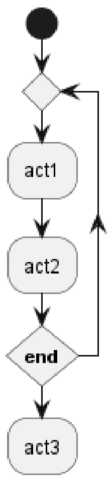
6.23.2 Diamond 样式

```
@startuml
skinparam conditionStyle diamond
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```

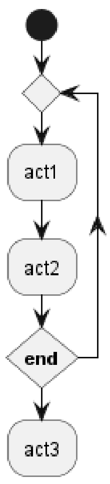


6.23.3 InsideDiamond (或 *Foo1*) 样式

```
@startuml
skinparam conditionStyle InsideDiamond
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```



```
@startuml
skinparam conditionStyle foo1
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```



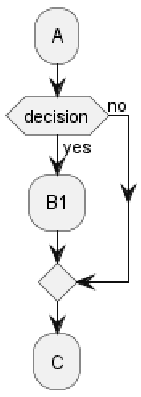
[参考：QA-1290 and #400]

6.24 判断的结束样式

6.24.1 Diamond 样式 (默认)

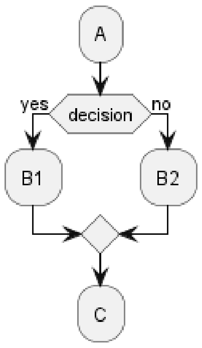
- With one branch

```
@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
    :B1;
else (no)
endif
:C;
@enduml
```



- 两个分支 (B1, B2)

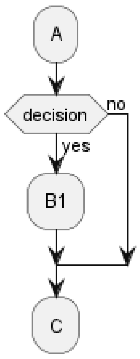
```
@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
    :B1;
else (no)
    :B2;
endif
:C;
@enduml
@enduml
```



6.24.2 水平线 (hline) 样式

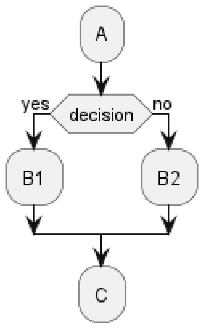
- 一个分

```
@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
    :B1;
else (no)
endif
:C;
@enduml
```



- 两个分支 (B1, B2)

```
@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
    :B1;
else (no)
    :B2;
endif
:C;
@enduml
@enduml
```

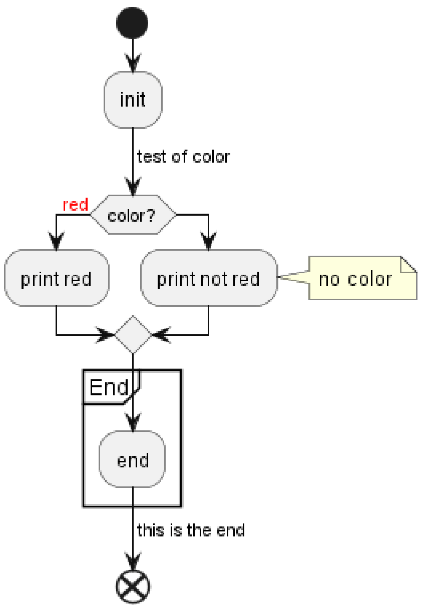


[Ref. QA-4015]

6.25 使用 sytle 定义 (全局) 样式

6.25.1 无样式 *(*默认*)*

```
@startuml
start
:init;
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
```

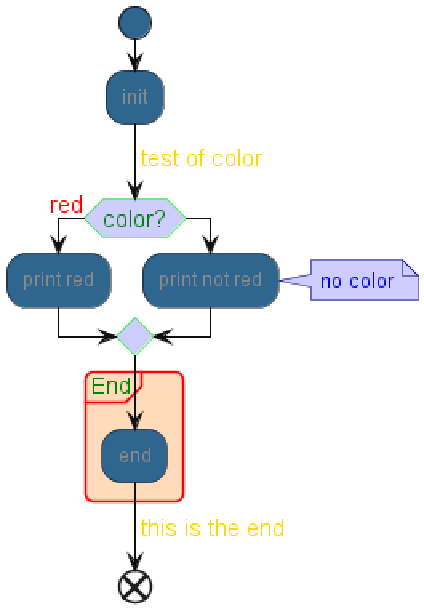


6.25.2 有样式

你可以使用 style 节点去定义样式然后改变渲染。

```
@startuml
```

```
<style>
activityDiagram {
  BackgroundColor #33668E
  BorderColor #33668E
  FontColor #888
  FontName arial

  diamond {
    BackgroundColor #ccf
    LineColor #00FF00
    FontColor green
    FontName arial
    FontSize 15
  }
  arrow {
    FontColor gold
    FontName arial
    FontSize 15
  }
  partition {
    LineColor red
    FontColor green
    RoundCorner 10
    BackgroundColor PeachPuff
  }
  note {
    FontColor Blue
    LineColor Navy
    BackgroundColor #ccf
  }
}
document {
   BackgroundColor transparent
}
</style>
start
:init;
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
```


init
test of color
red
color?
print red
print not red
no color
End
end
this is the end

7 组件图

我们来看几个例子。

Advantages of PlantUML:

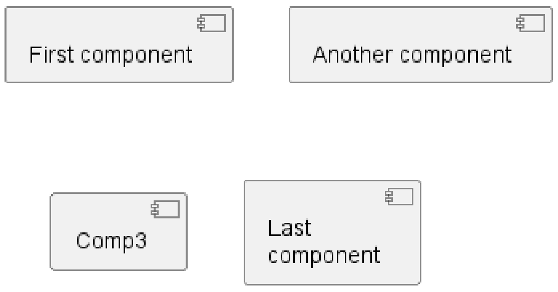
- **Simplicity**: With PlantUML, you can create component diagrams using simple and intuitive text-based descriptions, eliminating the need for complex drawing tools.
- **Integration**: PlantUML seamlessly integrates with various tools and platforms, making it a versatile choice for developers and architects.
- **Collaboration**: The PlantUML forum offers a platform for users to discuss, share, and seek assistance on their diagrams, fostering a collaborative community.

7.1 组件

组件必须用中括号括起来。

还可以使用关键字 `component` 定义一个组件。并且可以用关键字 `as` 给组件定义一个别名。这个别名可以在稍后定义关系的时候使用。

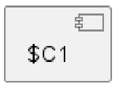
```
@startuml

[First component]
[Another component] as Comp2
component Comp3
component [Last\ncomponent] as Comp4

@enduml
```



7.1.1 命名例外

注意，以 `$` 开头的组件名以后不能隐藏或删除，因为 `hide` 和 `remove` 命令会将该名称视为 `$tag`，而不是组件名。要删除此类组件，必须为其添加别名或标记。

```
@startuml
component [$C1]
component [$C2] $C2
component [$C2] as dollarC2
remove $C1
remove $C2
remove dollarC2
@enduml
```



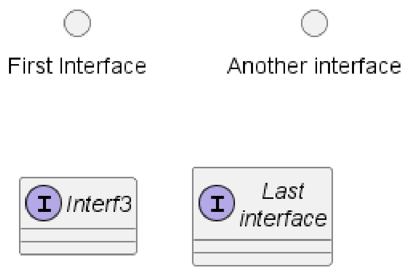
7.2 接口

接口可以使用 `()` 来定义 (因为这个看起来像个圆)。

还可以使用关键字 `interface` 关键字来定义接口。并且还可以使用关键字 `as` 定义一个别名。这个别名可以在稍后定义关系的时候使用。

我们稍后可以看到，接口的定义是可选的。

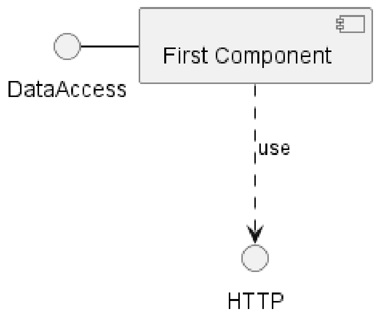
```
@startuml

() "First Interface"
() "Another interface" as Interf2
interface Interf3
interface "Last\ninterface" as Interf4

@enduml
```



7.3 基础的示例

元素之间可以使用虚线 (..)、直线 (--)、箭头 (-->) 进行连接。

```
@startuml

DataAccess - [First Component]
[First Component] ..> HTTP : use

@enduml
```

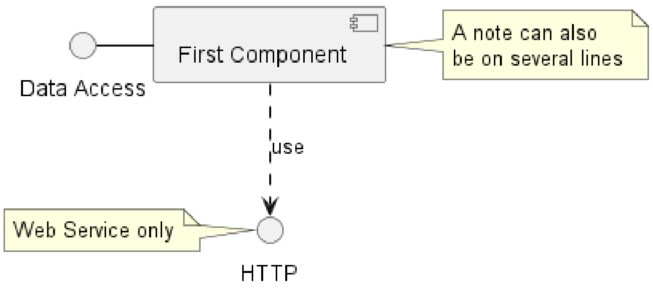


7.4 使用注释

你可以使用 `note left of` , `note right of` , `note top of` , `note bottom of` 等关键字定义相对于对象位置的注释。

也可以使用关键字 `note` 单独定义注释，然后使用虚线 (..) 将其连接到其他对象。

```
@startuml

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

note left of HTTP : Web Service only
note right of [First Component]
  A note can also
  be on several lines
end note

@enduml
```



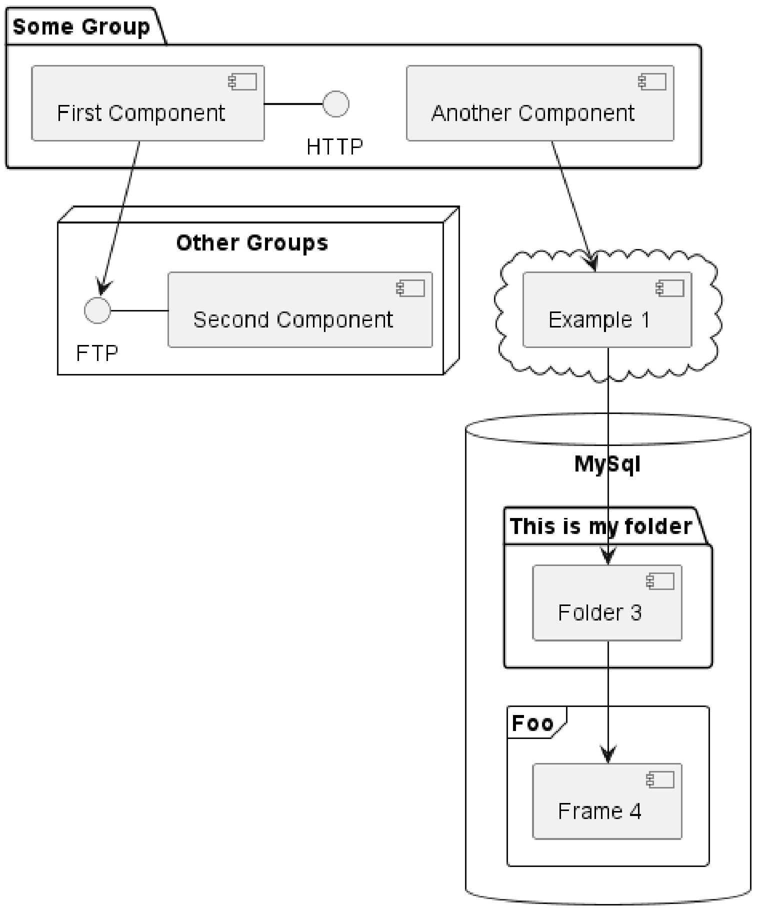
7.5 组合组件

你可以使用多个关键字将组件和接口组合在一起。

- package
- node
- folder
- frame
- cloud
- database

```
@startuml

package "Some Group" {
  HTTP - [First Component]
  [Another Component]
}

node "Other Groups" {
  FTP - [Second Component]
  [First Component] --> FTP
}

cloud {
  [Example 1]
}


database "MySql" {
  folder "This is my folder" {
    [Folder 3]
  }
  frame "Foo" {
    [Frame 4]
  }
}


[Another Component] --> [Example 1]
[Example 1] --> [Folder 3]
```

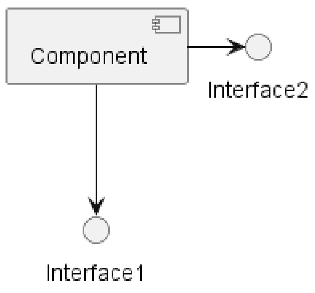
```
[Folder 3] --> [Frame 4]

@enduml
```



7.6 改变箭头方向

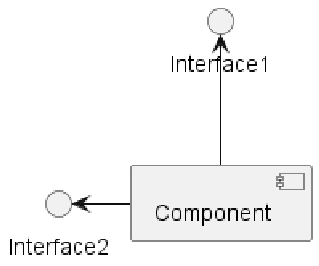
默认情况下，对象之间用--连接，并且连接是竖直的。不过可以使用一个横线或者点设置水平方向的连接，就行这样：

```
@startuml
[Component] --> Interface1
[Component] -> Interface2
@enduml
```



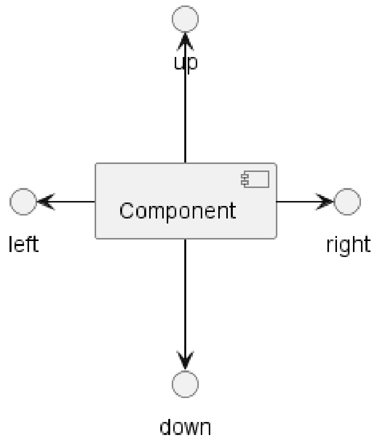
也可以使用反向连接：

```
@startuml
Interface1 <-- [Component]
Interface2 <- [Component]
@enduml
```



还可以使用关键字 `left`, `right`, `up` or `down` 改变箭头方向。

```
@startuml
[Component] -left-> left
[Component] -right-> right
[Component] -up-> up
[Component] -down-> down
@enduml
```



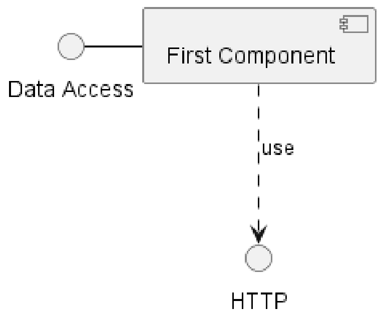
允许使用方向单词的首字母或者前两个字母表示方向 (例如`-d-`, `-do-`, `-down-`都是等价的)。

请不要乱用这些功能: *Graphviz*(PlantUML 的后端引擎) 不喜欢这个样子。

7.7 使用 UML2 标记

By default *(from v1.2020.13-14)*, UML2 notation is used.

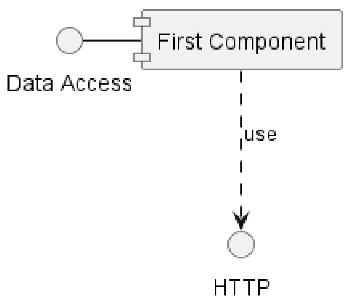
```
@startuml

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

@enduml
```



7.8 使用 UML1 标记符

命令 `skinparam componentStyle uml1` 可以切换到 UML1 标记符。

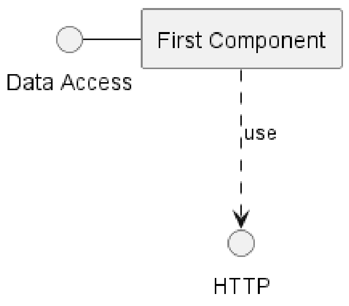
```
@startuml
skinparam componentStyle uml1

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

@enduml
```



7.9 Use rectangle notation (remove UML notation)

The `skinparam componentStyle rectangle` command is used to switch to rectangle notation *(without any UML notation)*.

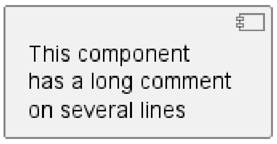
```
@startuml
skinparam componentStyle rectangle

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

@enduml
```



7.10 长描述

可以用方括号”[ ]” 在连线上添加描述。

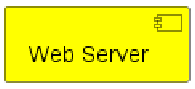
```
@startuml
component comp1 [
This component
has a long comment
on several lines
]
@enduml
```



7.11 不同的颜色表示

你可以在声明一个组件时加上颜色的声明。

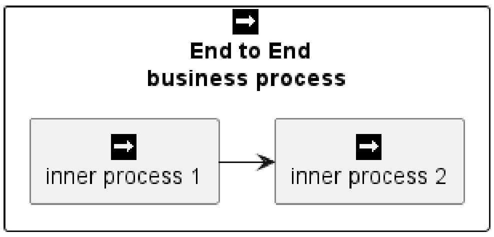
```
@startuml
component  [Web Server] #Yellow
@enduml
```



7.12 在定型组件中使用精灵图

你可以在定型组件中使用精灵图（sprite)。

```
@startuml
sprite $businessProcess [16x16/16] {
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFF0FFFFF
FFFFFFFFFF00FFFF
FF00000000000FFF
FF000000000000FF
FF00000000000FFF
FFFFFFFFFF00FFFF
FFFFFFFFFF0FFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
}


rectangle " End to End\nbusiness process" <<$businessProcess>> {
 rectangle "inner process 1" <<$businessProcess>> as src
 rectangle "inner process 2" <<$businessProcess>> as tgt
 src -> tgt
}
@enduml
```



7.13 显示参数

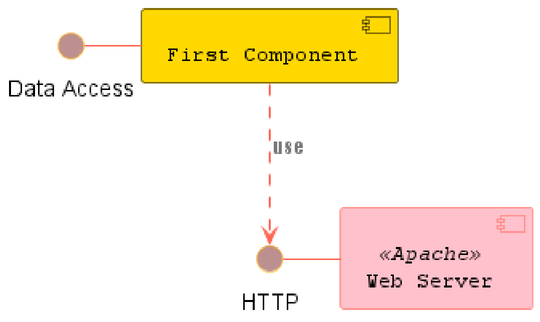
用 skinparam 改变字体和颜色。

可以在如下场景中使用：

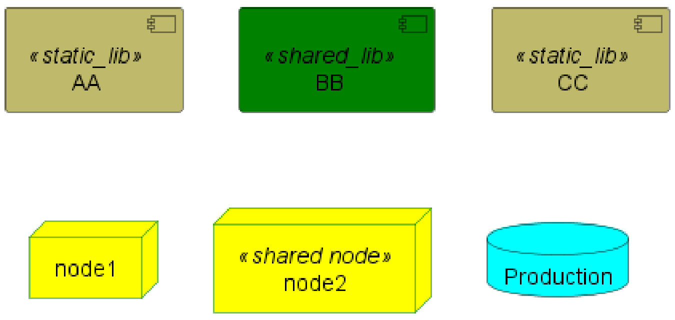
- 在图示的定义中，
- 在引入的文件中，
- 在命令行或者 ANT 任务提供的配置文件中。

可以为构造类型和接口定义特殊的颜色和字体。

```
@startuml

skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

skinparam component {
  FontSize 13
  BackgroundColor<<Apache>> Pink
  BorderColor<<Apache>> #FF6655
  FontName Courier
  BorderColor black
  BackgroundColor gold
  ArrowFontName Impact
  ArrowColor #FF6655
  ArrowFontColor #777777
}

() "Data Access" as DA
Component "Web Server" as WS << Apache >>

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - WS

@enduml
```



```
@startuml

skinparam component {
    backgroundColor<<static_lib>> DarkKhaki
    backgroundColor<<shared_lib>> Green
}

skinparam node {
borderColor Green
backgroundColor Yellow
backgroundColor<<shared_node>> Magenta
}
skinparam databaseBackgroundColor Aqua

[AA] <<static_lib>>
[BB] <<shared_lib>>
[CC] <<static_lib>>

node node1
node node2 <<shared node>>
database Production

@enduml
```

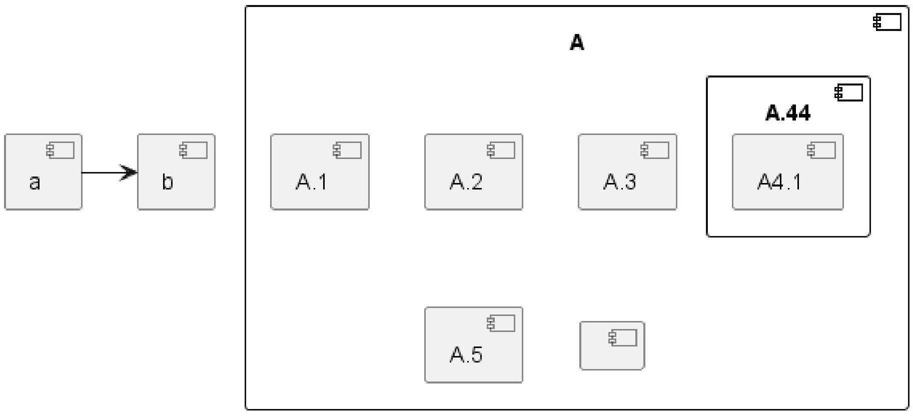


7.14 Specific SkinParameter

7.14.1 componentStyle

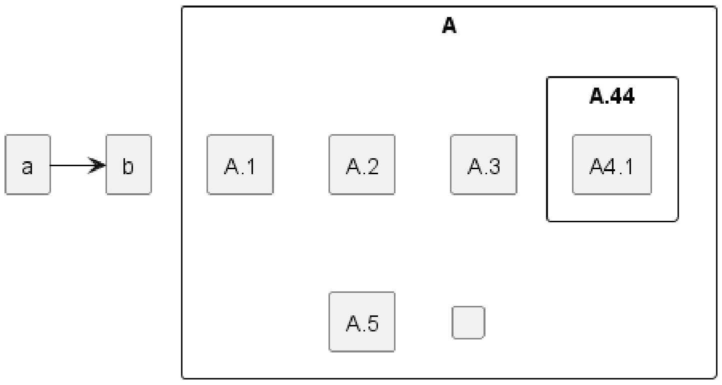
- By default (or with `skinparam componentStyle uml2`), you have an icon for component

```
@startuml
skinparam BackgroundColor transparent
skinparam componentStyle uml2
component A {
   component "A.1" {
}
   component A.44 {
      [A4.1]
}
   component "A.2"
   [A.3]
   component A.5 [
A.5]
   component A.6 [
]
}
[a]->[b]
@enduml
```



- If you want to suppress it, and to have only the rectangle, you can use `skinparam componentStyle rectangle`

```
@startuml
skinparam BackgroundColor transparent
skinparam componentStyle rectangle
component A {
   component "A.1" {
}
   component A.44 {
      [A4.1]
}
   component "A.2"
   [A.3]
   component A.5 [
A.5]
   component A.6 [
]
}
[a]->[b]
@enduml
```



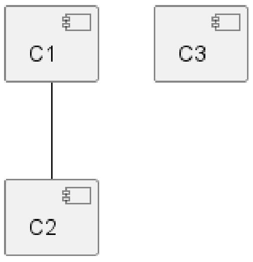
[Ref. 10798]

7.15 隐藏或删除未链接的组件

默认情况下，所有的组件都显示出来：

```
@startuml
component C1
component C2
component C3
C1 -- C2
```

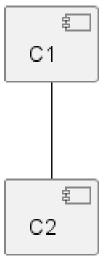
```
@enduml
```



但你可以：

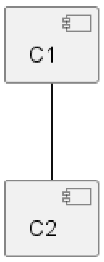
- `hide @unlinked` 组件：

```
@startuml
component C1
component C2
component C3
C1 -- C2

hide @unlinked
@enduml
```



- 或 `remove @unlinked` 组件：

```
@startuml
component C1
component C2
component C3
C1 -- C2

remove @unlinked
@enduml
```



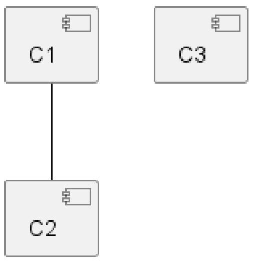
[参考 QA-11052]

7.16 Hide, Remove or Restore tagged component or wildcard

You can put `$tags` (using `$`) on components, then remove, hide or restore components either individually or by tags.

By default, all components are displayed:

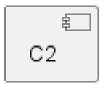
```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2
@enduml
```



But you can:

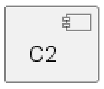
- `hide $tag13` components:

```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2

hide $tag13
@enduml
```



- or `remove $tag13` components:

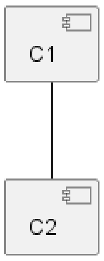
```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2

remove $tag13
@enduml
```



- or `remove $tag13 and restore $tag1` components:

```
@startuml
component C1 $tag13 $tag1
component C2
component C3 $tag13
C1 -- C2

remove $tag13
```

```
restore $tag1
@enduml
```



- or `remove * and restore $tag1` components:

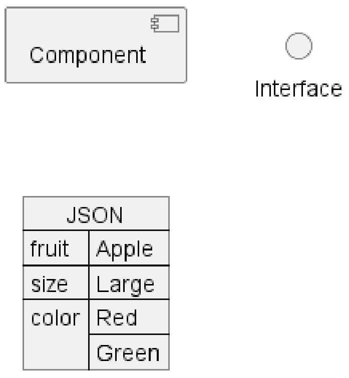
```
@startuml
component C1 $tag13 $tag1
component C2
component C3 $tag13
C1 -- C2

remove *
restore $tag1
@enduml
```

[Ref. QA-7337 and QA-11052]

7.17 在组件图上显示 JSON 数据

7.17.1 简单的例子

```
@startuml
allowmixing

component Component
()        Interface

json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



[Ref.QA-15481]

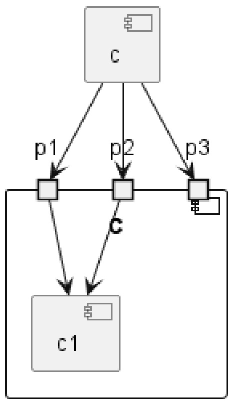
关于另一个例子，请看 [JSON 页面](json#2fyxla9p9ob6l3t3tjre)。

7.18 端口 [port, portIn, portOut]

您可以用 `port,portin` 和 `portout` 关键词添加端口。

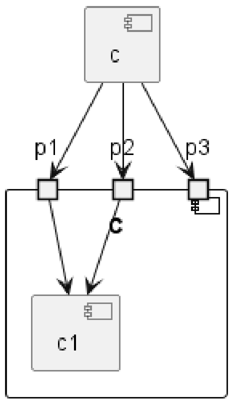
7.18.1 端口

```
@startuml
[c]
component C {
  port p1
  port p2
  port p3
  component c1
}

c --> p1
c --> p2
c --> p3
p1 --> c1
p2 --> c1
@enduml
```



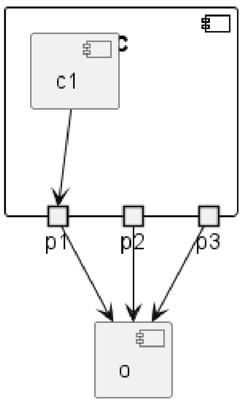
7.18.2 端口输入

```
@startuml
[c]
component C {
  portin p1
  portin p2
  portin p3
  component c1
}

c --> p1
c --> p2
c --> p3
p1 --> c1
p2 --> c1
@enduml
```



7.18.3 端口输出

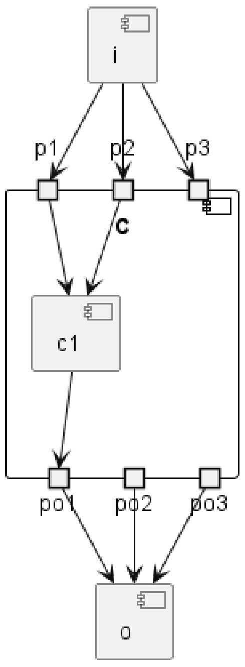
```
@startuml
component C {
  portout p1
  portout p2
  portout p3
  component c1
}
[o]
p1 --> o
p2 --> o
p3 --> o
c1 --> p1
@enduml
```



7.18.4 混合使用 **PortIn** 和 **PortOut**

```
@startuml
[i]
component C {
  portin p1
  portin p2
  portin p3
  portout po1
  portout po2
  portout po3
  component c1
}
[o]

i --> p1
```

```
i --> p2
i --> p3
p1 --> c1
p2 --> c1
po1 --> o
po2 --> o
po3 --> o
c1 --> po1
@enduml
```

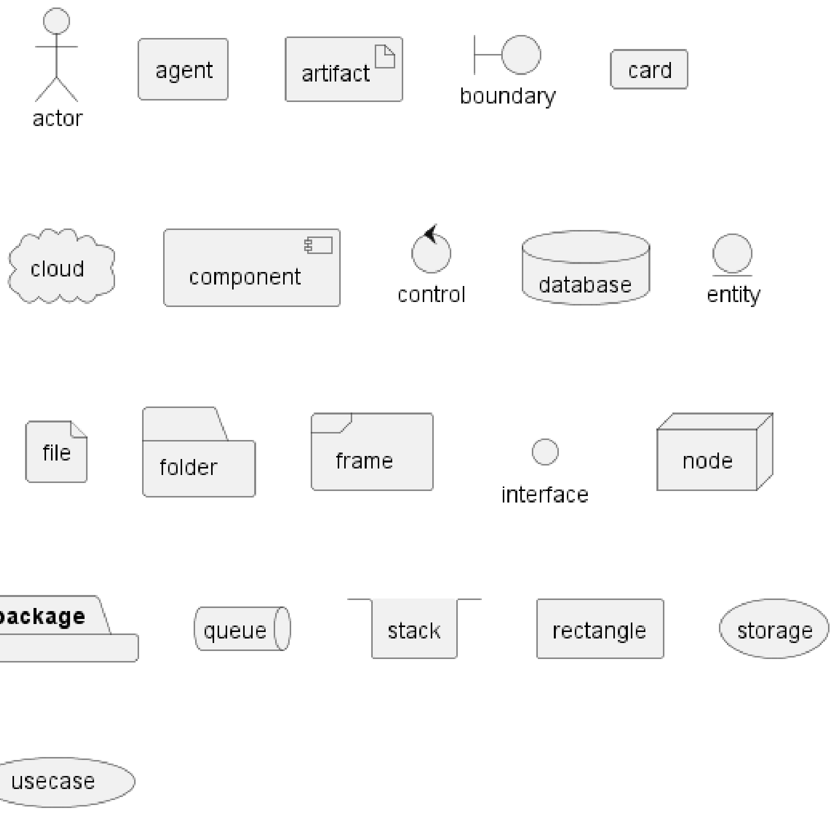


8 部署图

With [PlantUML](https:*plantuml.com/en/deployment-diagram), creating deployment diagrams becomes a breeze. The platform offers a simple and intuitive way to design these diagrams using plain text, ensuring rapid iterations and easy version control. Moreover, the [PlantUML forum](https:*forum.plantuml.net/) provides a vibrant community where users can seek help, share ideas, and collaborate on diagramming challenges. One of the key advantages of PlantUML is its ability to integrate seamlessly with various tools and platforms, making it a preferred choice for professionals and enthusiasts alike.

8.1 声明元素

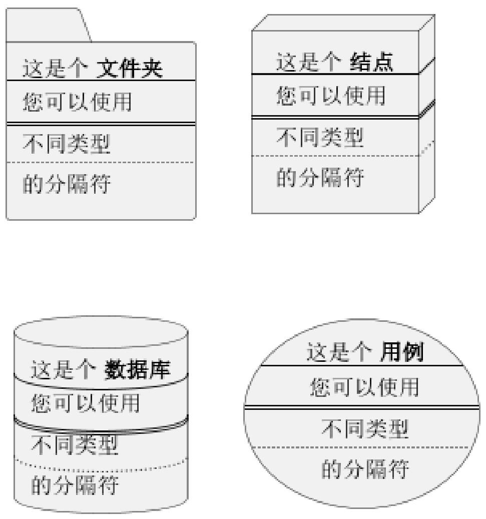
```
@startuml
actor actor
agent agent
artifact artifact
boundary boundary
card card
cloud cloud
component component
control control
database database
entity entity
file file
folder folder
frame frame
interface  interface
node node
package package
queue queue
stack stack
rectangle rectangle
storage storage
usecase usecase
@enduml
```



可选的，您可以使用方括号 [] 放置长描述文本。

```
@startuml
folder folder [
这是个 <b>文件夹
----
您可以使用
====
不同类型
....
的分隔符
]

node node [
这是个 <b>结点
----
您可以使用
====
不同类型
....
的分隔符
]

database database [
这是个 <b>数据库
----
您可以使用
====
不同类型
....
的分隔符
]
```

```
usecase usecase [
这是个 <b>用例
----
您可以使用
====
不同类型
....
的分隔符
]

@enduml
```



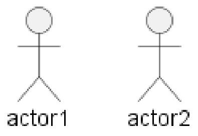
8.2 声明元素（使用简短的形式）

我们可以使用一些简短的形式声明元素。

长格式关键词	短格式关键词	长表示例	短格式的例子	参考文献。
`actor`	: *a* :	`actor actor1`	`:actor2:`	Actors
`component`	[*c*]	`component component1`	`[component2]`	组件
`interface`	() *i*	`interface interface1`	`() "interface2"`	接口
`usecase`	(*u*)	`usecase usecase1`	`(usecase2)`	用例

8.2.1 Actor

```
@startuml

actor actor1
:actor2:

@enduml
```

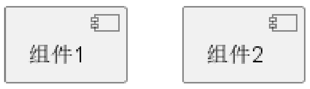


注意：有一种旧的语法是用 *guillemet* 来表示 *actor* 的，现在已经过时了，过几天就会删除。请不要在你的图表中使用。

8.2.2 组件

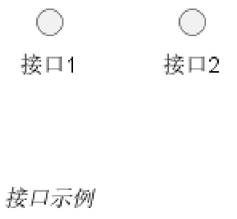
```
@startuml
```

```
component 组件1
[组件2]

@enduml
```



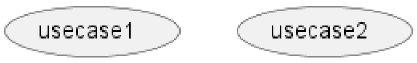
8.2.3 接口

```
@startuml

interface 接口1
() "接口2"

label "//接口示例//"
@enduml
```



8.2.4 Usecase

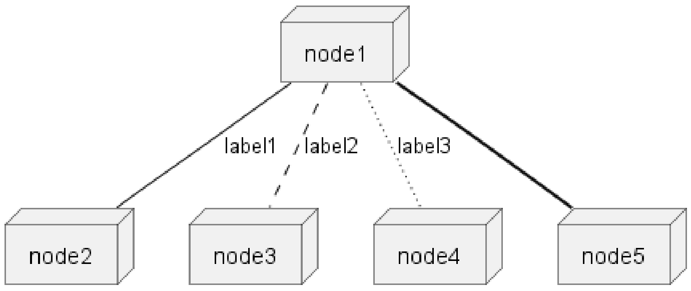
```
@startuml

usecase usecase1
(usecase2)

@enduml
```



8.3 链接或箭头

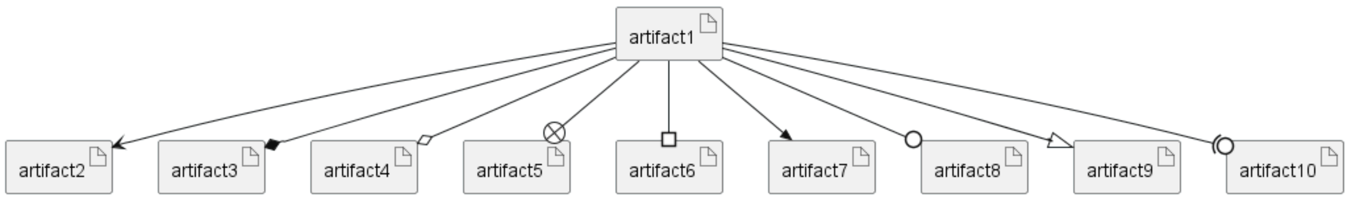
你可以在有或没有标签的元素之间创建简单的链接。

```
@startuml

node node1
node node2
node node3
node node4
node node5
node1 -- node2 : label1
node1 .. node3 : label2
node1 ~~ node4 : label3
node1 == node5

@enduml
```



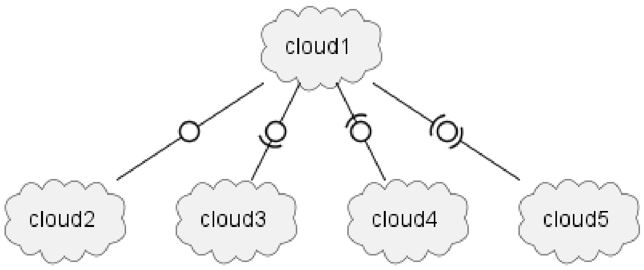
可以使用几种类型的链接。

```
@startuml

artifact artifact1
artifact artifact2
artifact artifact3
artifact artifact4
artifact artifact5
artifact artifact6
artifact artifact7
artifact artifact8
artifact artifact9
artifact artifact10
artifact1 --> artifact2
artifact1 --* artifact3
artifact1 --o artifact4
artifact1 --+ artifact5
artifact1 --# artifact6
artifact1 -->> artifact7
artifact1 --0 artifact8
artifact1 --^ artifact9
artifact1 --(0 artifact10

@enduml
```



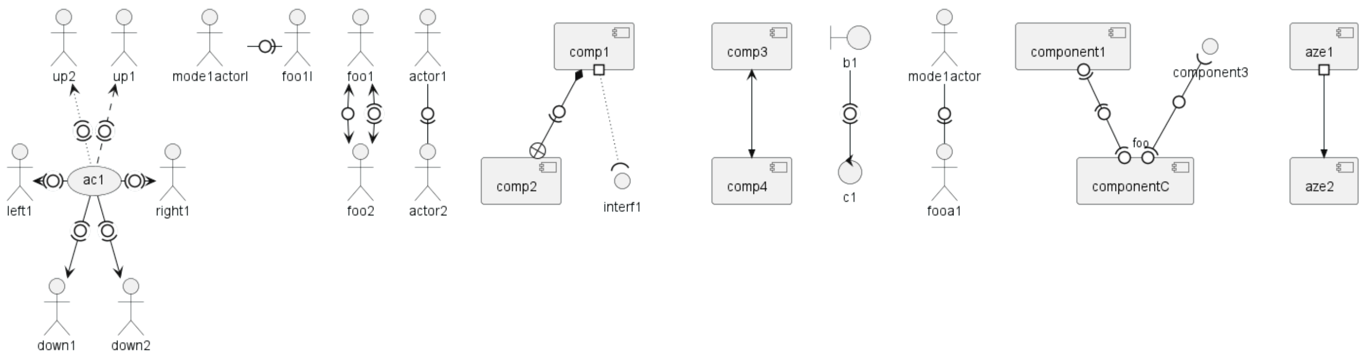
你也可以有以下类型。

```
@startuml

cloud cloud1
cloud cloud2
cloud cloud3
cloud cloud4
cloud cloud5
cloud1 -0- cloud2
cloud1 -0)- cloud3
cloud1 -(0- cloud4
cloud1 -(0)- cloud5

@enduml
```



或另一个例子。

```
@startuml
actor foo1
actor foo2
foo1 <-0-> foo2
foo1 <-(0)-> foo2

(ac1) -le(0)-> left1
ac1 -ri(0)-> right1
ac1 .up(0).> up1
ac1 ~up(0)~> up2
ac1 -do(0)-> down1
ac1 -do(0)-> down2

actor1 -0)- actor2

component comp1
component comp2
comp1 *-0)-+ comp2
[comp3] <-->> [comp4]

boundary b1
control c1
b1 -(0)- c1

component comp1
interface interf1
comp1 #~~( interf1

:mode1actor: -0)- fooa1
:mode1actorl: -ri0)- foo1l

[component1] 0)-(0-(0 [componentC]
() component3 )-0-(0 "foo" [componentC]

[aze1] #-->> [aze2]
@enduml
```



[Ref.QA-547andQA-1736]

见附录上的所有类型。

8.4 的箭头样式

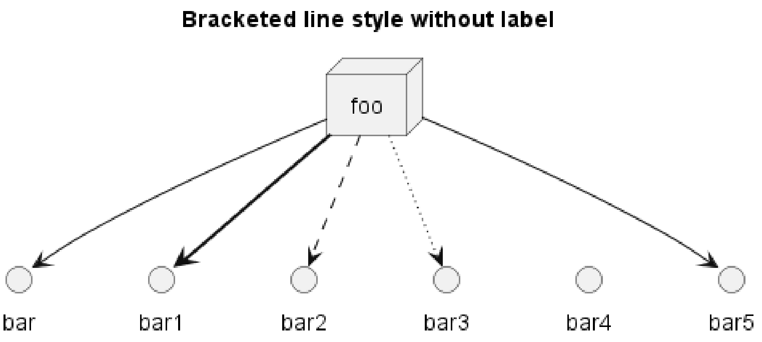
类似于括 号内的类关系（链接或箭头）样式

8.4.1 线条样式

也可以明确有 bold,dashed,dotted,hidden 或 plain 箭头:

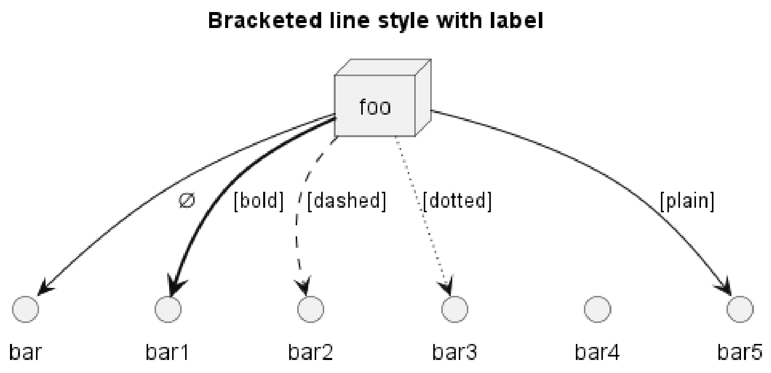
- 无标签

```
@startuml
node foo
title Bracketed line style without label
foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml
```



- 带标签

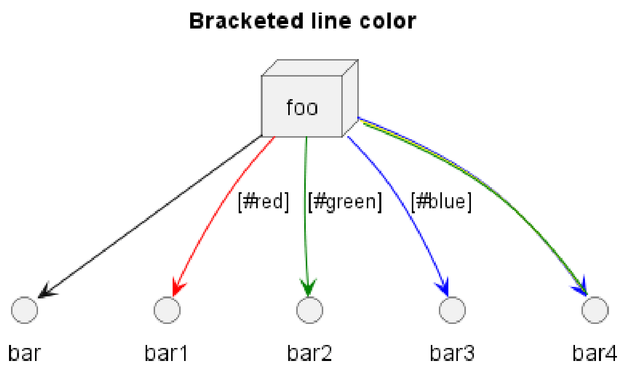
```
@startuml
title Bracketed line style with label
node foo
foo --> bar          :
foo -[bold]-> bar1   : [bold]
foo -[dashed]-> bar2 : [dashed]
foo -[dotted]-> bar3 : [dotted]
foo -[hidden]-> bar4 : [hidden]
foo -[plain]-> bar5  : [plain]
@enduml
```



[改编自 QA-4181]

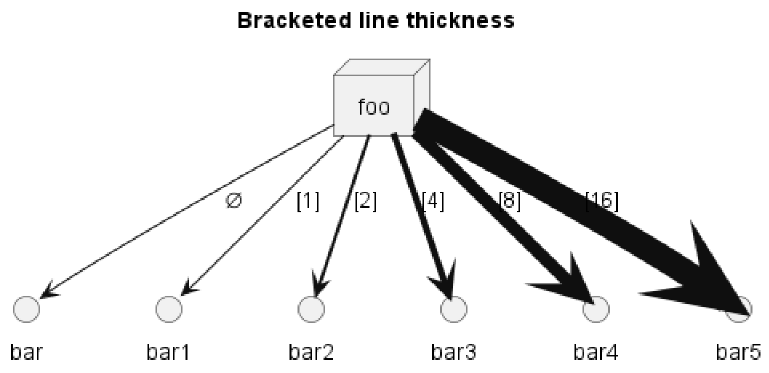
8.4.2 线条颜色

```
@startuml
title Bracketed line color
node  foo
foo --> bar
foo -[#red]-> bar1     : [#red]
foo -[#green]-> bar2   : [#green]
foo -[#blue]-> bar3    : [#blue]
foo -[#blue;#yellow;#green]-> bar4
@enduml
```



8.4.3 线条粗细

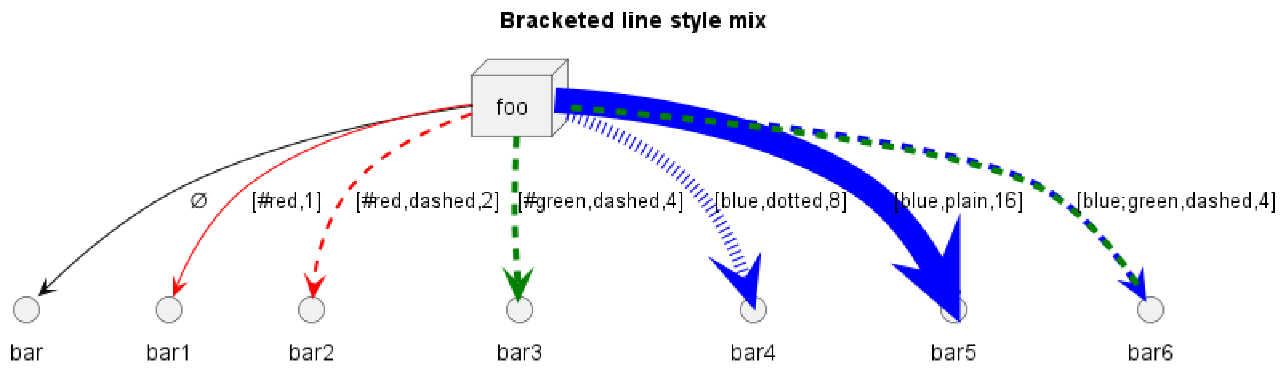
```
@startuml
title Bracketed line thickness
node foo
foo --> bar                 :
foo -[thickness=1]-> bar1   : [1]
foo -[thickness=2]-> bar2   : [2]
foo -[thickness=4]-> bar3   : [4]
foo -[thickness=8]-> bar4   : [8]
foo -[thickness=16]-> bar5  : [16]
@enduml
```



[改编自 QA-4949]

8.4.4 混合

```
@startuml
title Bracketed line style mix
node foo
foo --> bar                             :
foo -[#red,thickness=1]-> bar1          : [#red,1]
foo -[#red,dashed,thickness=2]-> bar2   : [#red,dashed,2]
foo -[#green,dashed,thickness=4]-> bar3 : [#green,dashed,4]
foo -[#blue,dotted,thickness=8]-> bar4  : [blue,dotted,8]
foo -[#blue,plain,thickness=16]-> bar5  : [blue,plain,16]
foo -[#blue;#green,dashed,thickness=4]-> bar6  : [blue;green,dashed,4]
@enduml
```

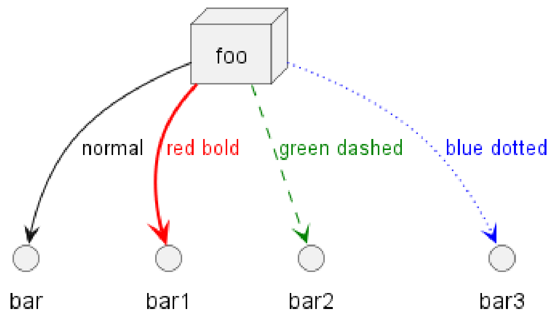


8.5 改变箭头的颜色和样式（内联式）

你可以使用内联式以下符号改变单个箭头的颜色或样式。

- `#color;line.[bold|dashed|dotted];text:color`

```
@startuml
node foo
foo --> bar : normal
foo --> bar1 #line:red;line.bold;text:red  : red bold
foo --> bar2 #green;line.dashed;text:green : green dashed
foo --> bar3 #blue;line.dotted;text:blue   : blue dotted
@enduml
```



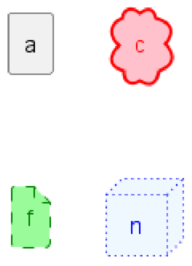
[参考 QA-3770 和 QA-3816] [[见类图](class-diagram#b5b0e4228f2e5022) 上的类似功能]

8.6 改变元素的颜色和样式（内联样式）

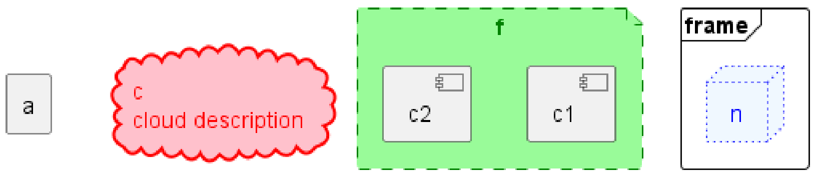
你可以用以下符号改变单个元素的颜色或样式。

- `#[color|back:color];line:color;line.[bold|dashed|dotted];text:color`

```
@startuml
agent a
cloud c #pink;line:red;line.bold;text:red
file  f #palegreen;line:green;line.dashed;text:green
node  n #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



```
@startuml
agent a
cloud c #pink;line:red;line.bold;text:red [
c
cloud description
]
file  f #palegreen;line:green;line.dashed;text:green {
[c1]
[c2]
}
frame frame {
node  n #aliceblue;line:blue;line.dotted;text:blue
}
@enduml
```

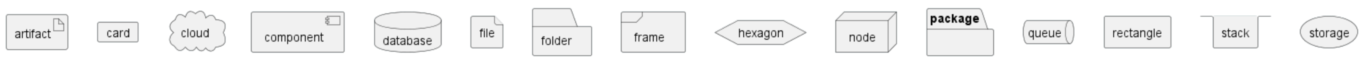


[Ref. QA-6852]

8.7 可嵌套的元素

这里是可嵌套的元素。

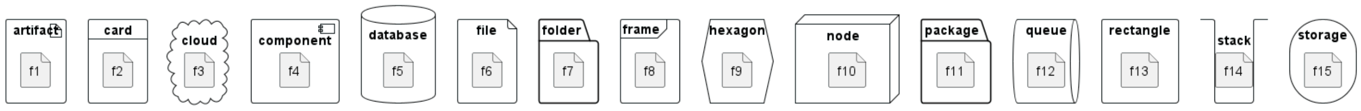
```
@startuml
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```



8.8 包和嵌套元素

8.8.1 一层的例子

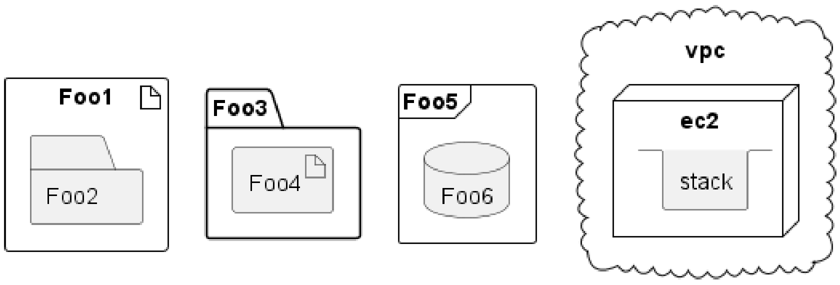
```
@startuml
artifact    artifactVeryL0000000000000000000g    as "artifact" {
file f1
}
card        cardVeryL0000000000000000000g        as "card" {
file f2
}
cloud       cloudVeryL0000000000000000000g       as "cloud" {
file f3
}
component   componentVeryL0000000000000000000g   as "component" {
file f4
}
database    databaseVeryL0000000000000000000g    as "database" {
file f5
}
file        fileVeryLOOOOOOOOOOOOOOOOOOOg        as "file" {
file f6
}
folder      folderVeryLOOOOOOOOOOOOOOOOOOOg      as "folder" {
file f7
}
frame       frameVeryLOOOOOOOOOOOOOOOOOOOg       as "frame" {
file f8
}
hexagon     hexagonVeryLOOOOOOOOOOOOOOOOOOOg     as "hexagon" {
file f9
}
node        nodeVeryLOOOOOOOOOOOOOOOOOOOg        as "node" {
file f10
}
package     packageVeryLOOOOOOOOOOOOOOOOOOOg     as "package" {
file f11
}
queue       queueVeryLOOOOOOOOOOOOOOOOOOOg       as "queue" {
file f12
}
rectangle   rectangleVeryLOOOOOOOOOOOOOOOOOOOg   as "rectangle" {
file f13
}
stack       stackVeryLOOOOOOOOOOOOOOOOOOOg       as "stack" {
file f14
}
storage     storageVeryLOOOOOOOOOOOOOOOOOOOg     as "storage" {
file f15
}
@enduml
```



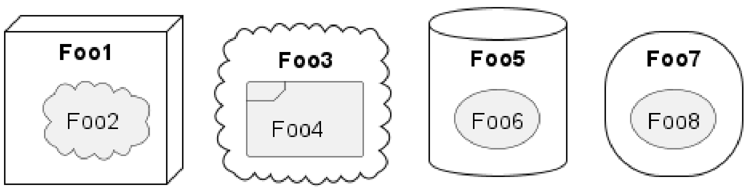
8.8.2 其他例子

```
@startuml
artifact Foo1 {
  folder Foo2
}

folder Foo3 {
  artifact Foo4
}

frame Foo5 {
  database Foo6
}

cloud vpc {
  node ec2 {
    stack stack
  }
}
```

```
@enduml
```



```
@startuml
node Foo1 {
 cloud Foo2
}

cloud Foo3 {
  frame Foo4
}

database Foo5  {
  storage Foo6
}

storage Foo7 {
  storage Foo8
}
@enduml
```

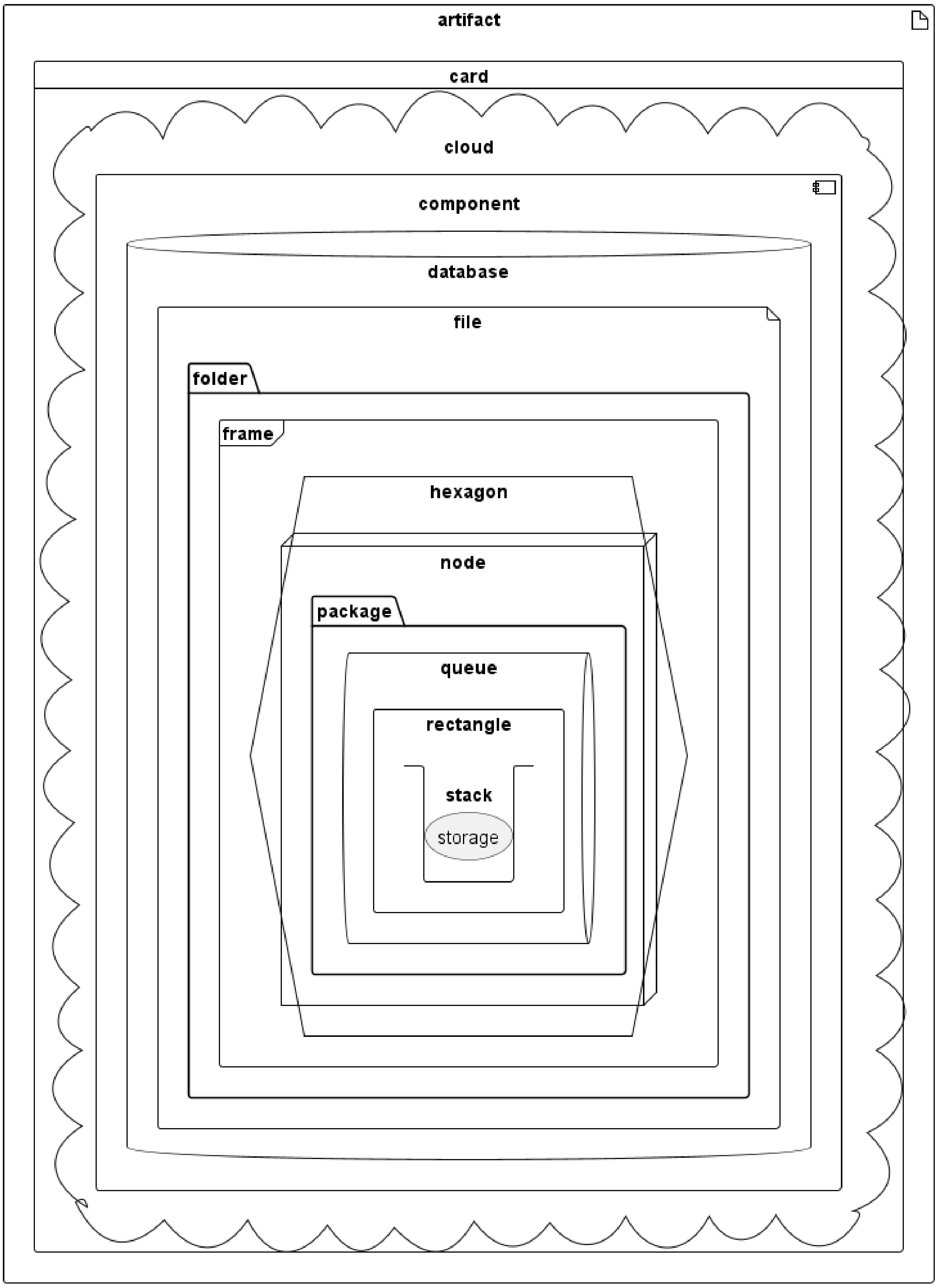


8.8.3 完全嵌套

这里是所有的嵌套元素:

- 按字母顺序。

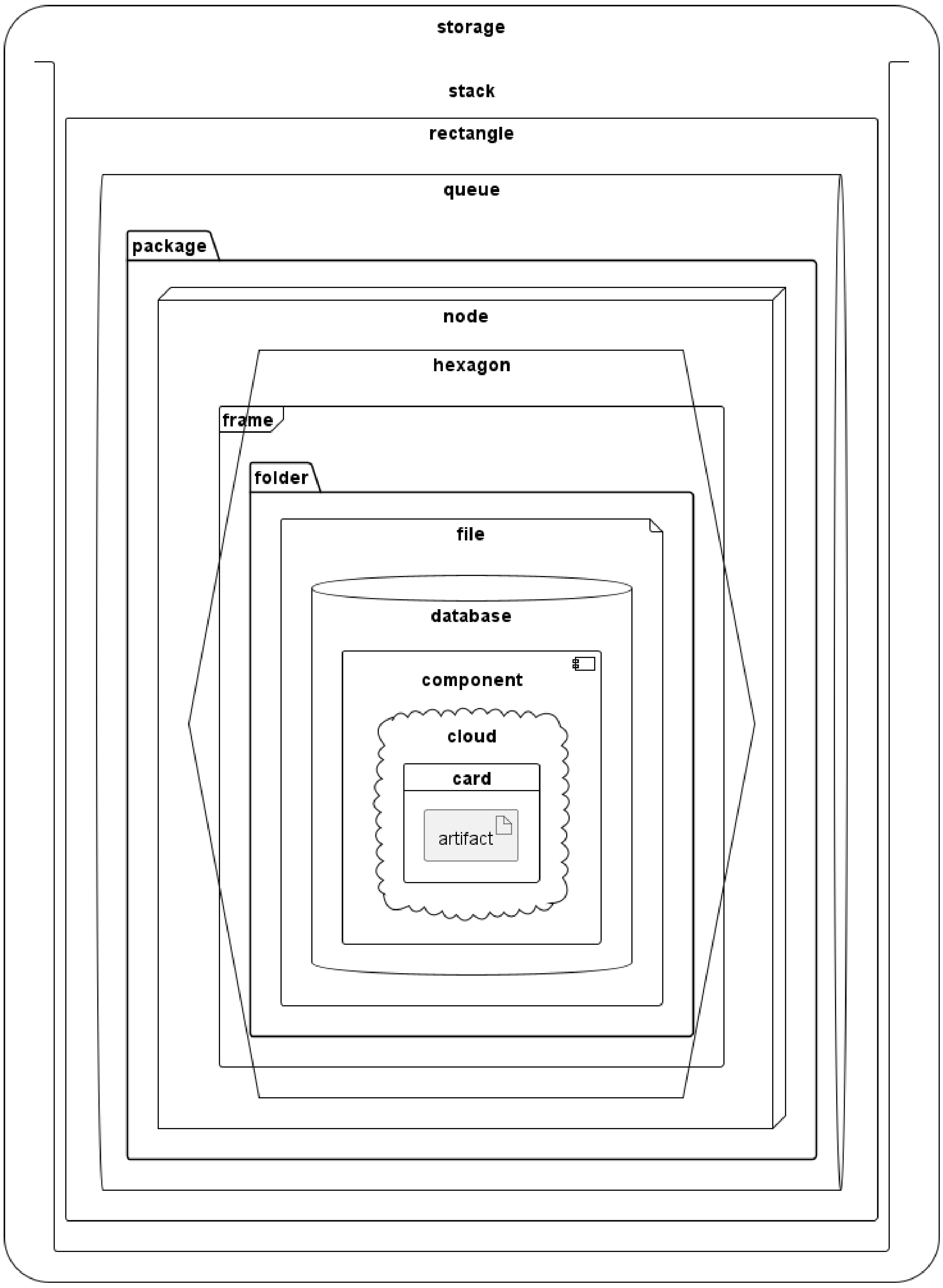
```
@startuml
artifact artifact {
card card {
cloud cloud {
component component {
database database {
file file {
folder folder {
frame frame {
hexagon hexagon {
node node {
package package {
queue queue {
rectangle rectangle {
stack stack {
storage storage {
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
@enduml
```



- 或相反的字母顺序

```
@startuml
storage storage {
stack stack {
rectangle rectangle {
queue queue {
package package {
```

```
node node {
hexagon hexagon {
frame frame {
folder folder {
file file {
database database {
component component {
cloud cloud {
card card {
artifact artifact {
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
@enduml
```

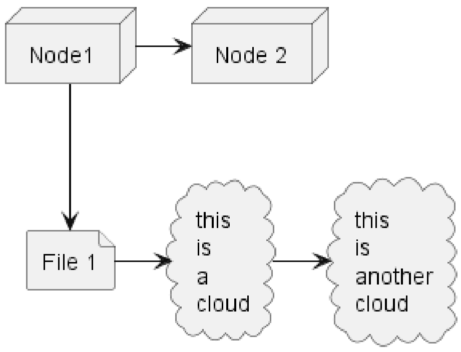


8.9 别名

8.9.1 简单的别名有 as

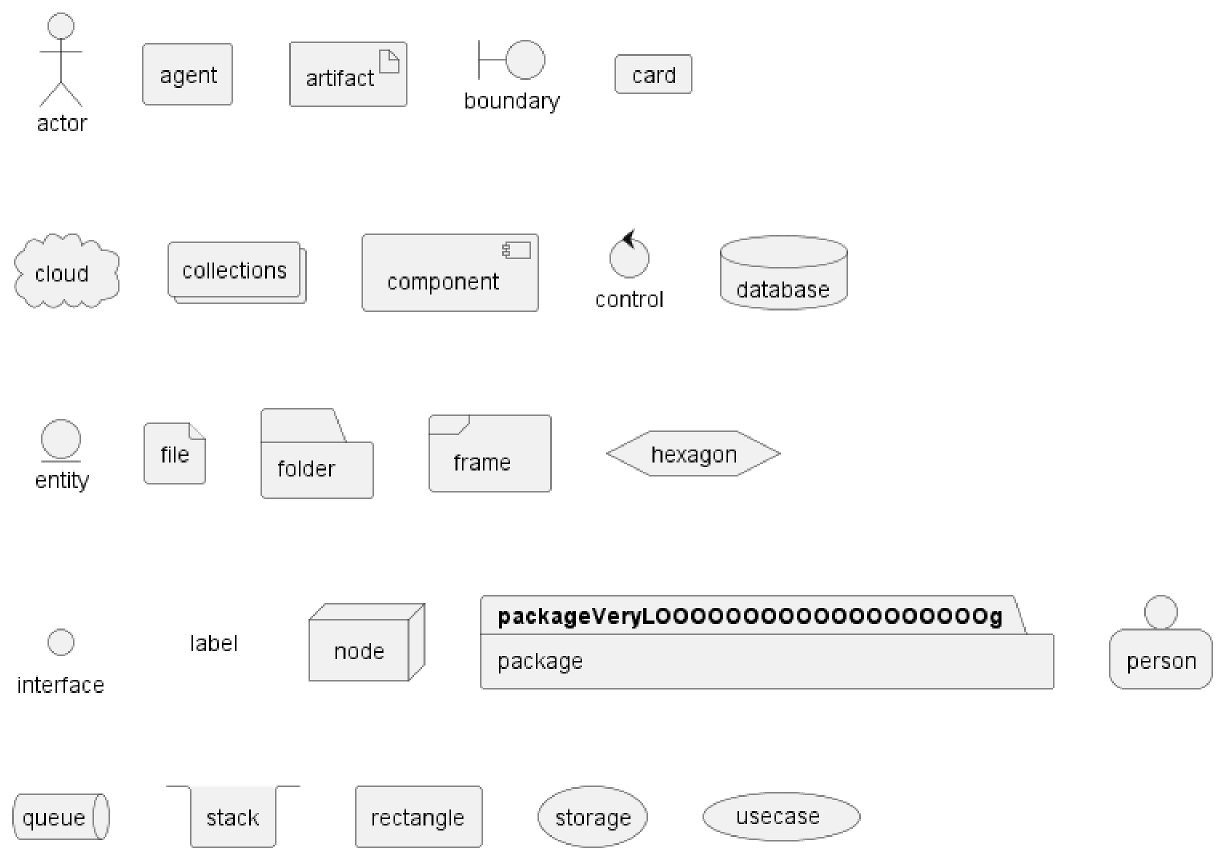
```
@startuml
node Node1 as n1
node "Node 2" as n2
```

```
file f1 as "File 1"
cloud c1 as "this
is
a
cloud"
cloud c2 [this
is
another
cloud]

n1 -> n2
n1 --> f1
f1 -> c1
c1 -> c2
@enduml
```

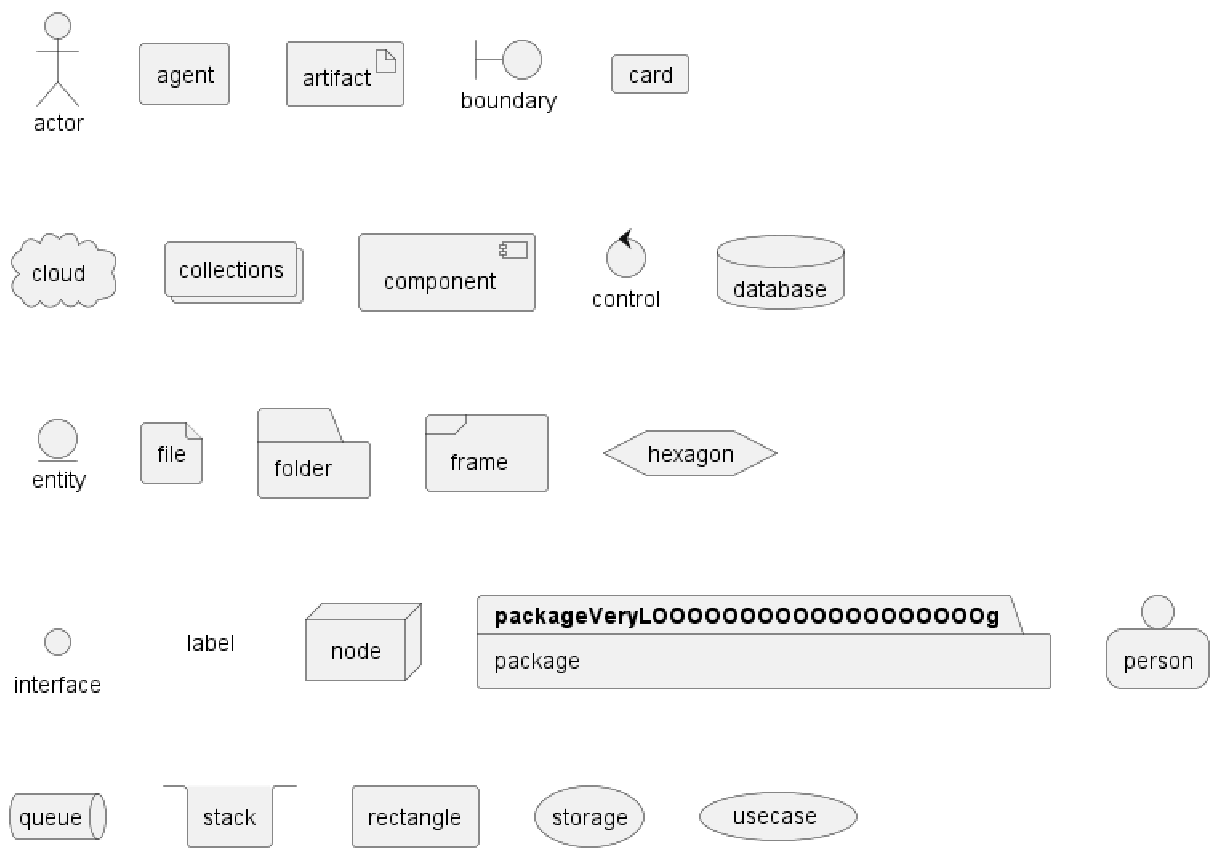


8.9.2 长别名的例子

```
@startuml
actor         "actor"        as actorVeryLOOOOOOOOOOOOOOOOOOOg
agent         "agent"        as agentVeryLOOOOOOOOOOOOOOOOOOOg
artifact      "artifact"     as artifactVeryLOOOOOOOOOOOOOOOOOOOg
boundary      "boundary"     as boundaryVeryLOOOOOOOOOOOOOOOOOOOg
card          "card"         as cardVeryLOOOOOOOOOOOOOOOOOOOg
cloud         "cloud"        as cloudVeryLOOOOOOOOOOOOOOOOOOOg
collections   "collections"  as collectionsVeryLOOOOOOOOOOOOOOOOOOOg
component     "component"    as componentVeryLOOOOOOOOOOOOOOOOOOOg
control       "control"      as controlVeryLOOOOOOOOOOOOOOOOOOOg
database      "database"     as databaseVeryLOOOOOOOOOOOOOOOOOOOg
entity        "entity"       as entityVeryLOOOOOOOOOOOOOOOOOOOg
file          "file"         as fileVeryLOOOOOOOOOOOOOOOOOOOg
folder        "folder"       as folderVeryLOOOOOOOOOOOOOOOOOOOg
frame         "frame"        as frameVeryLOOOOOOOOOOOOOOOOOOOg
hexagon       "hexagon"      as hexagonVeryLOOOOOOOOOOOOOOOOOOOg
interface     "interface"    as interfaceVeryLOOOOOOOOOOOOOOOOOOOg
label         "label"        as labelVeryLOOOOOOOOOOOOOOOOOOOg
node          "node"         as nodeVeryLOOOOOOOOOOOOOOOOOOOg
package       "package"      as packageVeryLOOOOOOOOOOOOOOOOOOOg
person        "person"       as personVeryLOOOOOOOOOOOOOOOOOOOg
queue         "queue"        as queueVeryLOOOOOOOOOOOOOOOOOOOg
stack         "stack"        as stackVeryLOOOOOOOOOOOOOOOOOOOg
rectangle     "rectangle"    as rectangleVeryLOOOOOOOOOOOOOOOOOOOg
storage       "storage"      as storageVeryLOOOOOOOOOOOOOOOOOOOg
usecase       "usecase"      as usecaseVeryLOOOOOOOOOOOOOOOOOOOg
@enduml
```



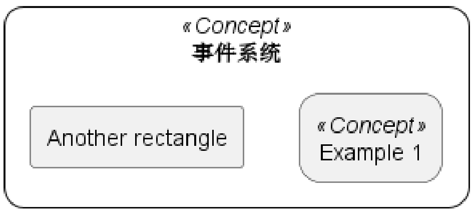
```
@startuml
actor       actorVeryLOOOOOOOOOOOOOOOOOOOg       as "actor"
agent       agentVeryLOOOOOOOOOOOOOOOOOOOg       as "agent"
artifact    artifactVeryLOOOOOOOOOOOOOOOOOOOg    as "artifact"
boundary    boundaryVeryLOOOOOOOOOOOOOOOOOOOg    as "boundary"
card        cardVeryLOOOOOOOOOOOOOOOOOOOg        as "card"
cloud       cloudVeryLOOOOOOOOOOOOOOOOOOOg       as "cloud"
collections collectionsVeryLOOOOOOOOOOOOOOOOOOOg as "collections"
component   componentVeryLOOOOOOOOOOOOOOOOOOOg   as "component"
control     controlVeryLOOOOOOOOOOOOOOOOOOOg     as "control"
database    databaseVeryLOOOOOOOOOOOOOOOOOOOg    as "database"
entity      entityVeryLOOOOOOOOOOOOOOOOOOOg      as "entity"
file        fileVeryLOOOOOOOOOOOOOOOOOOOg        as "file"
folder      folderVeryLOOOOOOOOOOOOOOOOOOOg      as "folder"
frame       frameVeryLOOOOOOOOOOOOOOOOOOOg       as "frame"
hexagon     hexagonVeryLOOOOOOOOOOOOOOOOOOOg     as "hexagon"
interface   interfaceVeryLOOOOOOOOOOOOOOOOOOOg   as "interface"
label       labelVeryLOOOOOOOOOOOOOOOOOOOg       as "label"
node        nodeVeryLOOOOOOOOOOOOOOOOOOOg        as "node"
package     packageVeryLOOOOOOOOOOOOOOOOOOOg     as "package"
person      personVeryLOOOOOOOOOOOOOOOOOOOg      as "person"
queue       queueVeryLOOOOOOOOOOOOOOOOOOOg       as "queue"
stack       stackVeryLOOOOOOOOOOOOOOOOOOOg       as "stack"
rectangle   rectangleVeryLOOOOOOOOOOOOOOOOOOOg   as "rectangle"
storage     storageVeryLOOOOOOOOOOOOOOOOOOOg     as "storage"
usecase     usecaseVeryLOOOOOOOOOOOOOOOOOOOg     as "usecase"
@enduml
```



[参考资料：QA-12082]

8.10 圆角

```
@startuml
skinparam rectangle {
    roundCorner<<Concept>> 25
}

rectangle "事件系统" <<Concept>> {
rectangle "Example 1" <<Concept>> as ex1
rectangle "Another rectangle"
}
@enduml
```

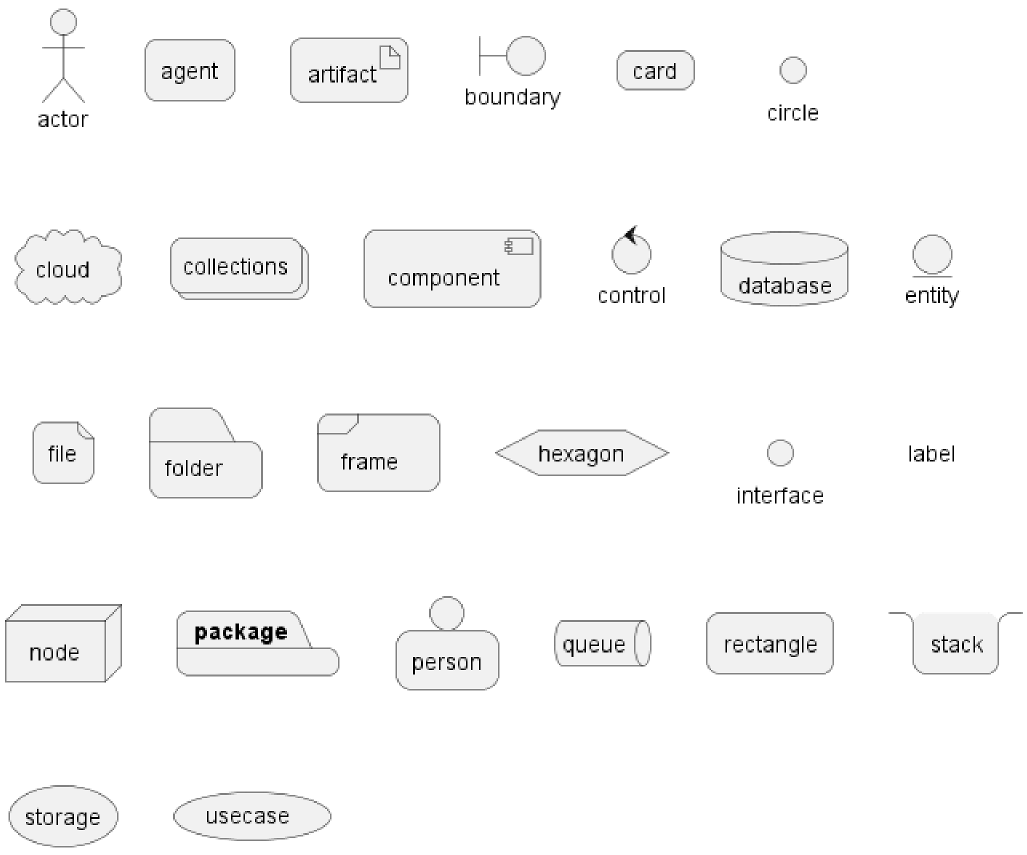


8.11 特定的 SkinParameter

8.11.1 圆角

```
@startuml
skinparam roundCorner 15
actor actor
agent agent
artifact artifact
```

```
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
@enduml
```

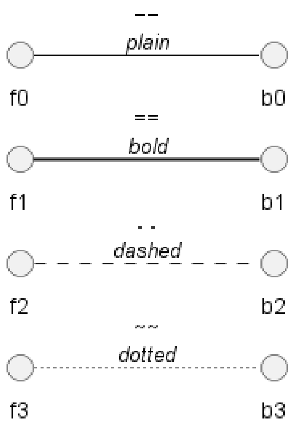


[参考 QA-5299,QA-6915,QA-11943] 。

8.12 附录。所有类型的箭线

```
@startuml
left to right direction
skinparam nodesep 5
```

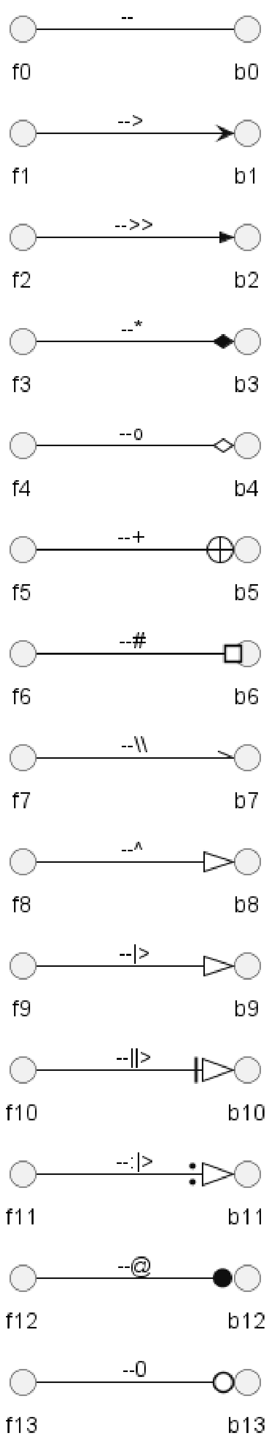
```
f3  ~~  b3  : ""~~""\n//dotted//
f2  ..  b2  : ""..""\n//dashed//
f1  ==  b1  : ""==""\n//bold//
f0  --  b0  : ""--""\n//plain//
@enduml
```



8.13 附录。所有类型的箭头或"0 " 箭头

8.13.1 箭头类型

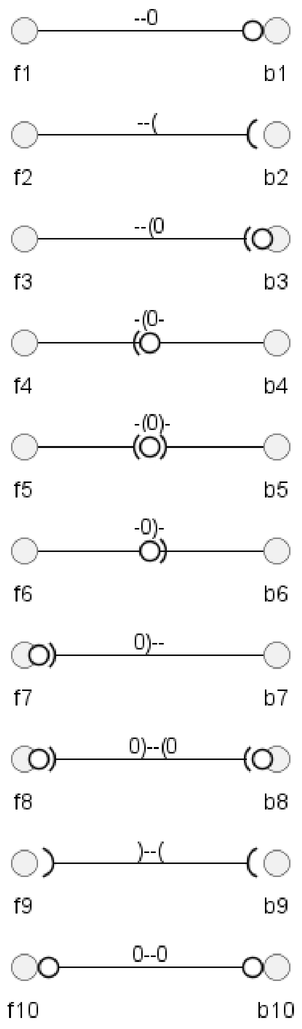
```
@startuml
left to right direction
skinparam nodesep 5

f13 --0   b13 : ""--0""
f12 --@   b12 : ""--@""
f11 --:|> b11 : ""--:|>""
f10 --||> b10 : ""--||>""
f9  --|>  b9  : ""--|>""
f8  --^   b8  : ""--^ ""
f7  --\\  b7  : ""--\\\\""
f6  --#   b6  : ""--# ""
f5  --+   b5  : ""--+ ""
f4  --o   b4  : ""--o ""
f3  --*   b3  : ""--* ""
f2  -->>  b2  : ""-->>""
f1  -->   b1  : ""--> ""
f0  --    b0  : ""--  ""
@enduml
```



8.13.2 0 ” 箭头或圆形箭头的类型

```
@startuml
left to right direction
skinparam nodesep 5

f10 0--0 b10 : "" 0--0 ""
f9 )--(  b9  : "" )--( ""
f8 0)--(0 b8 : "" 0)--(0""
f7 0)--  b7  : "" 0)-- ""
f6 -0)-  b6  : "" -0)- ""
f5 -(0)- b5  : "" -(0)-""
```

```
f4 -(0-  b4  : "" -(0- ""
f3 --(0  b3  : "" --(0 ""
f2 --(   b2  : "" --(  ""
f1 --0   b1  : "" --0  ""
@enduml
```

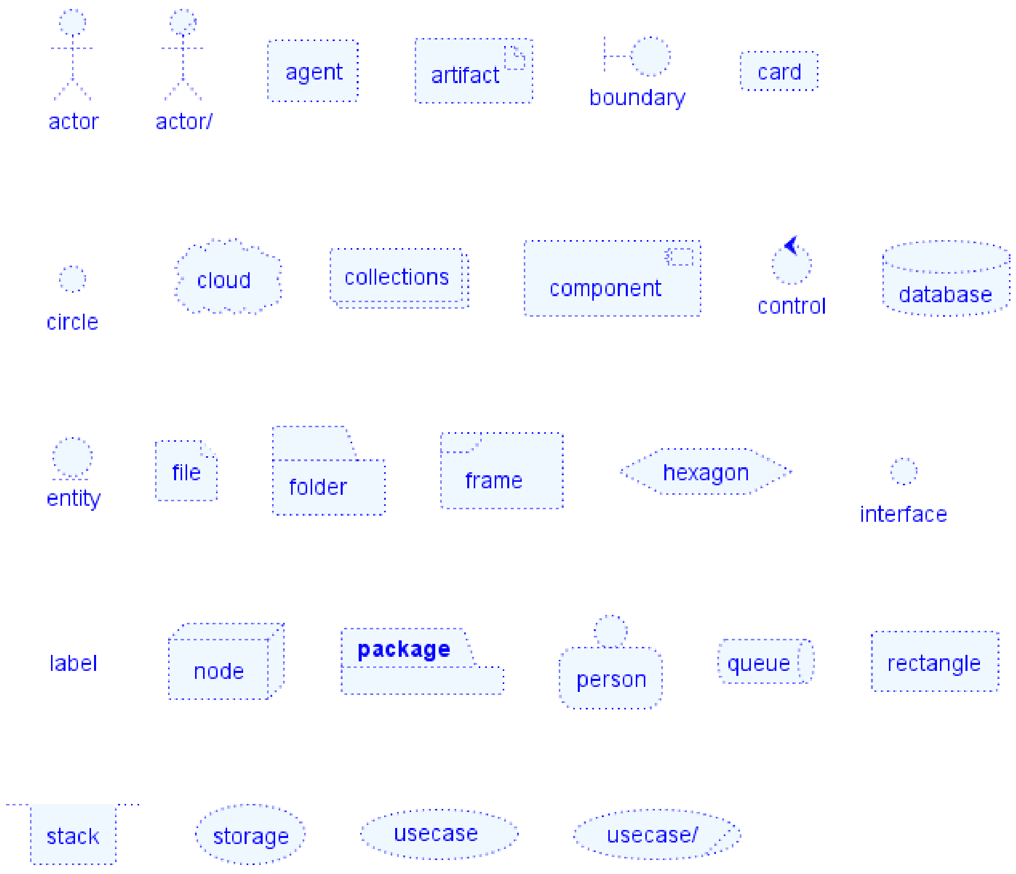


8.14 附录。测试所有元素的内联样式

8.14.1 简单元素

```
@startuml
actor actor             #aliceblue;line:blue;line.dotted;text:blue
actor/ "actor/"         #aliceblue;line:blue;line.dotted;text:blue
agent agent             #aliceblue;line:blue;line.dotted;text:blue
artifact artifact       #aliceblue;line:blue;line.dotted;text:blue
boundary boundary       #aliceblue;line:blue;line.dotted;text:blue
card card               #aliceblue;line:blue;line.dotted;text:blue
circle circle           #aliceblue;line:blue;line.dotted;text:blue
cloud cloud             #aliceblue;line:blue;line.dotted;text:blue
collections collections #aliceblue;line:blue;line.dotted;text:blue
component component     #aliceblue;line:blue;line.dotted;text:blue
control control         #aliceblue;line:blue;line.dotted;text:blue
database database       #aliceblue;line:blue;line.dotted;text:blue
entity entity           #aliceblue;line:blue;line.dotted;text:blue
file file               #aliceblue;line:blue;line.dotted;text:blue
folder folder           #aliceblue;line:blue;line.dotted;text:blue
```

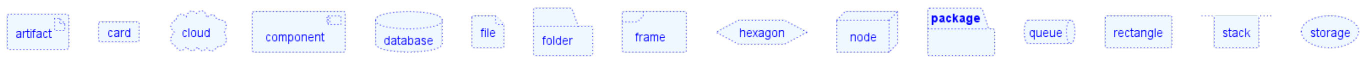
```
frame frame             #aliceblue;line:blue;line.dotted;text:blue
hexagon hexagon         #aliceblue;line:blue;line.dotted;text:blue
interface interface     #aliceblue;line:blue;line.dotted;text:blue
label label             #aliceblue;line:blue;line.dotted;text:blue
node node               #aliceblue;line:blue;line.dotted;text:blue
package package         #aliceblue;line:blue;line.dotted;text:blue
person person           #aliceblue;line:blue;line.dotted;text:blue
queue queue             #aliceblue;line:blue;line.dotted;text:blue
rectangle rectangle     #aliceblue;line:blue;line.dotted;text:blue
stack stack             #aliceblue;line:blue;line.dotted;text:blue
storage storage         #aliceblue;line:blue;line.dotted;text:blue
usecase usecase         #aliceblue;line:blue;line.dotted;text:blue
usecase/ "usecase/"     #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



8.14.2 嵌套元素

8.14.3 没有子元素

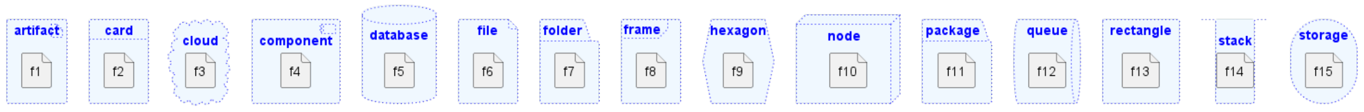
```
@startuml
artifact artifact #aliceblue;line:blue;line.dotted;text:blue {
}
card card #aliceblue;line:blue;line.dotted;text:blue {
}
cloud cloud #aliceblue;line:blue;line.dotted;text:blue {
}
component component #aliceblue;line:blue;line.dotted;text:blue {
}
database database #aliceblue;line:blue;line.dotted;text:blue {
}
file file #aliceblue;line:blue;line.dotted;text:blue {
}
folder folder #aliceblue;line:blue;line.dotted;text:blue {
}
frame frame #aliceblue;line:blue;line.dotted;text:blue {
}
hexagon hexagon #aliceblue;line:blue;line.dotted;text:blue {
}
node node #aliceblue;line:blue;line.dotted;text:blue {
}
package package #aliceblue;line:blue;line.dotted;text:blue {
}
queue queue #aliceblue;line:blue;line.dotted;text:blue {
}
rectangle rectangle #aliceblue;line:blue;line.dotted;text:blue {
}
stack stack #aliceblue;line:blue;line.dotted;text:blue {
}
storage storage #aliceblue;line:blue;line.dotted;text:blue {
}
@enduml
```



8.14.4 有子元素

```
@startuml
artifact    artifactVeryL0000000000000000000g    as "artifact" #aliceblue;line:blue;line.dotted;text
file f1
}
card        cardVeryL0000000000000000000g        as "card" #aliceblue;line:blue;line.dotted;text:blu
file f2
}
cloud       cloudVeryL0000000000000000000g       as "cloud" #aliceblue;line:blue;line.dotted;text:bl
file f3
}
component   componentVeryL0000000000000000000g   as "component" #aliceblue;line:blue;line.dotted;tex
file f4
}
database    databaseVeryL0000000000000000000g    as "database" #aliceblue;line:blue;line.dotted;text
file f5
}
file        fileVeryL0000000000000000000g        as "file" #aliceblue;line:blue;line.dotted;text:blu
file f6
}
folder      folderVeryL0000000000000000000g      as "folder" #aliceblue;line:blue;line.dotted;text:b
file f7
}
frame       frameVeryL0000000000000000000g       as "frame" #aliceblue;line:blue;line.dotted;text:bl
file f8
}
hexagon     hexagonVeryL0000000000000000000g     as "hexagon" #aliceblue;line:blue;line.dotted;text:
file f9
}
node        nodeVeryL0000000000000000000g        as "node" #aliceblue;line:blue;line.dotted;text:blu
file f10
}
package     packageVeryL0000000000000000000g     as "package" #aliceblue;line:blue;line.dotted;text:
```

```
file f11
}
queue       queueVeryL0000000000000000000g      as "queue" #aliceblue;line:blue;line.dotted;text:blu
file f12
}
rectangle   rectangleVeryL0000000000000000000g  as "rectangle" #aliceblue;line:blue;line.dotted;text
file f13
}
stack       stackVeryL0000000000000000000g      as "stack" #aliceblue;line:blue;line.dotted;text:blu
file f14
}
storage     storageVeryL0000000000000000000g    as "storage" #aliceblue;line:blue;line.dotted;text:b
file f15
}
@enduml
```



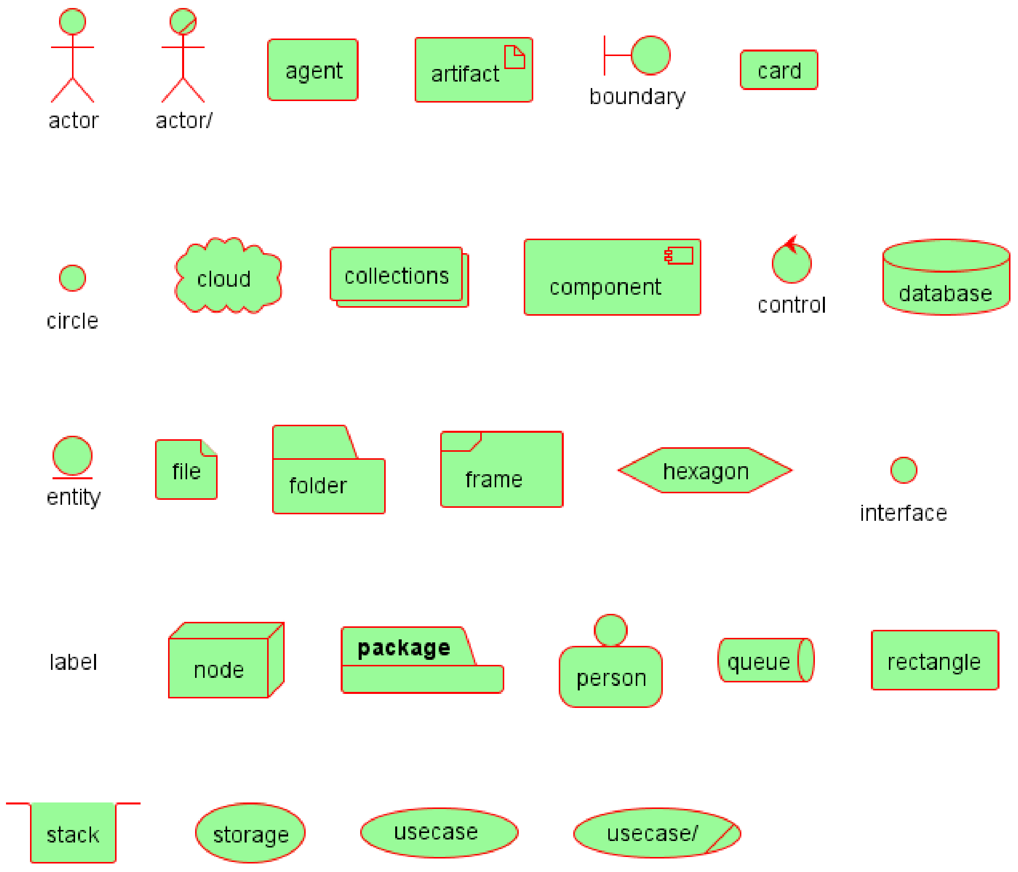
8.15 附录。对所有元素的风格测试

8.15.1 简单元素

8.15.2 全局风格（在 **ComponentDiagram** 上）。

```
@startuml
<style>
componentDiagram {
  BackGroundColor palegreen
  LineThickness 1
  LineColor red
}
document {
  BackGroundColor white
}
</style>
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
```

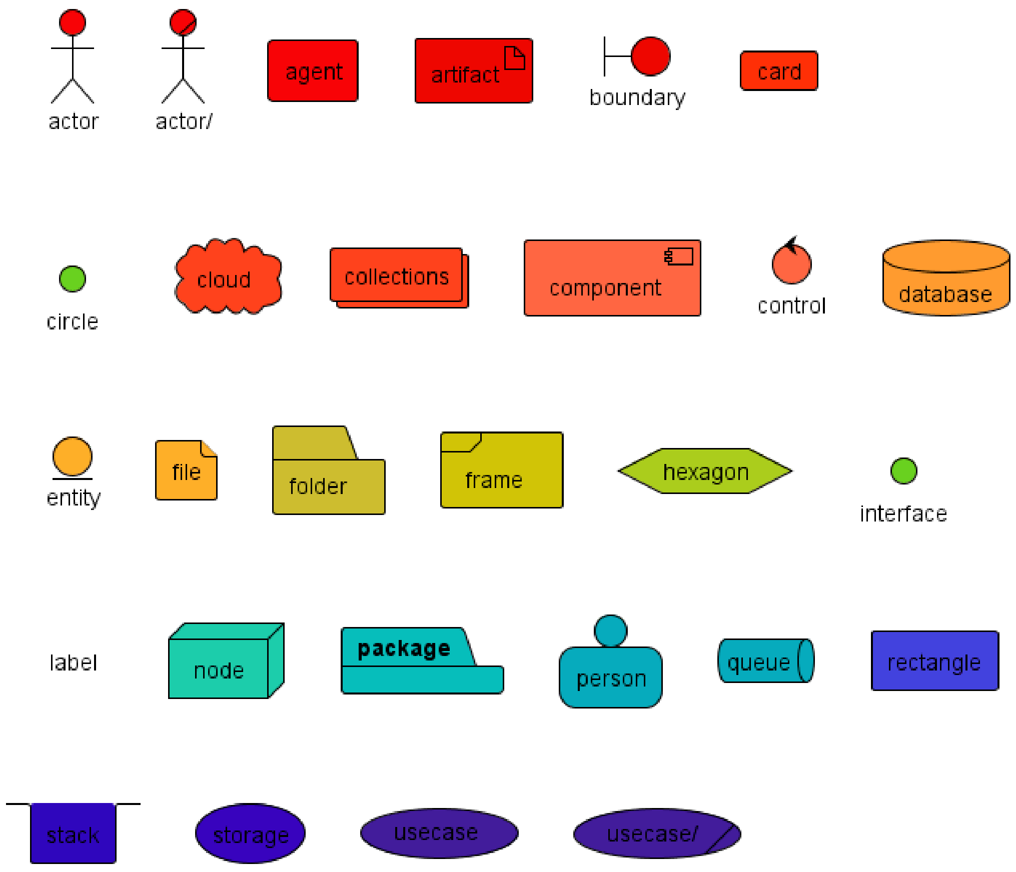
```
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml
```



8.15.3 每个元素的风格

```
@startuml
<style>
actor {
  BackGroundColor #f80c12
  LineThickness 1
  LineColor black
}
agent {
  BackGroundColor #f80c12
  LineThickness 1
  LineColor black
}
artifact {
  BackGroundColor #ee1100
  LineThickness 1
  LineColor black
}
boundary {
  BackGroundColor #ee1100
  LineThickness 1
  LineColor black
}
card {
  BackGroundColor #ff3311
  LineThickness 1
  LineColor black
}
circle {
  BackGroundColor #ff3311
  LineThickness 1
  LineColor black
}
cloud {
  BackGroundColor #ff4422
  LineThickness 1
  LineColor black
}
collections {
  BackGroundColor #ff4422
  LineThickness 1
  LineColor black
}
component {
  BackGroundColor #ff6644
  LineThickness 1
  LineColor black
}
control {
  BackGroundColor #ff6644
  LineThickness 1
  LineColor black
}
database {
  BackGroundColor #ff9933
  LineThickness 1
  LineColor black
}
entity {
  BackGroundColor #feae2d
  LineThickness 1
  LineColor black
}
file {
  BackGroundColor #feae2d
  LineThickness 1
  LineColor black
}
folder {
  BackGroundColor #ccbb33
  LineThickness 1
  LineColor black
}
frame {
  BackGroundColor #d0c310
  LineThickness 1
  LineColor black
}
hexagon {
  BackGroundColor #aacc22
  LineThickness 1
  LineColor black
}
interface {
  BackGroundColor #69d025
  LineThickness 1
  LineColor black
}
label {
  BackGroundColor black
  LineThickness 1
  LineColor black
}
node {
  BackGroundColor #22ccaa
  LineThickness 1
  LineColor black
}
package {
  BackGroundColor #12bdb9
  LineThickness 1
  LineColor black
}
person {
  BackGroundColor #11aabb
  LineThickness 1
  LineColor black
}
queue {
  BackGroundColor #11aabb
  LineThickness 1
  LineColor black
}
rectangle {
  BackGroundColor #4444dd
  LineThickness 1
  LineColor black
}
stack {
  BackGroundColor #3311bb
  LineThickness 1
  LineColor black
}
storage {
  BackGroundColor #3b0cbd
  LineThickness 1
  LineColor black
}
usecase {
  BackGroundColor #442299
  LineThickness 1
  LineColor black
}
</style>
actor actor
actor/ "actor/"
agent agent
```

```
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml
```

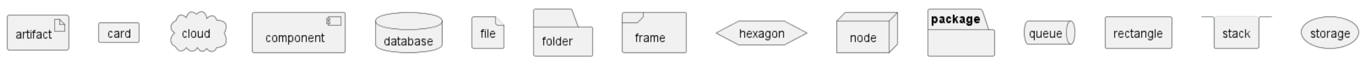


[参考 QA-13261]

8.15.4 嵌套元素 (无级别)

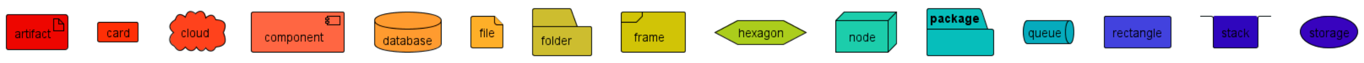
8.15.5 全局风格（在组件图上）。

```
@startuml
<style>
componentDiagram {
  BackGroundColor palegreen
  LineThickness 2
  LineColor red
}
</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```



8.15.6 每个嵌套元素的样式

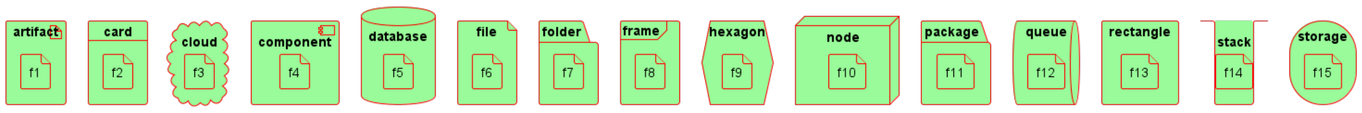
```
@startuml
<style>
artifact {
  BackGroundColor #ee1100
  LineThickness 1
  LineColor black
}
card {
  BackGroundColor #ff3311
  LineThickness 1
  LineColor black
}
cloud {
  BackGroundColor #ff4422
  LineThickness 1
  LineColor black
}
component {
  BackGroundColor #ff6644
  LineThickness 1
  LineColor black
}
database {
  BackGroundColor #ff9933
  LineThickness 1
  LineColor black
}
file {
  BackGroundColor #feae2d
  LineThickness 1
  LineColor black
}
folder {
  BackGroundColor #ccbb33
  LineThickness 1
  LineColor black
}
frame {
  BackGroundColor #d0c310
  LineThickness 1
  LineColor black
}
hexagon {
  BackGroundColor #aacc22
  LineThickness 1
  LineColor black
}
node {
  BackGroundColor #22ccaa
  LineThickness 1
  LineColor black
}
package {
  BackGroundColor #12bdb9
  LineThickness 1
  LineColor black
}
queue {
  BackGroundColor #11aabb
  LineThickness 1
  LineColor black
}
rectangle {
  BackGroundColor #4444dd
  LineThickness 1
  LineColor black
}
stack {
  BackGroundColor #3311bb
  LineThickness 1
  LineColor black
}
storage {
  BackGroundColor #3b0cbd
  LineThickness 1
  LineColor black
}

</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```



8.15.7 嵌套元素（有一个层次）

8.15.8 全局风格（在组件图上）。

```
@startuml
<style>
componentDiagram {
  BackGroundColor palegreen
  LineThickness 1
  LineColor red
}
document {
  BackGroundColor white
}
</style>
artifact e1 as "artifact" {
file f1
}
card e2 as "card" {
file f2
}
cloud e3 as "cloud" {
file f3
}
component e4 as "component" {
file f4
}
database e5 as "database" {
file f5
}
file e6 as "file" {
file f6
}
folder e7 as "folder" {
file f7
}
frame e8 as "frame" {
file f8
}
hexagon e9 as "hexagon" {
file f9
}
node e10 as "node" {
file f10
}
package e11 as "package" {
file f11
}
queue e12 as "queue" {
file f12
}
rectangle e13 as "rectangle" {
file f13
}
stack e14 as "stack" {
file f14
}
storage e15 as "storage" {
file f15
}
@enduml
```

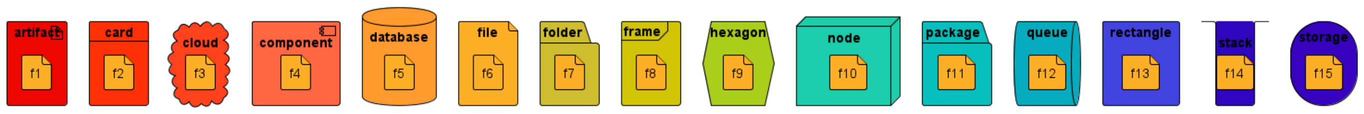


8.15.9 每个嵌套元素的样式

```
@startuml
<style>
```

```
artifact {
  BackGroundColor #ee1100
  LineThickness 1
  LineColor black
}
card {
  BackGroundColor #ff3311
  LineThickness 1
  LineColor black
}
cloud {
  BackGroundColor #ff4422
  LineThickness 1
  LineColor black
}
component {
  BackGroundColor #ff6644
  LineThickness 1
  LineColor black
}
database {
  BackGroundColor #ff9933
  LineThickness 1
  LineColor black
}
file {
  BackGroundColor #feae2d
  LineThickness 1
  LineColor black
}
folder {
  BackGroundColor #ccbb33
  LineThickness 1
  LineColor black
}
frame {
  BackGroundColor #d0c310
  LineThickness 1
  LineColor black
}
hexagon {
  BackGroundColor #aacc22
  LineThickness 1
  LineColor black
}
node {
  BackGroundColor #22ccaa
  LineThickness 1
  LineColor black
}
package {
  BackGroundColor #12bdb9
  LineThickness 1
  LineColor black
}
queue {
  BackGroundColor #11aabb
  LineThickness 1
  LineColor black
}
rectangle {
  BackGroundColor #4444dd
  LineThickness 1
  LineColor black
}
stack {
  BackGroundColor #3311bb
  LineThickness 1
  LineColor black
}
storage {
  BackGroundColor #3b0cbd
  LineThickness 1
  LineColor black
}
</style>
artifact e1 as "artifact" {
file f1
}
card e2 as "card" {
file f2
}
cloud e3 as "cloud" {
file f3
}
component e4 as "component" {
file f4
}
database e5 as "database" {
file f5
}
file e6 as "file" {
file f6
}
folder e7 as "folder" {
file f7
}
frame e8 as "frame" {
file f8
}
hexagon e9 as "hexagon" {
file f9
}
node e10 as "node" {
file f10
}
package e11 as "package" {
file f11
}
queue e12 as "queue" {
file f12
}
rectangle e13 as "rectangle" {
file f13
}
stack e14 as "stack" {
```

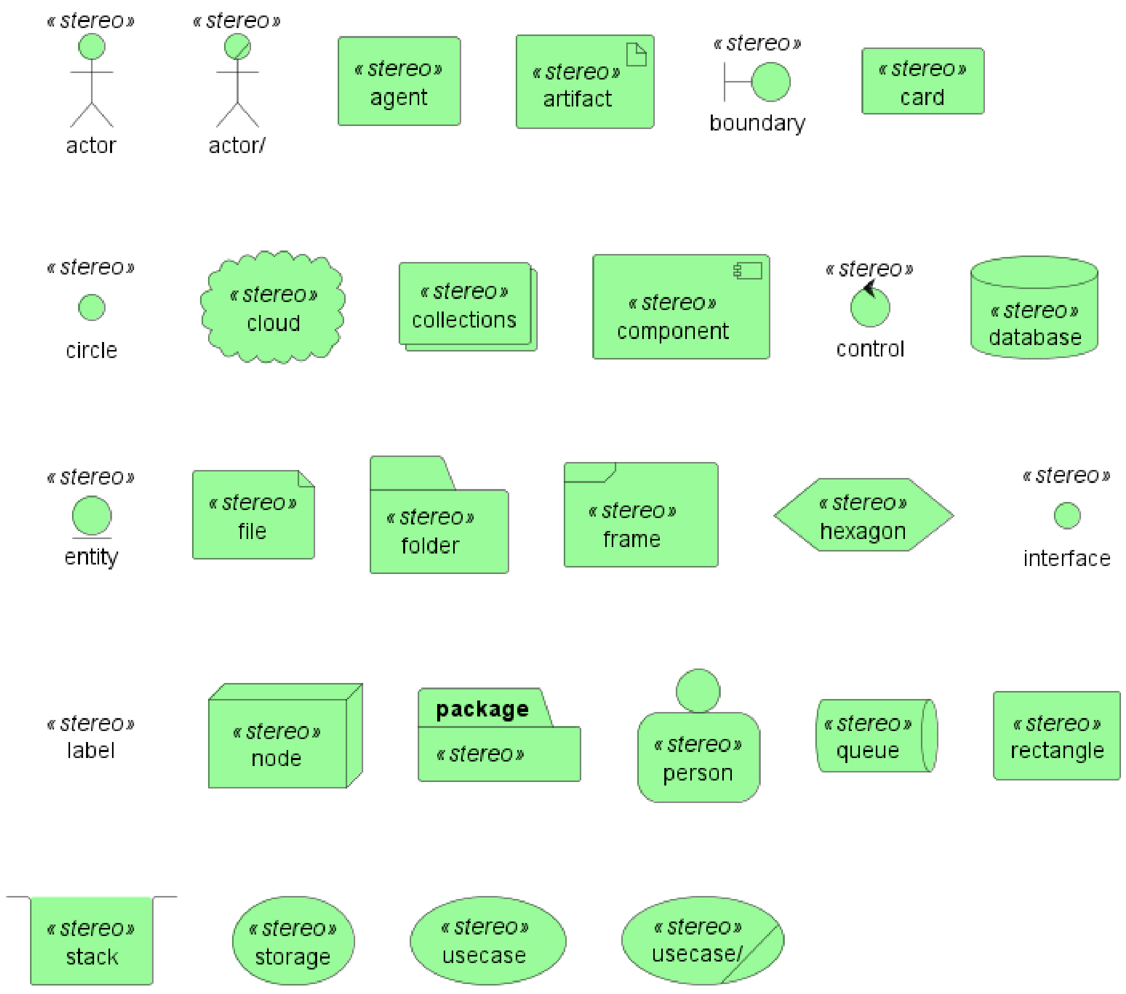
```
file f14
}
storage e15 as "storage" {
file f15
}
@enduml
```



8.16 附录。在所有元素上用风格测试定型

8.16.1 简单元素

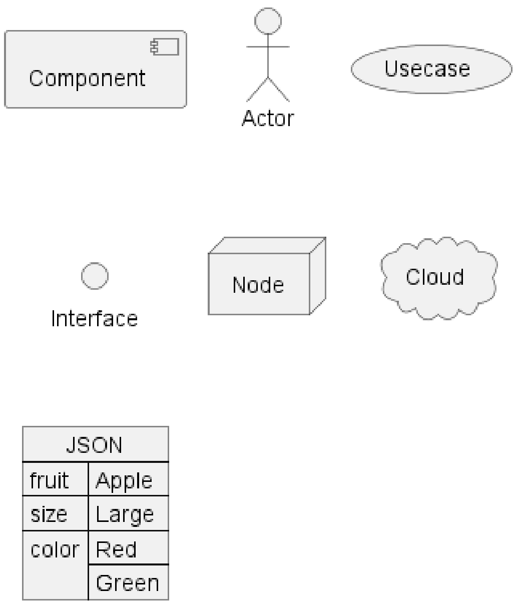
```
@startuml
<style>
.stereo {
  BackgroundColor palegreen
}
</style>
actor actor << stereo >>
actor/ "actor/" << stereo >>
agent agent << stereo >>
artifact artifact << stereo >>
boundary boundary << stereo >>
card card << stereo >>
circle circle << stereo >>
cloud cloud << stereo >>
collections collections << stereo >>
component component << stereo >>
control control << stereo >>
database database << stereo >>
entity entity << stereo >>
file file << stereo >>
folder folder << stereo >>
frame frame << stereo >>
hexagon hexagon << stereo >>
interface interface << stereo >>
label label << stereo >>
node node << stereo >>
package package << stereo >>
person person << stereo >>
queue queue << stereo >>
rectangle rectangle << stereo >>
stack stack << stereo >>
storage storage << stereo >>
usecase usecase << stereo >>
usecase/ "usecase/" << stereo >>
@enduml
```



8.17 Display JSON Data on Deployment diagram

8.17.1 Simple example

```
@startuml
allowmixing

component Component
actor     Actor
usecase   Usecase
()        Interface
node      Node
cloud     Cloud

json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

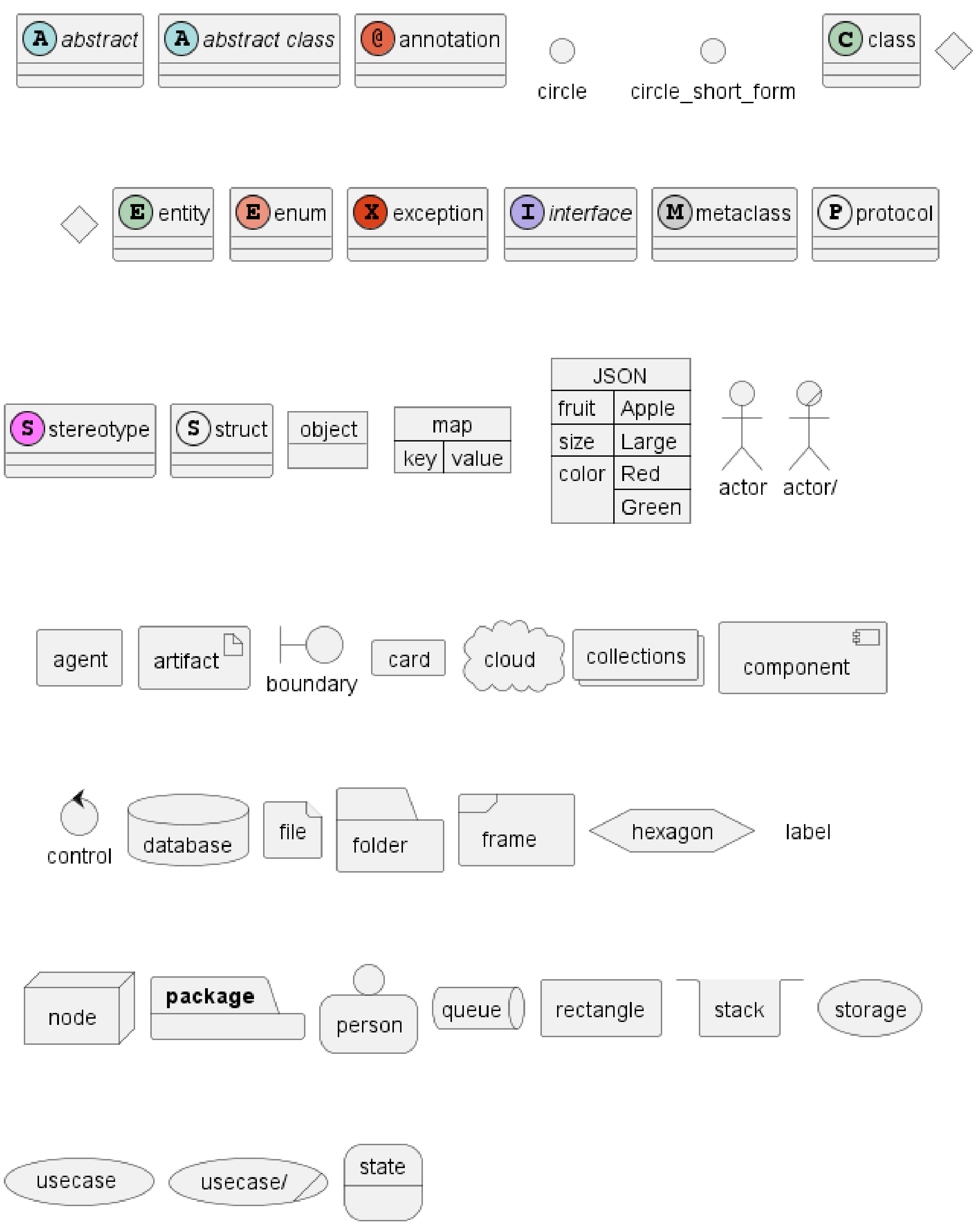
For another example, see on JSON page.

8.18 Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram

In order to add a Deployment element or a State element within a Class or Object diagram, you can use the `allowmixing` or `allow_mixing` directive.

8.18.1 Mixing all elements

```
@startuml

allowmixing

skinparam nodesep 10
abstract        abstract
abstract class  "abstract class"
annotation      annotation
circle          circle
()              circle_short_form
class           class
diamond         diamond
<>              diamond_short_form
entity          entity
enum            enum
exception       exception
interface       interface
metaclass       metaclass
protocol        protocol
stereotype      stereotype
struct          struct
object          object
map map {
 key => value
}
json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
state state
@enduml
```



[Ref. QA-2335 and QA-5329]

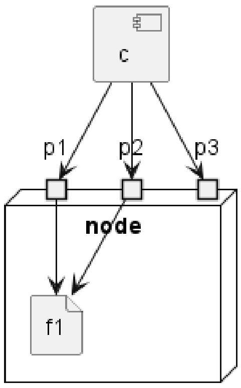
8.19 Port [port, portIn, portOut]

You can added **port** with `port`, `portin`and `portout` keywords.

8.19.1 Port

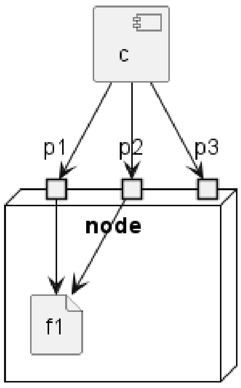
```
@startuml
[c]
node node {
  port p1
  port p2
  port p3
  file f1
}

c --> p1
```

```
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml
```



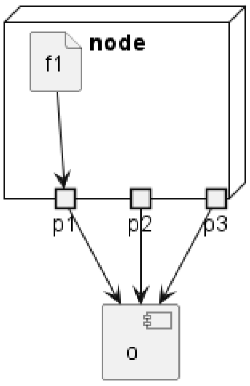
8.19.2 PortIn

```
@startuml
[c]
node node {
  portin p1
  portin p2
  portin p3
  file f1
}

c --> p1
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml
```



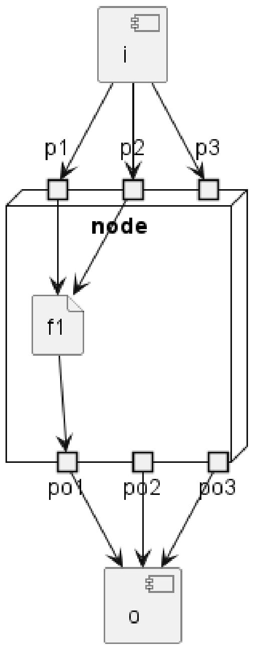
8.19.3 PortOut

```
@startuml
node node {
  portout p1
  portout p2
  portout p3
  file f1
}
[o]
p1 --> o
p2 --> o
p3 --> o
f1 --> p1
@enduml
```



8.19.4 Mixing PortIn & PortOut

```
@startuml
[i]
node node {
  portin p1
  portin p2
  portin p3
  portout po1
  portout po2
  portout po3
  file f1
}
[o]

i --> p1
i --> p2
i --> p3
p1 --> f1
p2 --> f1
po1 --> o
po2 --> o
po3 --> o
f1 --> po1
@enduml
```


i
p1
p2
p3
node
f1
po1
po2
po3
o

9 状态图

状态图被用来对系统的行为进行抽象描述。这种行为被表示为一系列的事件，可以在一个或多个可能的状态下发生。

Using [**PlantUML**](https://plantuml.com/) to create state diagrams offers several advantages:

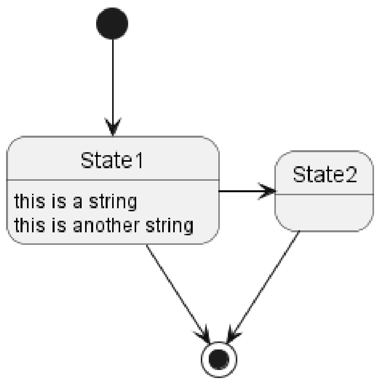
- **Text-Based Language**: Quickly define and visualize the states and transitions without the hassle of manual drawing.
- **Efficiency and Consistency**: Ensure streamlined diagram creation and easy version control.
- **Versatility**: Integrate with various documentation platforms and support multiple output formats.
- **Open-Source & Community Support**: Backed by a [**strong community**](https://forum.plantuml.net/) that continuously contributes to its enhancements and offers invaluable resources.

9.1 普通状态

使用 (`[*]`) 绘制状态图的起点或终点。

使用`-->` 添加箭头。

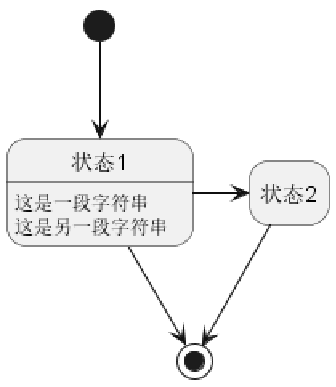
```
@startuml

[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]

@enduml
```



9.2 简化状态

你可以使用隐藏空描述，即 `hide empty description` 关键字，渲染一个简单的状态。

```
@startuml
hide empty description
[*] --> 状态1
状态1 --> [*]
状态1 : 这是一段字符串
状态1 : 这是另一段字符串

状态1 -> 状态2
状态2 --> [*]
@enduml
```

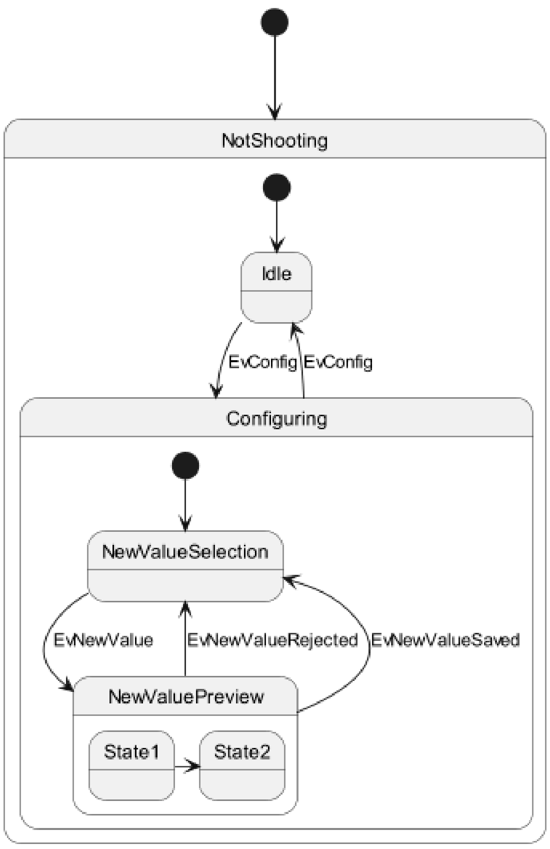


9.3 复杂状态

一个状态也可能是嵌套的，必须使用关键字 `state` 和花括号来定义复杂状态。

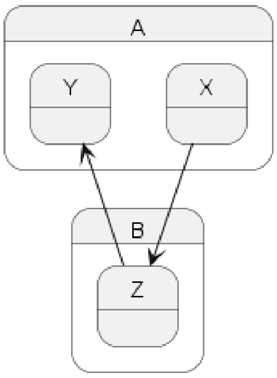
9.3.1 内部子状态

```
@startuml
scale 350 width
[*] --> NotShooting

state NotShooting {
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

state Configuring {
  [*] --> NewValueSelection
  NewValueSelection --> NewValuePreview : EvNewValue
  NewValuePreview --> NewValueSelection : EvNewValueRejected
  NewValuePreview --> NewValueSelection : EvNewValueSaved

  state NewValuePreview {
     State1 -> State2
  }

}
@enduml
```



9.3.2 子状态间的连接

```
@startuml
state A {
  state X {
  }
  state Y {
  }
}

state B {
  state Z {
  }
}

X --> Z
Z --> Y
@enduml
```

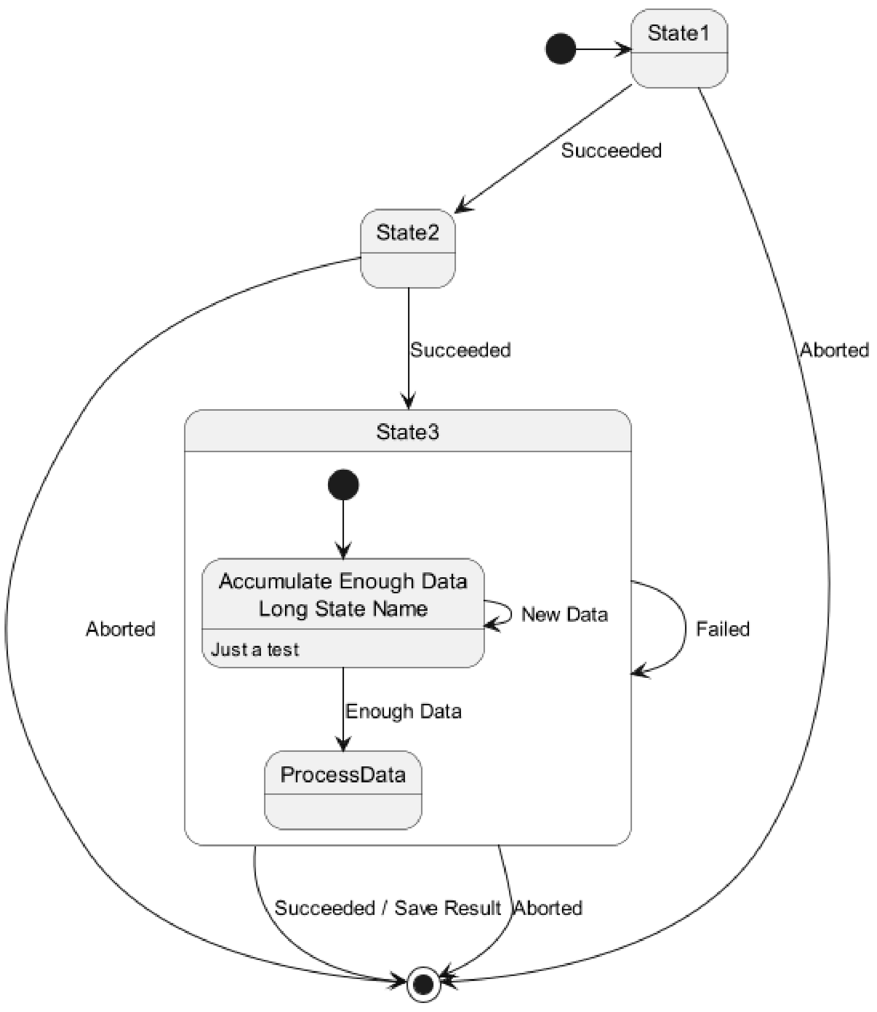


[Ref. QA-3300]

9.4 长状态名

也可以使用关键字 state 来给状态描述较长的状态名，并定义其指代名。

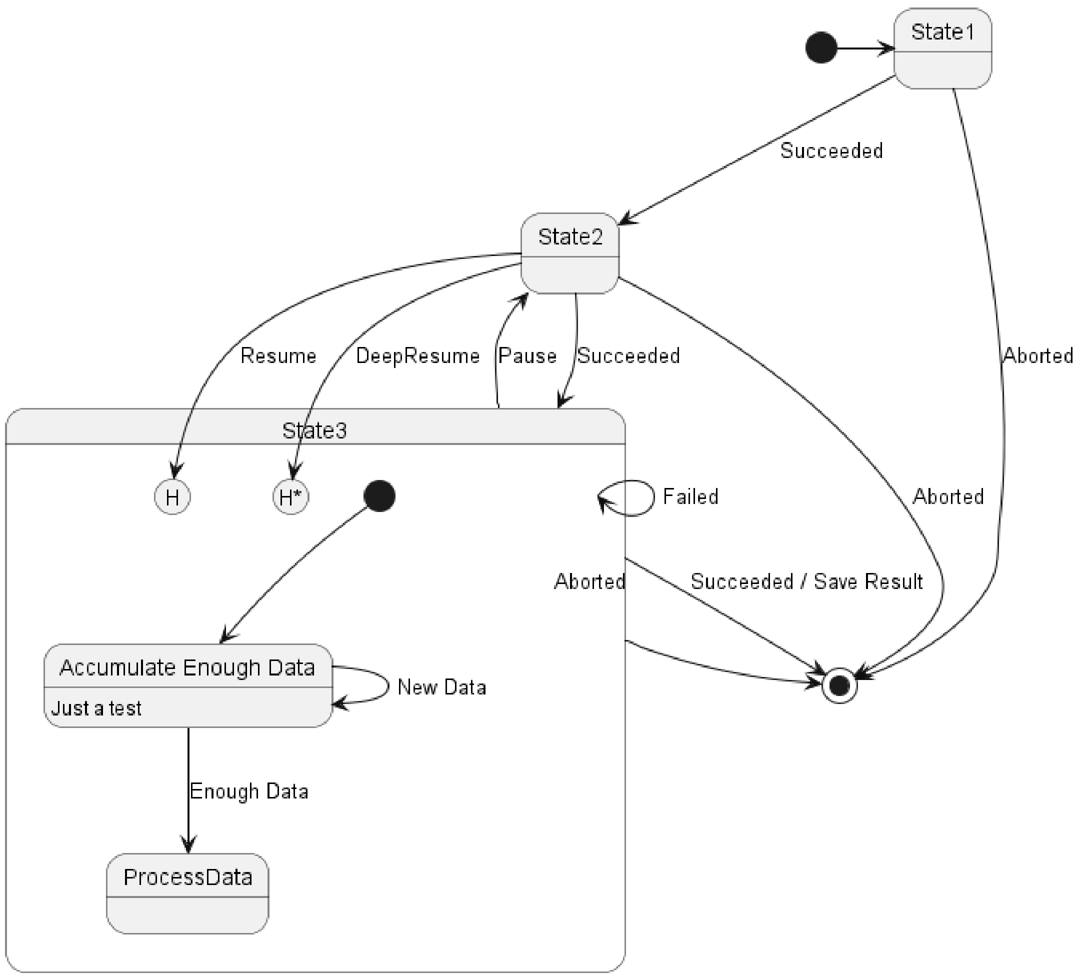
```
@startuml
scale 600 width

[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
  state "Accumulate Enough Data\nLong State Name" as long1
  long1 : Just a test
  [*] --> long1
  long1 --> long1 : New Data
  long1 --> ProcessData : Enough Data
}
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted

@enduml
```



9.5 历史状态 [[H], [H*]]

在嵌套状态中，你可以用 [H] 来表示历史状态，[H*] 表示深层历史状态.

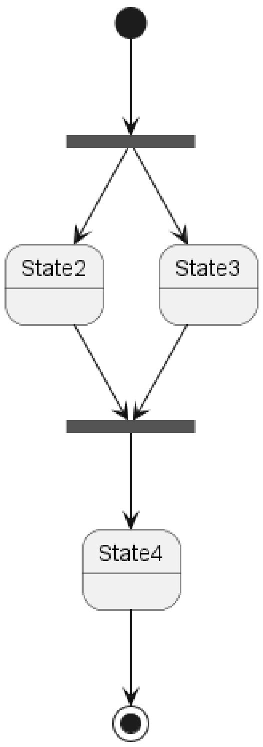
```
@startuml
[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
  state "Accumulate Enough Data" as long1
  long1 : Just a test
  [*] --> long1
  long1 --> long1 : New Data
  long1 --> ProcessData : Enough Data
  State2 --> [H]: Resume
}
State3 --> State2 : Pause
State2 --> State3[H*]: DeepResume
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted
@enduml
```



9.6 分支状态 [fork, join]

你可以使用版型 <<fork>> 和 <<join>> 来表示状态的分叉及合并。

```
@startuml

state fork_state <<fork>>
[*] --> fork_state
fork_state --> State2
fork_state --> State3

state join_state <<join>>
State2 --> join_state
State3 --> join_state
join_state --> State4
State4 --> [*]

@enduml
```

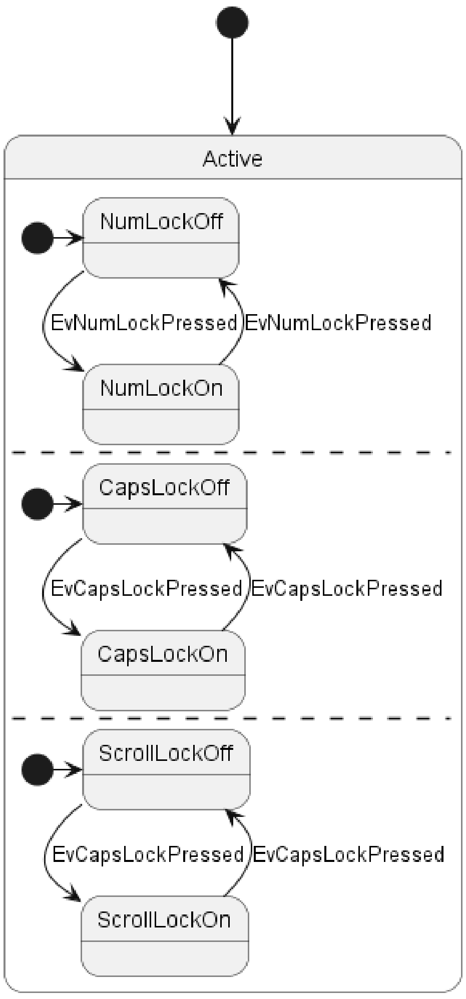


9.7 并发状态 [–, ||]

用-- or || 作为分隔符来合成并发状态。

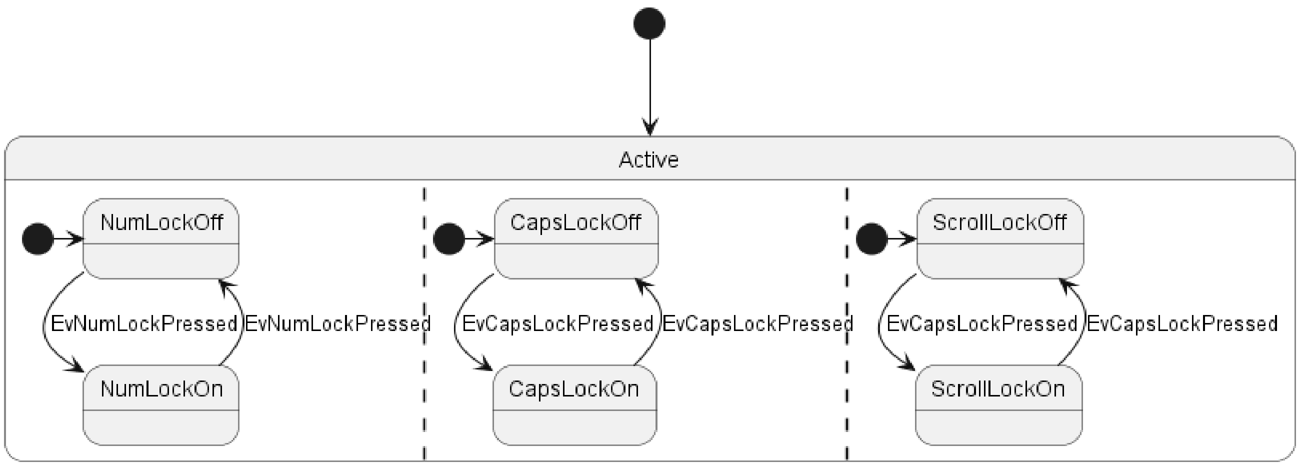
9.7.1 水平分隔 --

```
@startuml
[*] --> Active

state Active {
  [*] -> NumLockOff
  NumLockOff --> NumLockOn : EvNumLockPressed
  NumLockOn --> NumLockOff : EvNumLockPressed
  --
  [*] -> CapsLockOff
  CapsLockOff --> CapsLockOn : EvCapsLockPressed
  CapsLockOn --> CapsLockOff : EvCapsLockPressed
  --
  [*] -> ScrollLockOff
  ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
  ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml
```



9.7.2 竖直分隔 ||

```
@startuml
[*] --> Active

state Active {
  [*] -> NumLockOff
  NumLockOff --> NumLockOn : EvNumLockPressed
  NumLockOn --> NumLockOff : EvNumLockPressed
  ||
  [*] -> CapsLockOff
  CapsLockOff --> CapsLockOn : EvCapsLockPressed
  CapsLockOn --> CapsLockOff : EvCapsLockPressed
  ||
  [*] -> ScrollLockOff
  ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
  ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml
```

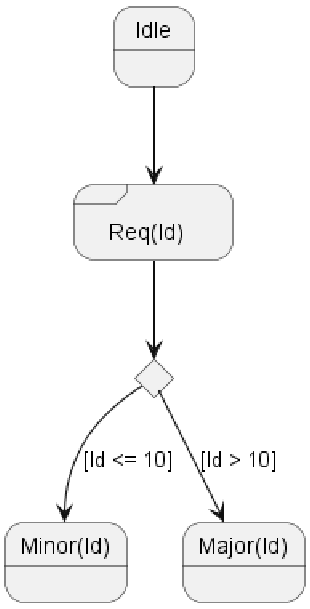


[Ref. [QA-3086](https://forum.plantuml.net/3086/state-diagram-concurrent-state-horizontal-line)]

9.8 选择结点 [choice]

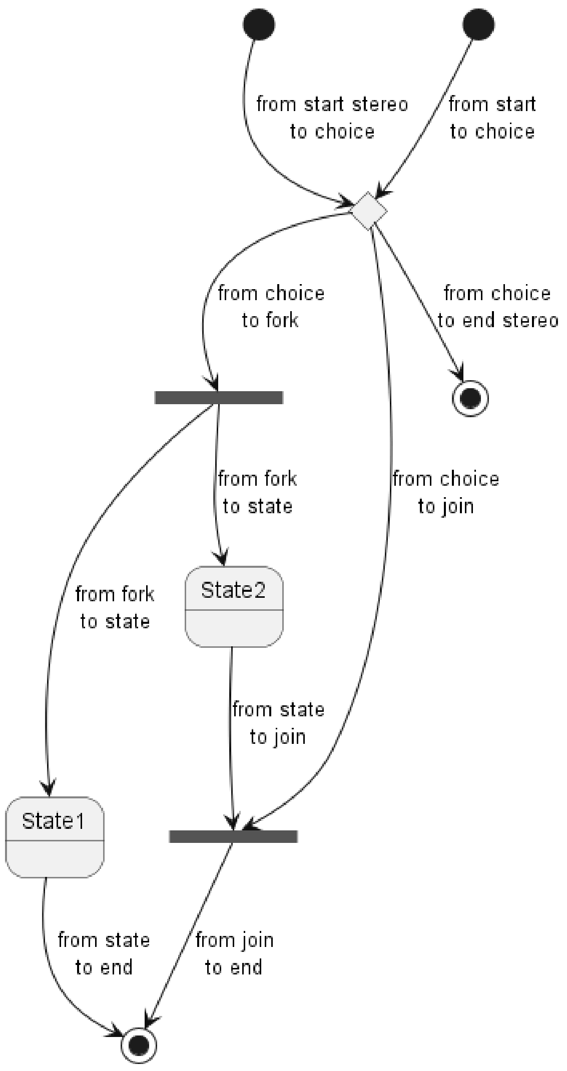
版型 `<<choice>>` 可以用来表示一个选择结点，表示状态条件。

```
@startuml
state "Req(Id)" as ReqId <<sdlreceive>>
state "Minor(Id)" as MinorId
state "Major(Id)" as MajorId

state c <<choice>>

Idle --> ReqId
ReqId --> c
c --> MinorId : [Id <= 10]
c --> MajorId : [Id > 10]
@enduml
```



9.9 一个使用版型的完整样例 [start, choice, fork, join, end]

```
@startuml
state start1  <<start>>
state choice1 <<choice>>
state fork1   <<fork>>
state join2   <<join>>
state end3    <<end>>

[*]     --> choice1 : from start\nto choice
start1  --> choice1 : from start stereo\nto choice

choice1 --> fork1   : from choice\nto fork
choice1 --> join2   : from choice\nto join
choice1 --> end3    : from choice\nto end stereo

fork1   ---> State1 : from fork\nto state
fork1   --> State2  : from fork\nto state

State2  --> join2   : from state\nto join
State1  --> [*]     : from state\nto end

join2   --> [*]     : from join\nto end
@enduml
```

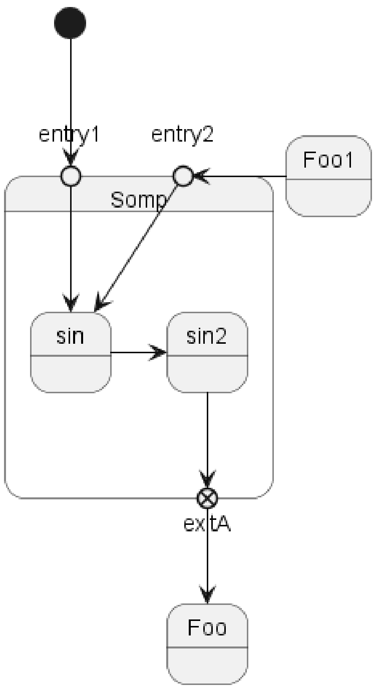


[参考 QA-404,QA-1159 和 GH-887]

9.10 入口和出口 [entryPoint, exitPoint]

你可以用以下版型给合成状态添加入口结点 `<<entryPoint>>` 和出口结点 `<<exitPoint>>`。

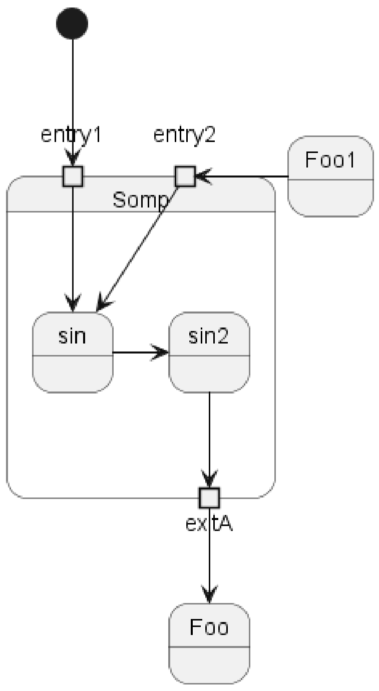
```
@startuml
state Somp {
  state entry1 <<entryPoint>>
  state entry2 <<entryPoint>>
  state sin
  entry1 --> sin
  entry2 -> sin
  sin -> sin2
  sin2 --> exitA <<exitPoint>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml
```



9.11 引脚 [inputPin, outputPin]

你可以用以下版型添加引脚结点 `<<inputPin>>` 和 `<<outputPin>>`。

```
@startuml
state Somp {
  state entry1 <<inputPin>>
  state entry2 <<inputPin>>
  state sin
  entry1 --> sin
  entry2 -> sin
  sin -> sin2
  sin2 --> exitA <<outputPin>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml
```

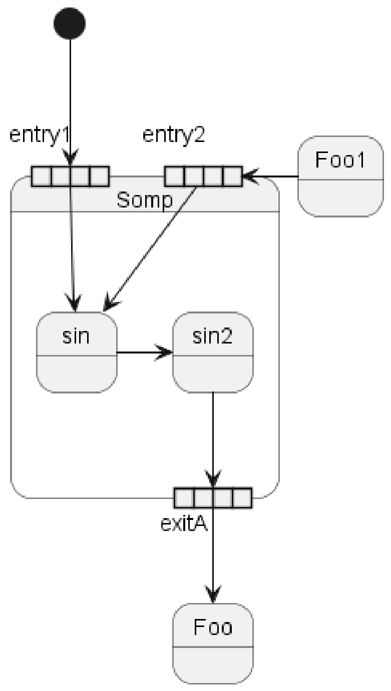


[Ref. QA-4309]

9.12 扩展 [expansionInput, expansionOutput]

你可以用以下版型添加扩展结点 `<<expansionInput>>` 和 `<<expansionOutput>>`。

```
@startuml
state Somp {
  state entry1 <<expansionInput>>
  state entry2 <<expansionInput>>
  state sin
  entry1 --> sin
  entry2 -> sin
  sin -> sin2
  sin2 --> exitA <<expansionOutput>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml
```



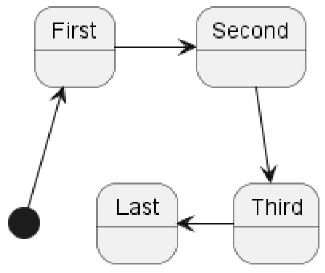
[Ref. QA-4309]

9.13 箭头方向

使用-> 定义水平箭头，也可以使用下列格式强制设置箭头方向：

- `-down->` (default arrow)
- `-right->` or `->`
- `-left->`
- `-up->`

```
@startuml

[*] -up-> First
First -right-> Second
Second --> Third
Third -left-> Last

@enduml
```



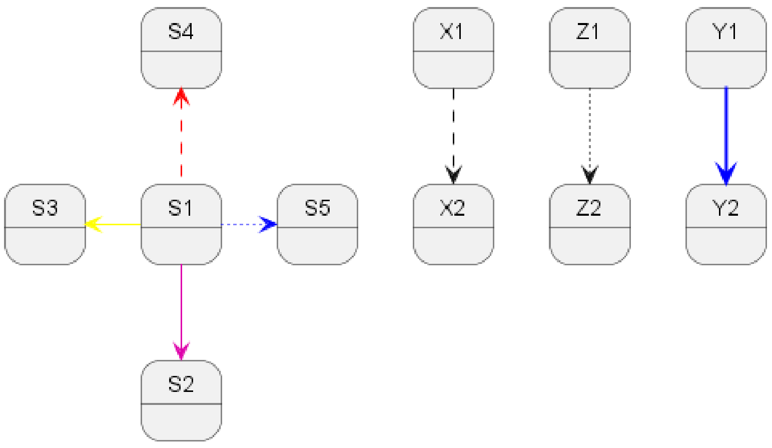
可以用首字母缩写或者开始的两个字母定义方向 (如, `-d-`，`-down-`和`-do-`是完全等价的)。

请不要滥用这些功能，*Graphviz* 不喜欢这样。

9.14 更改箭头线条的颜色和风格

你可以更改线条的颜色及风格.

```
@startuml
State S1
State S2
S1 -[#DD00AA]-> S2
S1 -left[#yellow]-> S3
S1 -up[#red,dashed]-> S4
S1 -right[dotted,#blue]-> S5

X1 -[dashed]-> X2
Z1 -[dotted]-> Z2
Y1 -[#blue,bold]-> Y2
@enduml
```



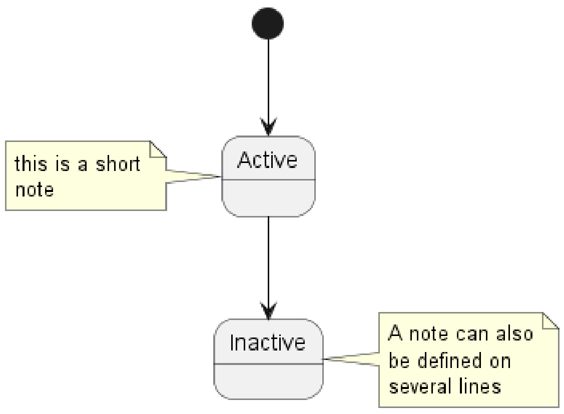
[Ref. Incubation: Change line color in state diagrams]

9.15 注释

可以用 `note left of`, `note right of`, `note top of`, `note bottom of` 关键字来定义注释。

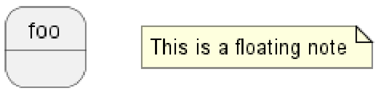
还可以定义多行注释。

```
@startuml

[*] --> Active
Active --> Inactive

note left of Active : this is a short\nnote

note right of Inactive
  A note can also
  be defined on
  several lines
end note

@enduml
```



以及浮动注释。

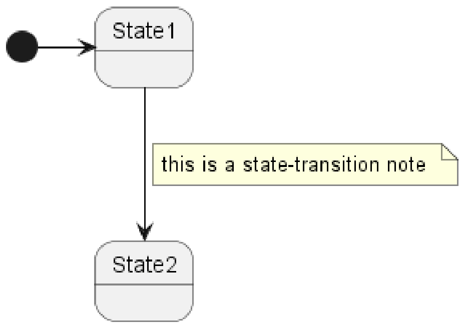
```
@startuml

state foo
note "This is a floating note" as N1

@enduml
```



9.16 在箭头上添加注释

你可以在连接箭头上使用 `note on link` 关键字来添加注释.

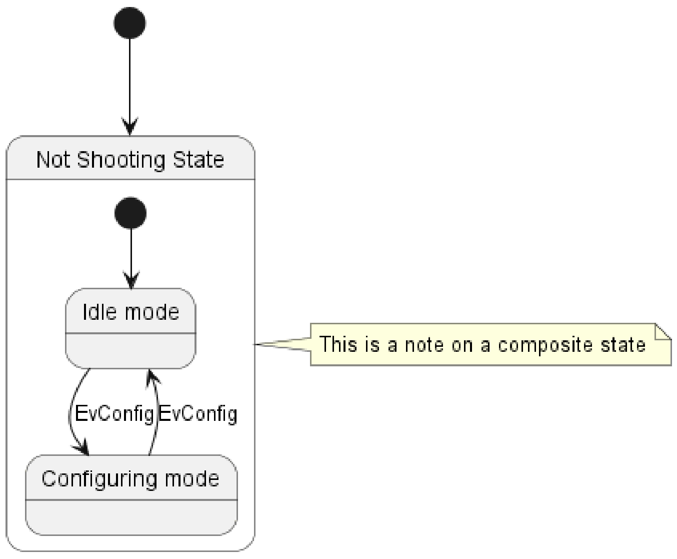
```
@startuml
[*] -> State1
State1 --> State2
note on link
  this is a state-transition note
end note
@enduml
```



9.17 给复杂状态添加注释

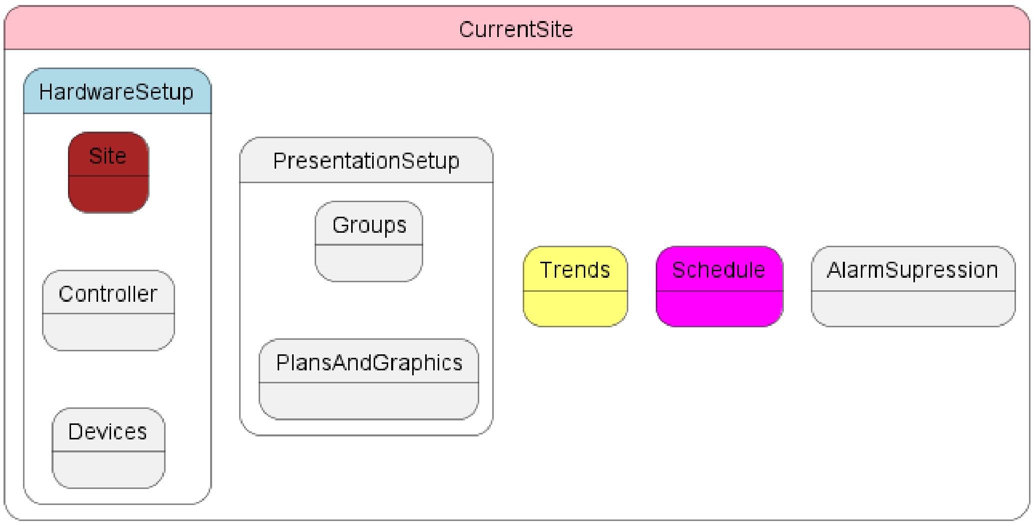
可以给嵌套的状态添加注释。

```
@startuml

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
  state "Idle mode" as Idle
  state "Configuring mode" as Configuring
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

note right of NotShooting : This is a note on a composite state

@enduml
```



9.18 颜色

```
@startuml
state CurrentSite #pink {
    state HardwareSetup #lightblue {
       state Site #brown
        Site -[hidden]-> Controller
        Controller -[hidden]-> Devices
    }
    state PresentationSetup{
        Groups -[hidden]-> PlansAndGraphics
    }
    state Trends #FFFF77
    state Schedule #magenta
    state AlarmSupression
}
@enduml
```



[参考 QA-1812]

9.19 显示参数

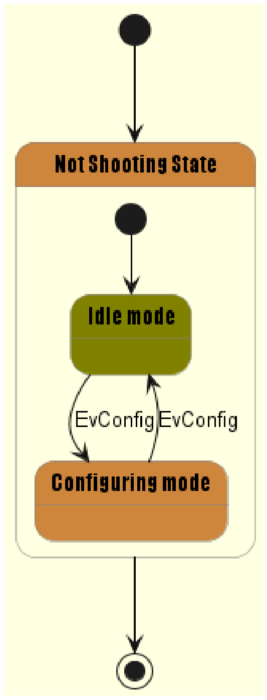
用 skinparam 改变字体和颜色。

可以在如下场景中使用：

- 在图示的定义中，
- 在导入的文件中，
- 在命令行或者 ANT 任务提供的配置文件中。

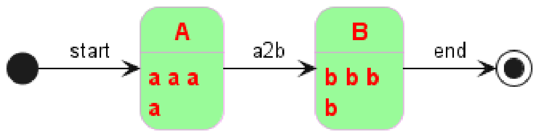
还可以为状态的构造类型指定特殊的字体和颜色。

```
@startuml
skinparam backgroundColor LightYellow
skinparam state {
  StartColor MediumBlue
  EndColor Red
  BackgroundColor Peru
  BackgroundColor<<Warning>> Olive
  BorderColor Gray
  FontName Impact
}

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
  state "Idle mode" as Idle <<Warning>>
  state "Configuring mode" as Configuring
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

NotShooting --> [*]
@enduml
```



9.19.1 状态图所有显示参数测试

```
@startuml
skinparam State {
  AttributeFontColor blue
  AttributeFontName serif
  AttributeFontSize  9
  AttributeFontStyle italic
  BackgroundColor palegreen
  BorderColor violet
  EndColor gold
  FontColor red
  FontName Sanserif
  FontSize 15
  FontStyle bold
  StartColor silver
}

state A : a a a\na
state B : b b b\nb

[*] -> A  : start
A -> B : a2b
B -> [*] : end
@enduml
```

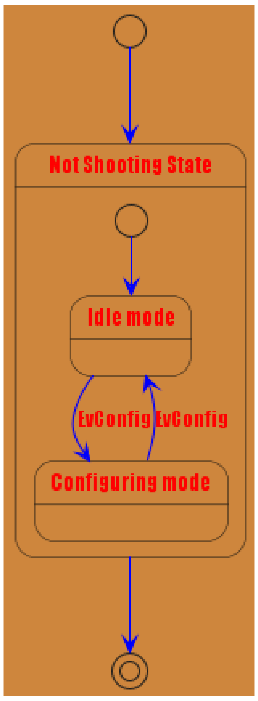


9.20 Changing style

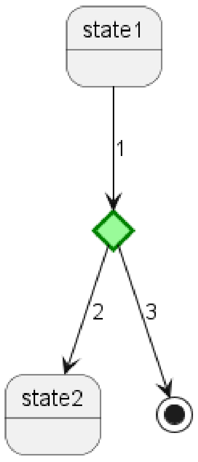
You can change style.

```
@startuml
```

```
<style>
stateDiagram {
  BackgroundColor Peru
  'LineColor Gray
  FontName Impact
  FontColor Red
  arrow {
    FontSize 13
    LineColor Blue
  }
}
</style>


[*] --> NotShooting

state "Not Shooting State" as NotShooting {
  state "Idle mode" as Idle <<Warning>>
  state "Configuring mode" as Configuring
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

NotShooting --> [*]
@enduml
```



```
@startuml
<style>
  diamond {
    BackgroundColor #palegreen
    LineColor #green
    LineThickness 2.5
}
</style>
state state1
state state2
state choice1 <<choice>>
state end3    <<end>>

state1  --> choice1 : 1
choice1 --> state2  : 2
choice1 --> end3    : 3
@enduml
```



[Ref. GH-880]

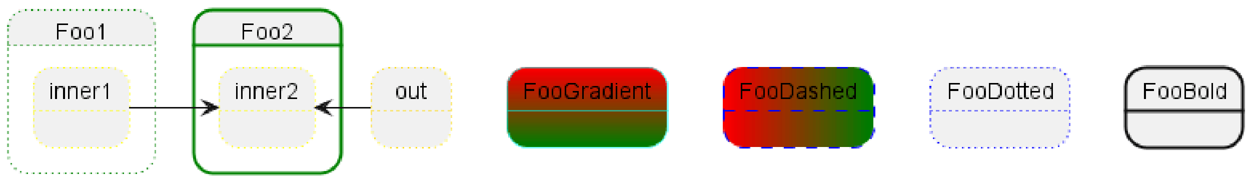
9.21 Change state color and style (inline style)

You can change the color or style of individual state using the following notation:

- `#color ##[style]color`

With background color first (`#color`), then line style and line color (`##[style]color`).

```
@startuml
state FooGradient #red-green ##00FFFF
state FooDashed #red|green ##[dashed]blue {
}
state FooDotted ##[dotted]blue {
}
state FooBold ##[bold] {
}
state Foo1 ##[dotted]green {
state inner1 ##[dotted]yellow
}

state out ##[dotted]gold

state Foo2 ##[bold]green {
state inner2 ##[dotted]yellow
}
inner1 -> inner2
out -> inner2
@enduml
```

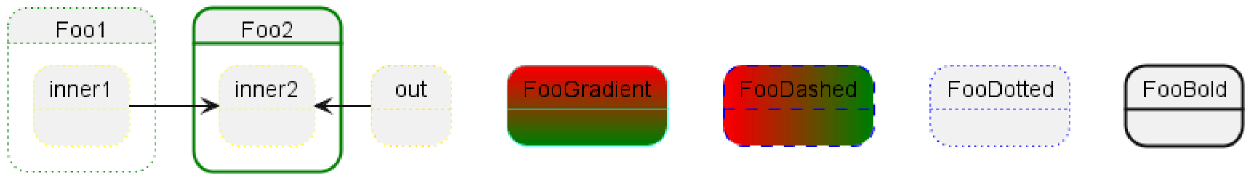


[Ref. QA-1487]

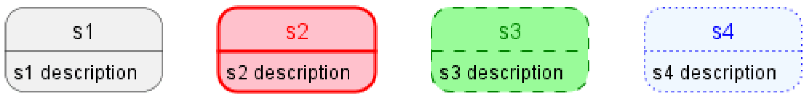
- `#color;line:color;line.[bold|dashed|dotted];text:color`

TODO: FIXME `text:color` seems not to be taken into account **TODO:** FIXME

```
@startuml
@startuml
state FooGradient #red-green;line:00FFFF
state FooDashed #red|green;line.dashed;line:blue {
}
state FooDotted #line.dotted;line:blue {
}
state FooBold #line.bold {
}
state Foo1 #line.dotted;line:green {
state inner1 #line.dotted;line:yellow
}

state out #line.dotted;line:gold

state Foo2 #line.bold;line:green {
state inner2 #line.dotted;line:yellow
}
inner1 -> inner2
out -> inner2
@enduml
@enduml
```



```
@startuml
state s1 : s1 description
state s2 #pink;line:red;line.bold;text:red : s2 description
state s3 #palegreen;line:green;line.dashed;text:green : s3 description
state s4 #aliceblue;line:blue;line.dotted;text:blue   : s4 description
@enduml
```

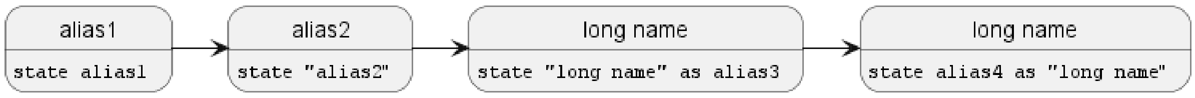


[Adapted from QA-3770]

9.22 别名

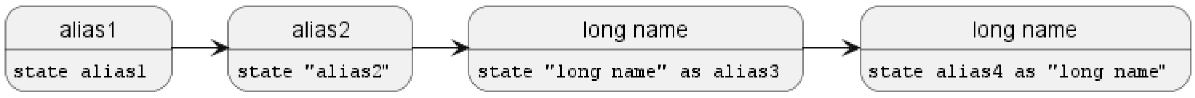
有了 State，你可以使用 `alias`，比如。

```
@startuml
state alias1
state "alias2"
state "long name" as alias3
state alias4 as "long name"

alias1 : ""state alias1""
alias2 : ""state "alias2"""
alias3 : ""state "long name" as alias3""
alias4 : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml
```



或。

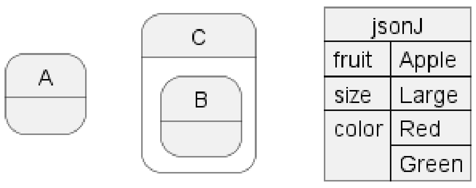
```
@startuml
state alias1 : ""state alias1""
state "alias2" : ""state "alias2"""
state "long name" as alias3 : ""state "long name" as alias3""
state alias4 as "long name" : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml
```



9.23 Display JSON Data on State diagram

9.23.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
 state B
}

json jsonJ {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-17275]

For another example, see on JSON page.

10 时序图

这仍然是在建设中。如果你需要一些新的功能，你可以提出。

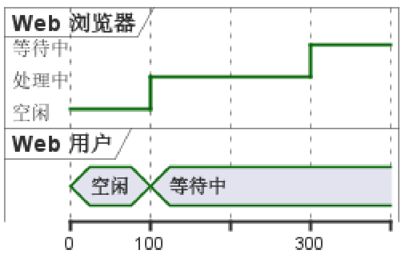
10.1 声明参与者

使用以下关键字声明参与者, 可根据需要选择显示样式。

关键词	描述
analog	模拟信号的变化是连续的，其值在给定值之间会进行线性插值。
binary	二进制信号，只有两个状态。(binary)
clock	时钟信号，即从高电平到低电平反复转换的信号，需要用 period 设置周期。可选的参数还有脉冲时长 pul
concise	简明的图形化数据表示，可表示数据的移动（非常适合表示信息）。
robust	用信号线表示的状态，便于呈现状态间的转化（可设置多种状态）。

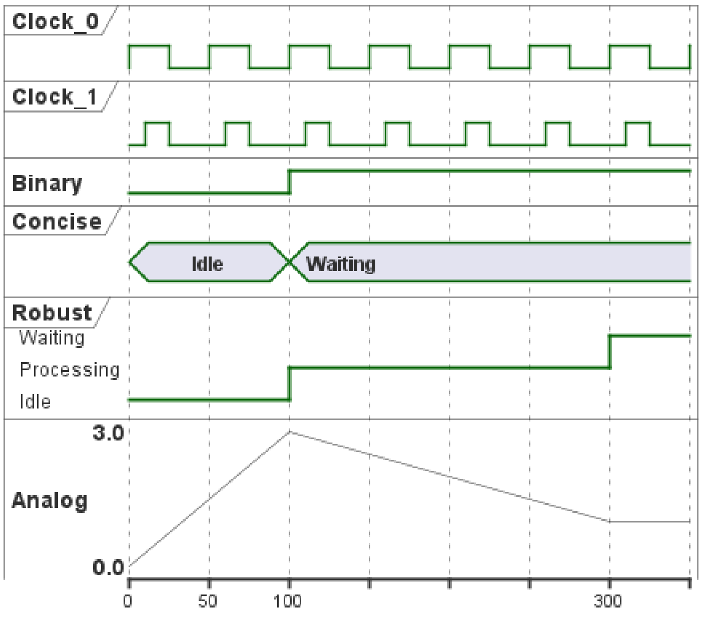
通过 @ 标注, 和 is 动词定义状态.

```
@startuml
robust "Web 浏览器" as WB
concise "Web 用户" as WU

@0
WU is 空闲
WB is 空闲

@100
WU is 等待中
WB is 处理中

@300
WB is 等待中
@enduml
```



```
@startuml
clock   "Clock_0"   as C0 with period 50
clock   "Clock_1"   as C1 with period 50 pulse 15 offset 10
binary  "Binary"  as B
concise "Concise" as C
robust  "Robust"  as R
analog  "Analog"  as A


@0
C is Idle
R is Idle
A is 0

@100
B is high
C is Waiting
R is Processing
```

```
A is 3

@300
R is Waiting
A is 1
@enduml
```



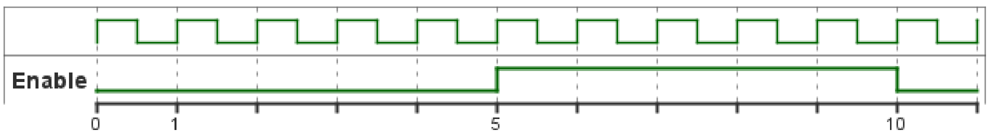
[Ref. QA-14631 and QA-14647 and QA-11288]

10.2 二进制及时钟信号

使用以下关键字可以绘制二进制及时钟信号：

- `binary`
- `clock`

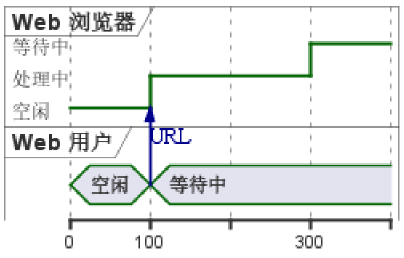
```
@startuml
clock clk with period 1
binary "Enable" as EN

@0
EN is low

@5
EN is high

@10
EN is low
@enduml
```



10.3 增加标示

使用下述的语法，增加对某一时刻信号变动的描述。

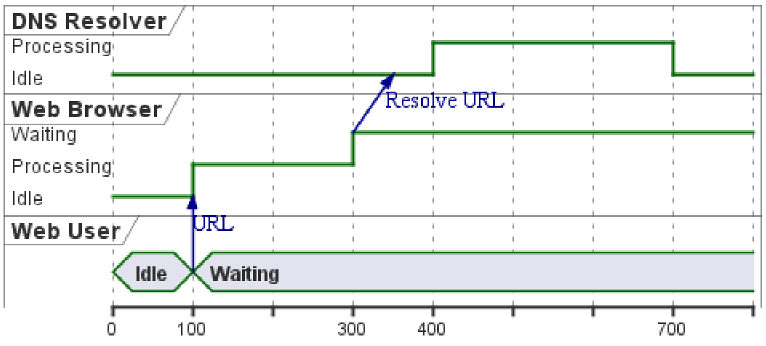
```
@startuml
robust "Web 浏览器" as WB
concise "Web 用户" as WU

@0
WU is 空闲
WB is 空闲

@100
WU -> WB : URL
WU is 等待中
WB is 处理中

@300
WB is 等待中
@enduml
@enduml
```



10.4 相对时间

可以使用 @+ 和 @- 符号表示相对时间.

```
@startuml
robust "DNS Resolver" as DNS
robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle
DNS is Idle

@+100
WU -> WB : URL
WU is Waiting
WB is Processing

@+200
WB is Waiting
WB -> DNS@+50 : Resolve URL

@+100
DNS is Processing

@+300
DNS is Idle
@enduml
```

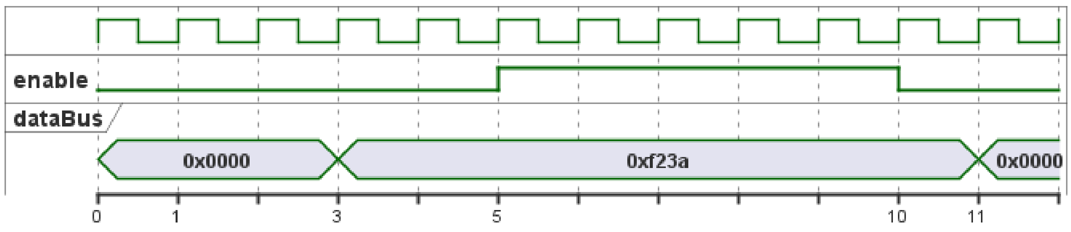


10.5 锚点

，而不是在绝对时间上使用绝对时间或相对时间，你可以通过使用 as 关键字和以：开始的名称来定义一个时间作为锚点。

```
@XX as :<anchor point name>
```

```
@startuml
clock clk with period 1
binary "enable" as EN
concise "dataBus" as db

@0 as :start
@5 as :en_high
@10 as :en_low
@:en_high-2 as :en_highMinus2

@:start
EN is low
db is "0x0000"

@:en_high
EN is high

@:en_low
EN is low

@:en_highMinus2
db is "0xf23a"

@:en_high+6
db is "0x0000"
@enduml
```

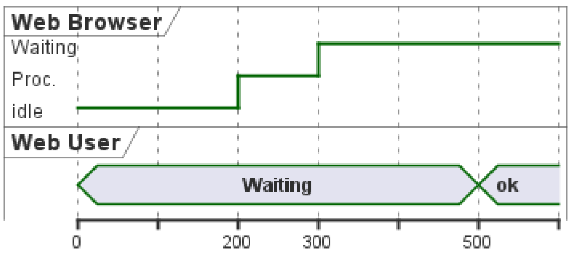


10.6 参与者

按时间顺序来描述状态变化可能会有些麻烦，不如将每个参与者的变化放在一起。

```
@startuml
```

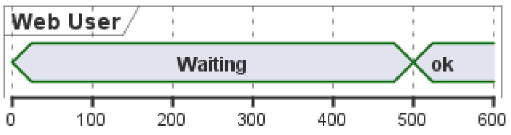
```
robust "Web Browser" as WB
concise "Web User" as WU

@WB
0 is idle
+200 is Proc.
+100 is Waiting

@WU
0 is Waiting
+500 is ok
@enduml
```



10.7 设置缩放

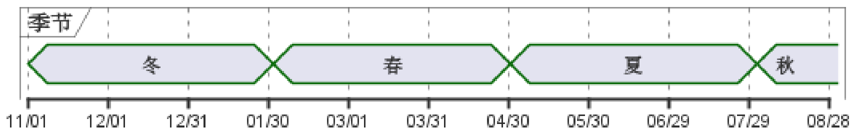
你还可以设置缩放比例。

```
@startuml
concise "Web User" as WU
scale 100 as 50 pixels

@WU
0 is Waiting
+500 is ok
@enduml
```



当使用绝对时间或者日期时，1 单位为 1 秒。

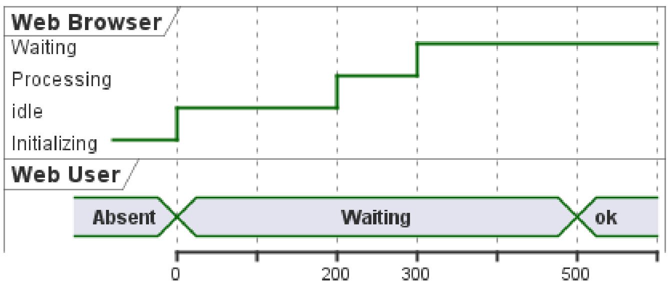
```
@startuml
concise "季节" as S
'这里将30天缩放到50像素
scale 2592000 as 50 pixels

@2000/11/01
S is "冬"

@2001/02/01
S is "春"

@2001/05/01
S is "夏"

@2001/08/01
S is "秋"
@enduml
```



10.8 初始状态

你可以定义初始状态。

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
+100 is Waiting

@WU
0 is Waiting
+500 is ok
@enduml
```



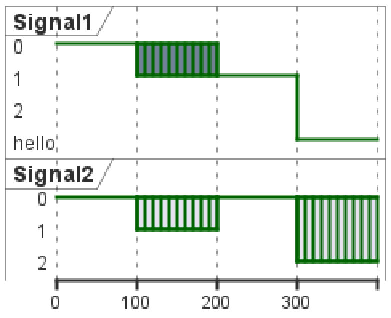
10.9 模糊状态

一个信号在某个时段可能是模糊的，有多个可能的状态区间。

10.9.1 模糊或者未定义鲁棒状态信号

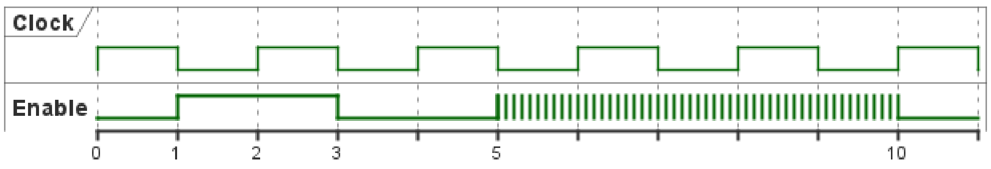
```
@startuml
robust "Signal1" as S1
robust "Signal2" as S2
S1 has 0,1,2,hello
S2 has 0,1,2
@0
S1 is 0
S2 is 0
@100
S1 is {0,1} #SlateGrey
S2 is {0,1}
@200
S1 is 1
S2 is 0
@300
S1 is hello
```

```
S2 is {0,2}
@enduml
```



10.9.2 模糊或者未定义二进制信号

```
@startuml
clock "Clock" as C with period 2
binary "Enable" as EN

@0
EN is low
@1
EN is high
@3
EN is low
@5
EN is {low,high}
@10
EN is low
@enduml
```



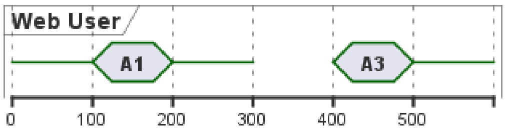
[Ref. QA-11936 and QA-15933]

10.10 隐藏状态

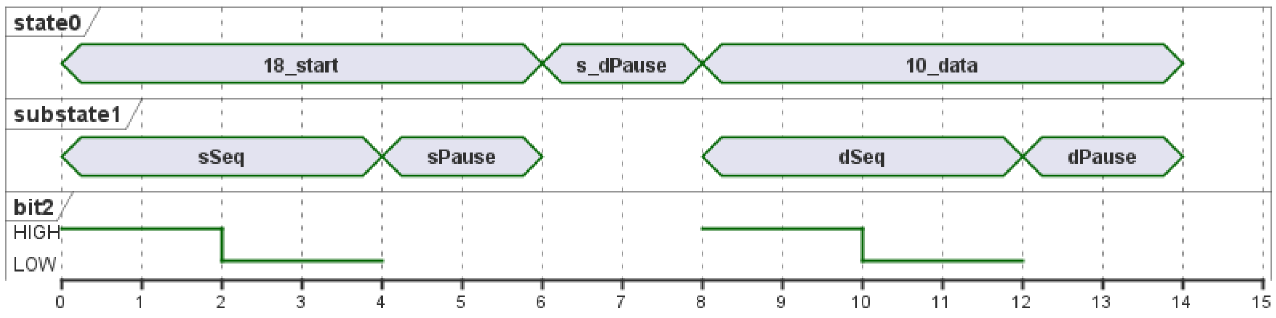
可以隐藏某个时段的状态。

```
@startuml
concise "Web User" as WU

@0
WU is {-}

@100
WU is A1

@200
WU is {-}

@300
WU is {hidden}
```

```
@400
WU is A3

@500
WU is {-}
@enduml
```



```
@startuml
scale 1 as 50 pixels

concise state0
concise substate1
robust bit2

bit2 has HIGH,LOW

@state0
0 is 18_start
6 is s_dPause
8 is 10_data
14 is {hidden}

@substate1
0 is sSeq
4 is sPause
6 is {hidden}
8 is dSeq
12 is dPause
14 is {hidden}

@bit2
0 is HIGH
2 is LOW
4 is {hidden}
8 is HIGH
10 is LOW
12 is {hidden}
@enduml
```

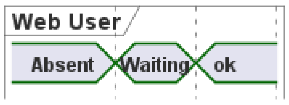


[Ref. QA-12222]

10.11 隐藏时间轴

可以隐藏时间轴。

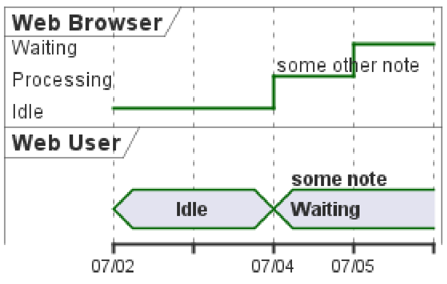
```
@startuml
hide time-axis
concise "Web User" as WU

WU is Absent

@WU
0 is Waiting
+500 is ok
@enduml
```



10.12 使用日期

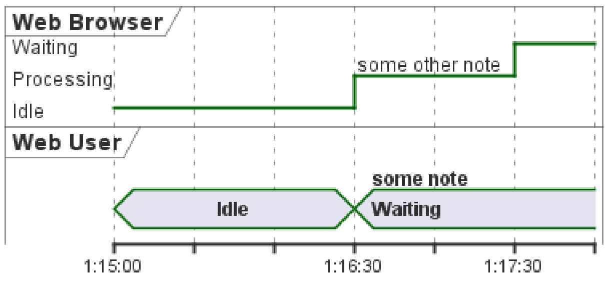
时间轴除时间以外也可以改用日期表示。

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@2019/07/02
WU is Idle
WB is Idle

@2019/07/04
WU is Waiting : some note
WB is Processing : some other note

@2019/07/05
WB is Waiting
@enduml
```



```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@1:15:00
WU is Idle
WB is Idle

@1:16:30
WU is Waiting : some note
WB is Processing : some other note
```

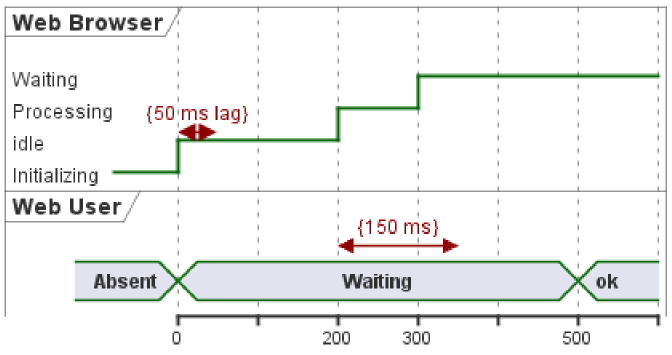
```
@1:17:30
WB is Waiting
@enduml
```



10.13 添加约束

可以在图上标示时间约束。

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}

@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```

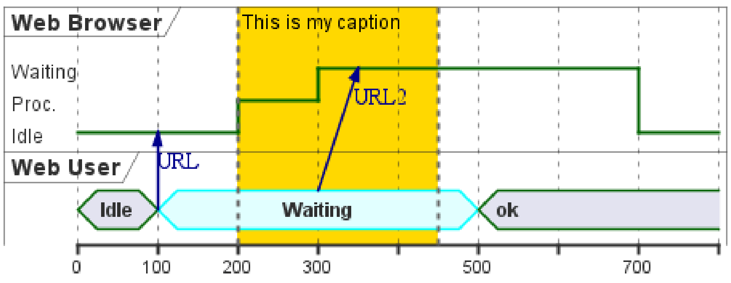


10.13.1 = 设置高亮

你可以给图表中的某一时段设置高亮。

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@0
```

```
WU is Idle
WB is Idle

@100
WU -> WB : URL
WU is Waiting #LightCyan;line:Aqua

@200
WB is Proc.

@300
WU -> WB@350 : URL2
WB is Waiting

@+200
WU is ok

@+200
WB is Idle

highlight 200 to 450 #Gold;line:DimGrey : This is my caption
@enduml
```

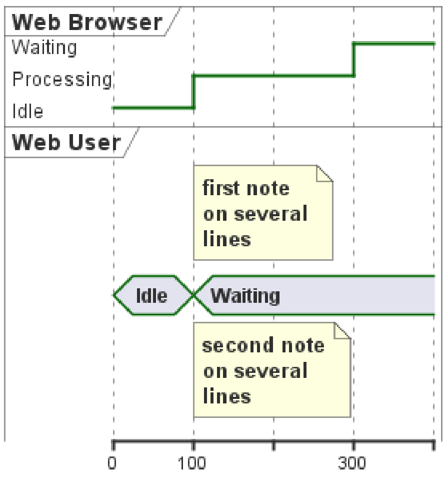


[Ref. [QA-10868](https://forum.plantuml.net/10868/highlighted-periods-in-timing-diagrams)]

10.14 添加注释

你可以使用 `note top of` 或 `note bottom of` 关键字在某一时刻或参与者的上方或下方添加注释（只可以在 `concise` 类型的参与者中使用）。

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing
note top of WU : first note\non several\nlines
note bottom of WU : second note\non several\nlines

@300
WB is Waiting
@enduml
```



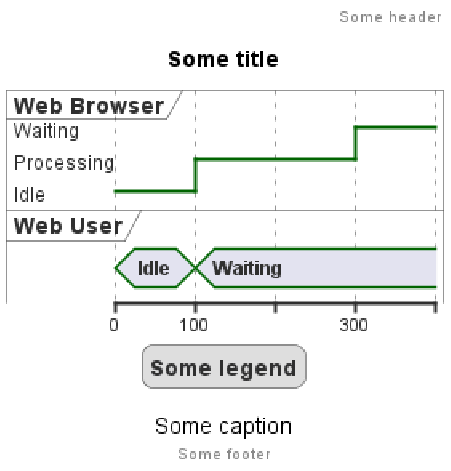
[Ref. QA-6877]

*[Ref. [QA-6877](https:*forum.plantuml.net/6877), [GH-1465](https:*github.com/plantuml/plantuml/issues/1465)]*

10.15 添加文本

你可以选择添加标题、页眉、页脚、图例和说明。

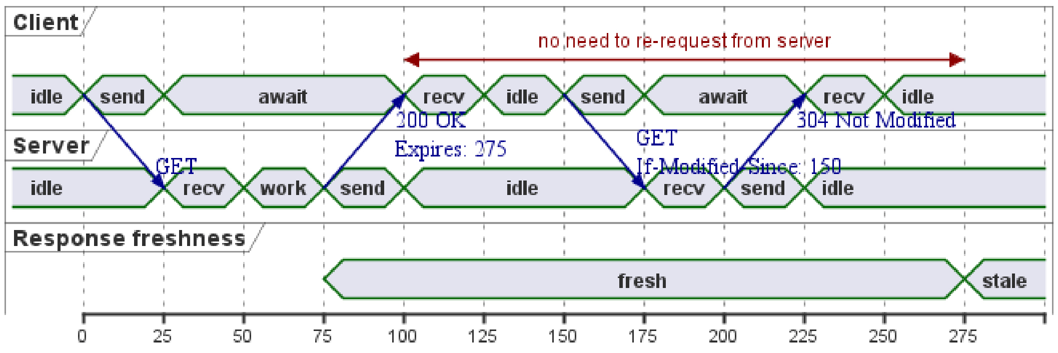
```
@startuml
Title Some title
header: Some header
footer: Some footer
legend
Some legend
end legend
caption Some caption

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting
@enduml
```



10.16 完整样例

感谢 Adam Rosien 提供该样例。

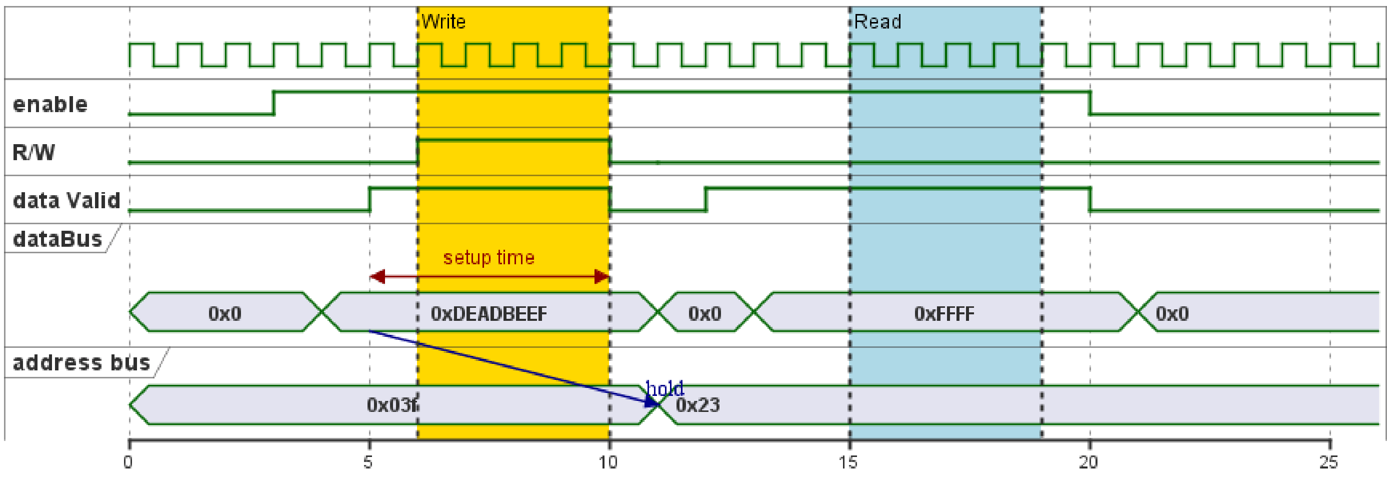
```
@startuml
concise "Client" as Client
concise "Server" as Server
concise "Response freshness" as Cache

Server is idle
Client is idle

@Client
0 is send
Client -> Server@+25 : GET
+25 is await
+75 is recv
+25 is idle
+25 is send
Client -> Server@+25 : GET\nIf-Modified-Since: 150
+25 is await
+50 is recv
+25 is idle
@100 <-> @275 : no need to re-request from server

@Server
25 is recv
+25 is work
+25 is send
Server -> Client@+25 : 200 OK\nExpires: 275
+25 is idle
+75 is recv
+25 is send
Server -> Client@+25 : 304 Not Modified
+25 is idle

@Cache
75 is fresh
+200 is stale
@enduml
```



10.17 数据表示样例

```
@startuml
scale 5 as 150 pixels

clock clk with period 1
binary "enable" as en
binary "R/W" as rw
binary "data Valid" as dv
concise "dataBus" as db
concise "address bus" as addr

@6 as :write_beg
@10 as :write_end

@15 as :read_beg
@19 as :read_end


@0
en is low
db is "0x0"
addr is "0x03f"
rw is low
dv is 0

@:write_beg-3
 en is high
@:write_beg-2
 db is "0xDEADBEEF"
@:write_beg-1
dv is 1
@:write_beg
rw is high


@:write_end
rw is low
dv is low
@:write_end+1
rw is low
db is "0x0"
addr is "0x23"

@12
dv is high
```

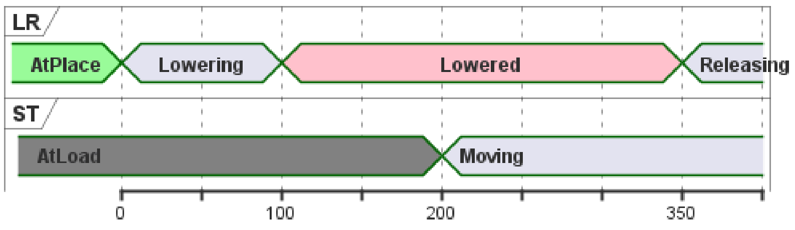
```
@13
db is "0xFFFF"

@20
en is low
dv is low
@21
db is "0x0"

highlight :write_beg to :write_end #Gold:Write
highlight :read_beg to :read_end #lightBlue:Read

db@:write_beg-1 <-> @:write_end : setup time
db@:write_beg-1 -> addr@:write_end+1 : hold
@enduml
```



10.18 颜色

你可以为图表添加颜色。

```
@startuml
concise "LR" as LR
concise "ST" as ST

LR is AtPlace #palegreen
ST is AtLoad #gray

@LR
0 is Lowering
100 is Lowered #pink
350 is Releasing

@ST
200 is Moving
@enduml
```

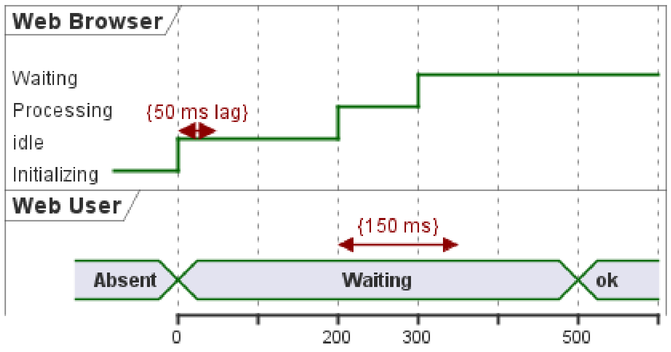


[Ref.QA-5776]

10.19 使用（全局）样式

10.19.1 无样式（默认样式）。

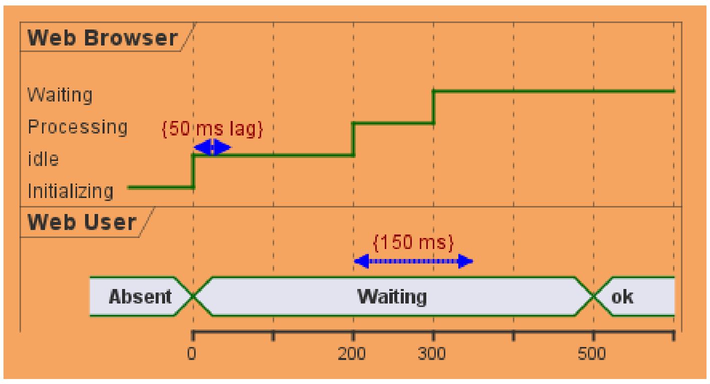
```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}

@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



10.19.2 设置样式

你可以使用样式来改变元素的渲染。

```
@startuml
<style>
timingDiagram {
  document {
    BackGroundColor SandyBrown
  }
 constraintArrow {
  LineStyle 2-1
  LineThickness 3
  LineColor Blue
 }
}
</style>
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}

@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



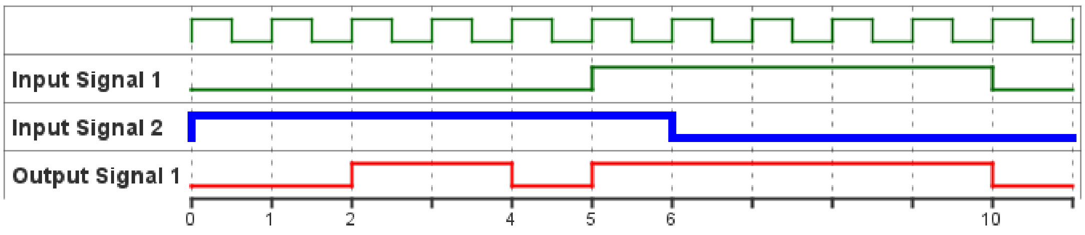
[Ref. QA-14340]

10.20 对特定行着色

可以使用 `<style>` 标记和模板将行属性命名

```
@startuml
<style>
timingDiagram {
  .red {
    LineColor red
  }
  .blue {
    LineColor blue
    LineThickness 5
  }
}
</style>

clock clk with period 1
binary "Input Signal 1"  as IS1
binary "Input Signal 2"  as IS2 <<blue>>
binary "Output Signal 1" as OS1 <<red>>

@0
IS1 is low
```

```
IS2 is high
OS1 is low
@2
OS1 is high
@4
OS1 is low
@5
IS1 is high
OS1 is high
@6
IS2 is low
@10
IS1 is low
OS1 is low
@enduml
```



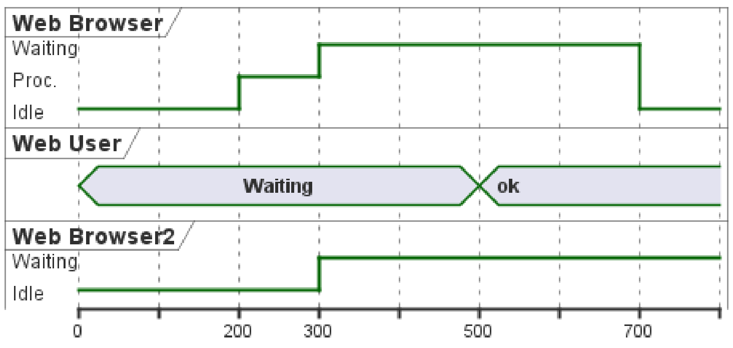
[Ref. QA-15870]

10.21 紧凑模式

可以使用 compact 命令进行紧凑排版

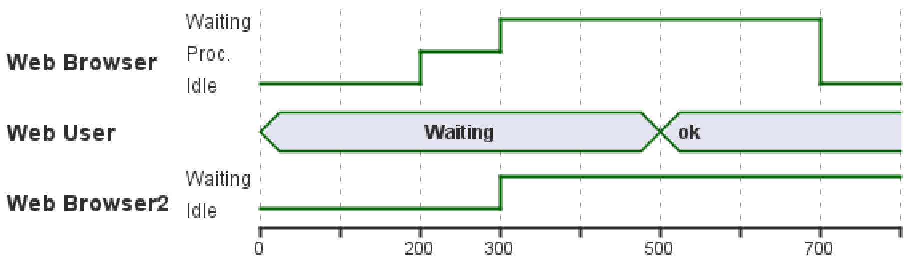
10.21.1 默认模式

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2

@0
WU is Waiting
WB is Idle
WB2 is Idle

@200
WB is Proc.

@300
WB is Waiting
WB2 is Waiting

@500
WU is ok

@700
WB is Idle
@enduml
```



10.21.2 使用 `mode compact` 的全局紧凑模式

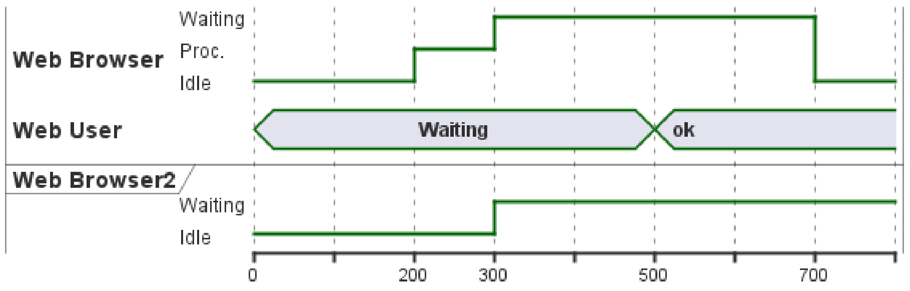
```
@startuml
mode compact
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2

@0
WU is Waiting
WB is Idle
WB2 is Idle

@200
WB is Proc.

@300
WB is Waiting
WB2 is Waiting

@500
WU is ok

@700
WB is Idle
@enduml
```



10.21.3 使用 `compact` 实现仅针对元素的局部紧凑模式

```
@startuml
compact robust "Web Browser" as WB
compact concise "Web User" as WU
robust "Web Browser2" as WB2

@0
WU is Waiting
WB is Idle
```

```
WB2 is Idle

@200
WB is Proc.

@300
WB is Waiting
WB2 is Waiting

@500
WU is ok

@700
WB is Idle
@enduml
```



[Ref. QA-11130]

11 Display JSON Data

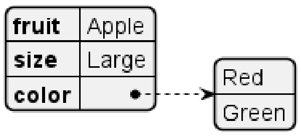
JSON format is widely used in software.

You can use PlantUML to visualize your data.

To activate this feature, the diagram must:

- begin with `@startjson` keyword
- end with `@endjson` keyword.

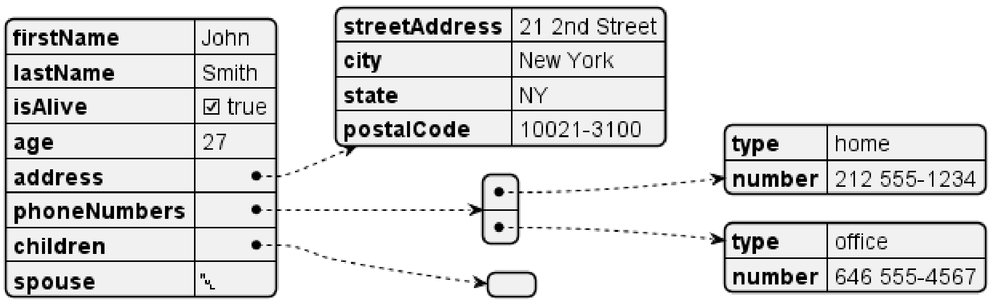
```
@startjson
{
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@endjson
```



11.1 Complex example

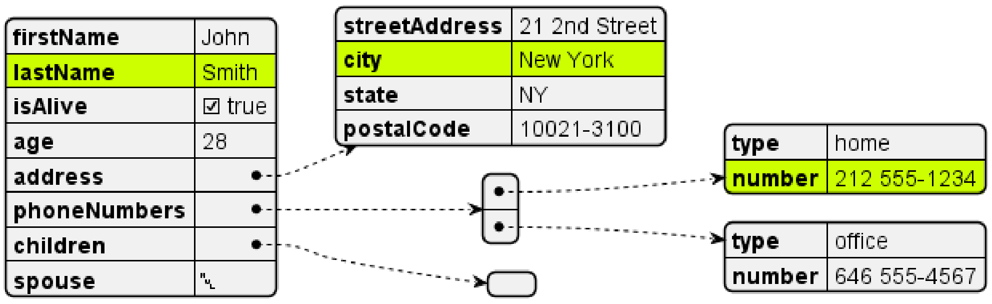
You can use complex JSON structure.

```
@startjson
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson
```



11.2 Highlight parts

```
@startjson
#highlight "lastName"
#highlight "address" / "city"
#highlight "phoneNumbers" / "0" / "number"
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 28,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson
```

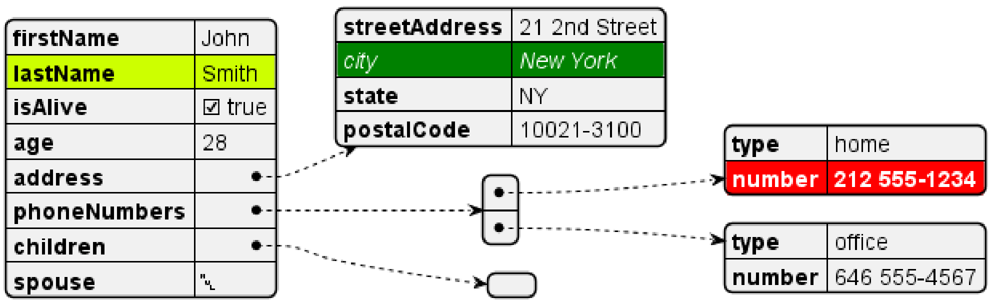


11.3 Using different styles for highlight

It is possible to have different styles for different highlights.

```
@startjson
```

```
<style>
  .h1 {
    BackGroundColor green
    FontColor white
    FontStyle italic
  }
  .h2 {
    BackGroundColor red
    FontColor white
    FontStyle bold
  }
</style>
#highlight "lastName"
#highlight "address" / "city" <<h1>>
#highlight "phoneNumbers" / "0" / "number" <<h2>>
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 28,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson
```



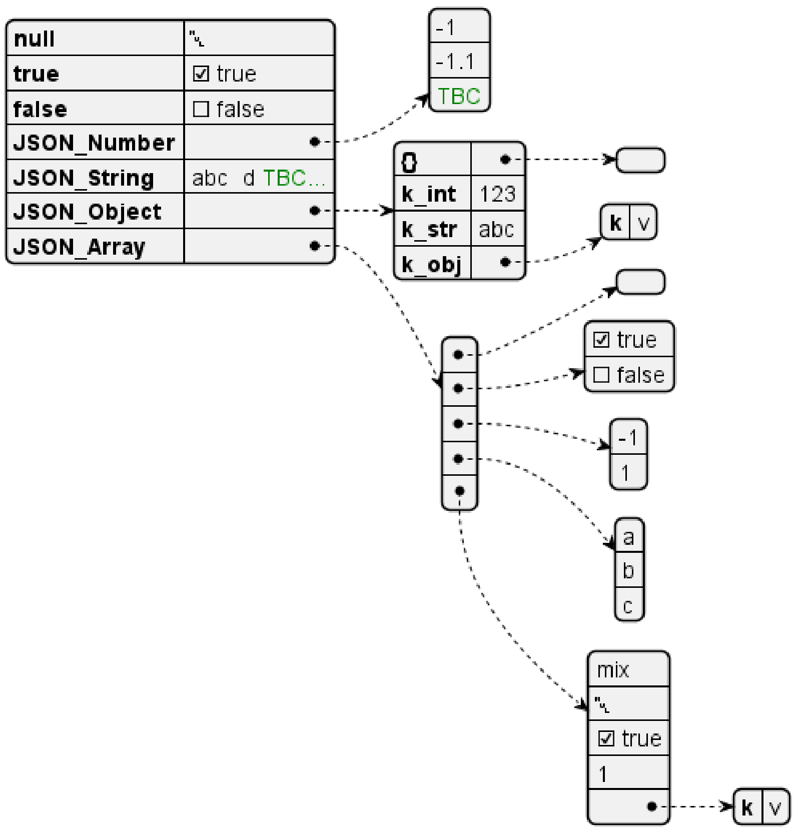
[Ref. QA-15756, GH-1393]

11.4 JSON basic element

11.4.1 Synthesis of all JSON basic element

```
@startjson
{
```

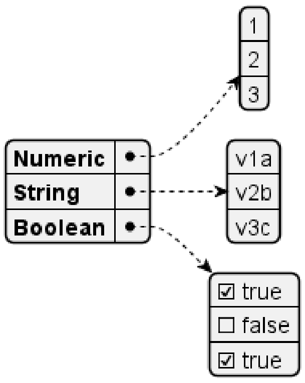
```
"null": null,
"true": true,
"false": false,
"JSON_Number": [-1, -1.1, "<color:green>TBC"],
"JSON_String": "a\nb\rc\td <color:green>TBC...",
"JSON_Object": {
  "{}": {},
  "k_int": 123,
  "k_str": "abc",
  "k_obj": {"k": "v"}
},
"JSON_Array" : [
  [],
  [true, false],
  [-1, 1],
  ["a", "b", "c"],
  ["mix", null, true, 1, {"k": "v"}]
]
}
@endjson
```



11.5 JSON array or table

11.5.1 Array type

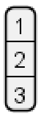
```
@startjson
{
"Numeric": [1, 2, 3],
"String ": ["v1a", "v2b", "v3c"],
"Boolean": [true, false, true]
}
@endjson
```



11.5.2 Minimal array or table

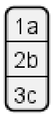
11.5.3 Number array

```
@startjson
[1, 2, 3]
@endjson
```



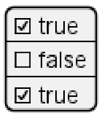
11.5.4 String array

```
@startjson
["1a", "2b", "3c"]
@endjson
```



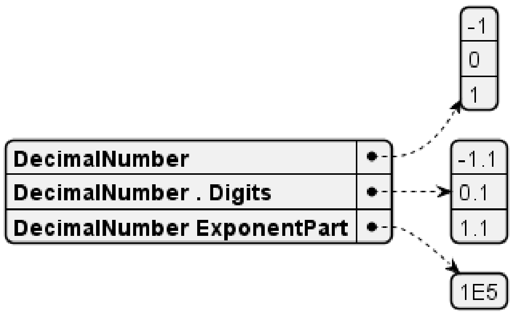
11.5.5 Boolean array

```
@startjson
[true, false, true]
@endjson
```



11.6 JSON numbers

```
@startjson
{
"DecimalNumber": [-1, 0, 1],
"DecimalNumber . Digits": [-1.1, 0.1, 1.1],
"DecimalNumber ExponentPart": [1E5]
}
@endjson
```

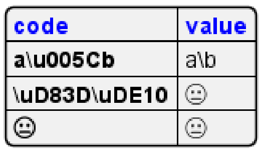


11.7 JSON strings

11.7.1 JSON Unicode

On JSON you can use Unicode directly or by using escaped form like .

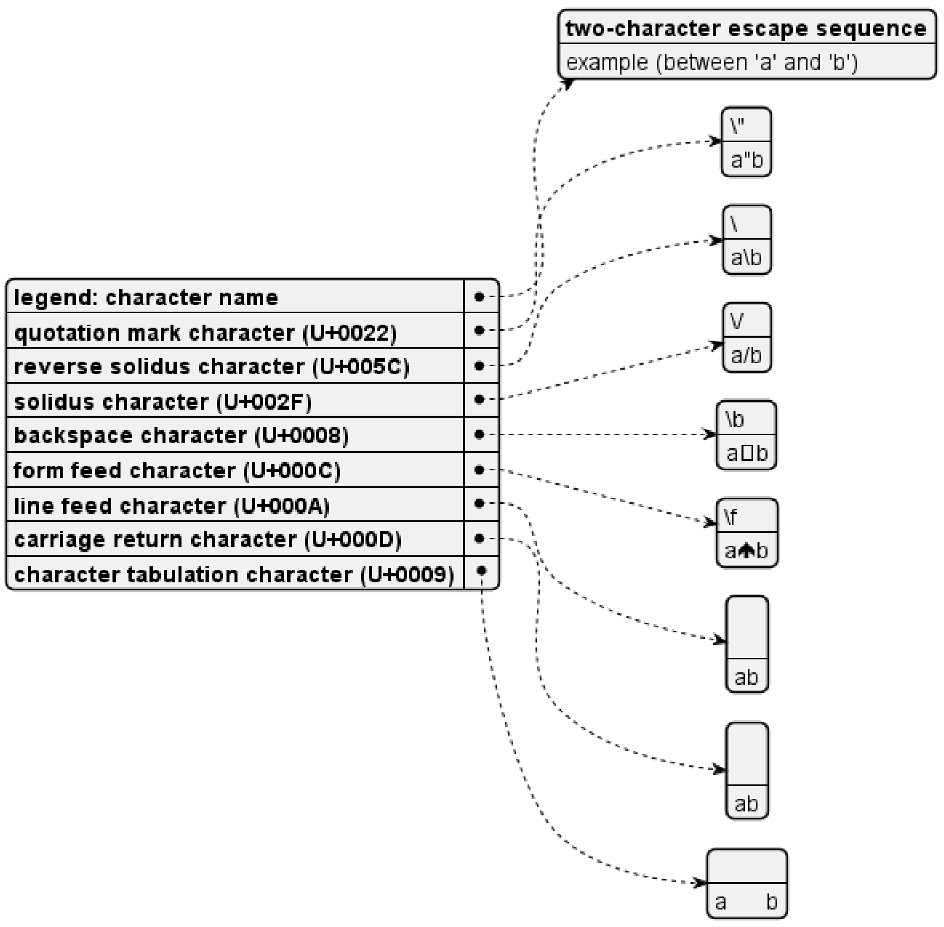
```
@startjson
{
  "<color:blue><b>code":  "<color:blue><b>value",
  "a\\u005Cb":            "a\u005Cb",
  "\\uD83D\\uDE10":       "\uD83D\uDE10",
  " ":                   " "
}
@endjson
```



code	value
a\u005Cb	a\b
\uD83D\uDE10	😐
😐	😐

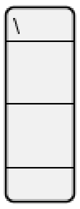
11.7.2 JSON two-character escape sequence

```
@startjson
{
 "**legend**: character name":               ["**two-character escape sequence**", "example (between
 "quotation mark character (U+0022)":        ["\\\"", "a\"b"],
 "reverse solidus character (U+005C)":       ["\\\\", "a\\b"],
 "solidus character (U+002F)":               ["\\\/", "a\/b"],
 "backspace character (U+0008)":             ["\\b", "a\bb"],
 "form feed character (U+000C)":             ["\\f", "a\fb"],
 "line feed character (U+000A)":             ["\\n", "a\nb"],
 "carriage return character (U+000D)":       ["\\r", "a\rb"],
 "character tabulation character (U+0009)":  ["\\t", "a\tb"]
}
@endjson
```


two-character escape sequence
example (between 'a' and 'b')
legend: character name
quotation mark character (U+0022)
reverse solidus character (U+005C)
solidus character (U+002F)
backspace character (U+0008)
form feed character (U+000C)
line feed character (U+000A)
carriage return character (U+000D)
character tabulation character (U+0009)
\"
a"b
\
a\b
\/
a/b
\b
a b
\f
a b
ab
ab
a b

TODO: FIXME FIXME or not , on the same item as management in PlantUML *See Report Bug on QA-13066* **TODO:** FIXME

```
@startjson
[
"\\\\",
"\\n",
"\\r",
"\\t"
]
@endjson
```

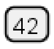


11.8 Minimal JSON examples

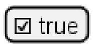
```
@startjson
"Hello world!"
@endjson
```

Hello world!

```
@startjson
42
@endjson
```



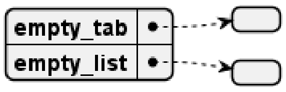
```
@startjson
true
@endjson
```



(Examples come from STD 90 - Examples)

11.9 Empty table or list

```
@startjson
{
  "empty_tab": [],
  "empty_list": {}
}
@endjson
```

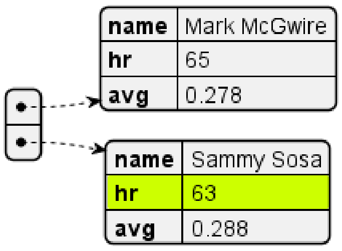


[Ref. QA-14397]

11.10 Using (global) style

11.10.1 Without style *(by default)*

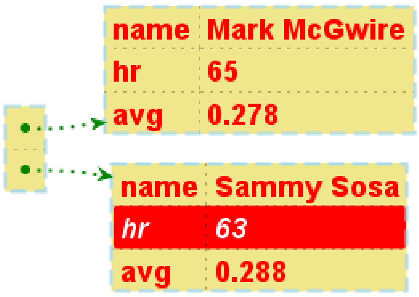
```
@startjson
#highlight "1" / "hr"
[
  {
    "name": "Mark McGwire",
    "hr":   65,
    "avg":  0.278
  },
  {
    "name": "Sammy Sosa",
    "hr":   63,
    "avg":  0.288
  }
]
@endjson
```



11.10.2 With style

You can use style to change rendering of elements.

```
@startjson
<style>
jsonDiagram {
  node {
    BackGroundColor Khaki
    LineColor lightblue
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    RoundCorner 0
    LineThickness 2
    LineStyle 10-5
    separator {
      LineThickness 0.5
      LineColor black
      LineStyle 1-5
    }
  }
  arrow {
    BackGroundColor lightblue
    LineColor green
    LineThickness 2
    LineStyle 2-5
  }
  highlight {
    BackGroundColor red
    FontColor white
    FontStyle italic
  }
}
</style>
#highlight "1" / "hr"
[
  {
    "name": "Mark McGwire",
    "hr":   65,
    "avg":  0.278
  },
  {
    "name": "Sammy Sosa",
    "hr":   63,
    "avg":  0.288
  }
]
@endjson
```

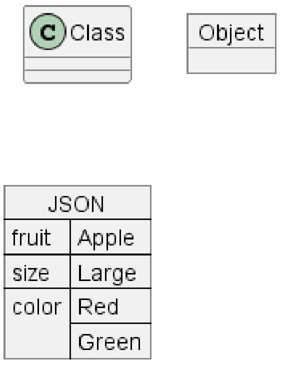


[Adapted from QA-13123 and QA-13288]

11.11 Display JSON Data on Class or Object diagram

11.11.1 Simple example

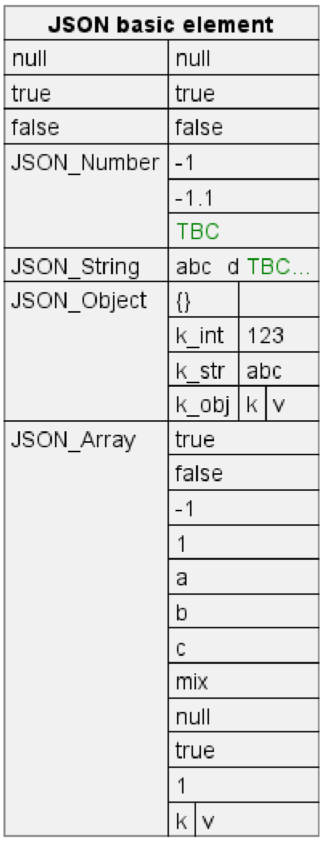
```
@startuml
class Class
object Object
json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-15481]

11.11.2 Complex example: with all JSON basic element

```
@startuml
json "<b>JSON basic element" as J {
"null": null,
"true": true,
"false": false,
"JSON_Number": [-1, -1.1, "<color:green>TBC"],
"JSON_String": "a\nb\rc\td <color:green>TBC...",
"JSON_Object": {
  "{}": {},
  "k_int": 123,
  "k_str": "abc",
  "k_obj": {"k": "v"}
},
"JSON_Array" : [
  [],
  [true, false],
  [-1, 1],
  ["a", "b", "c"],
  ["mix", null, true, 1, {"k": "v"}]
]
}
@enduml
```

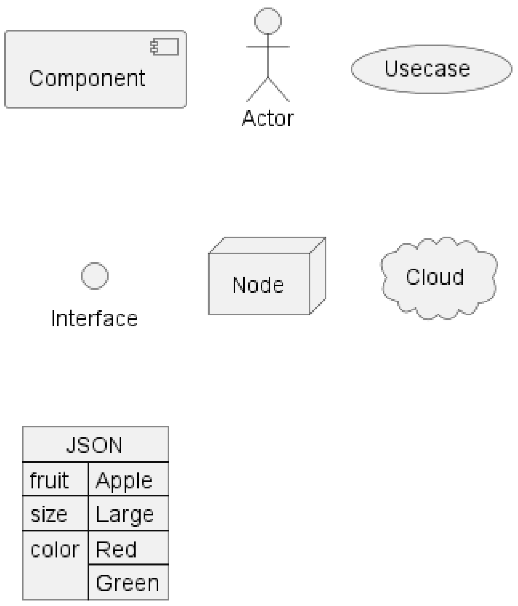


JSON basic element			
null	null		
true	true		
false	false		
JSON_Number	-1		
	-1.1		
	TBC		
JSON_String	abc d TBC...		
JSON_Object	{}		
	k_int	123	
	k_str	abc	
	k_obj	k	v
JSON_Array	true		
	false		
	-1		
	1		
	a		
	b		
	c		
	mix		
	null		
	true		
	1		
	k	v	

11.12 Display JSON Data on Deployment (Usecase, Component, Deployment) diagram

11.12.1 Simple example

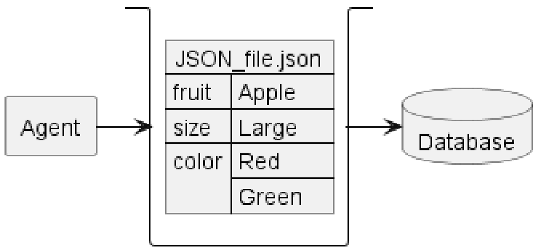
```
@startuml
allowmixing

component Component
actor     Actor
usecase   Usecase
()        Interface
node      Node
cloud     Cloud

json JSON {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-15481]

Complex example: with arrow

```
@startuml
allowmixing

agent Agent
stack {
  json "JSON_file.json" as J {
    "fruit":"Apple",
    "size":"Large",
    "color": ["Red", "Green"]
  }
}
database Database

Agent -> J
J -> Database
@enduml
```

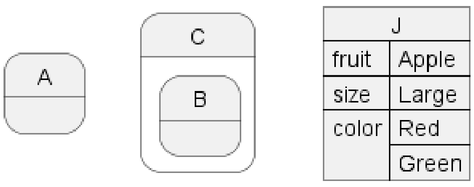


11.13 Display JSON Data on State diagram

11.13.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
 state B
}
```

```
json J {
   "fruit":"Apple",
   "size":"Large",
   "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-17275]

12 YAML 显示效果图

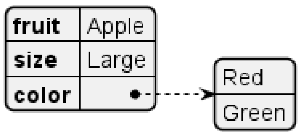
YAML 格式广泛的在软件中使用.

可以使用 PlantUML 可视化您的数据.

要激活此功能, 需要:

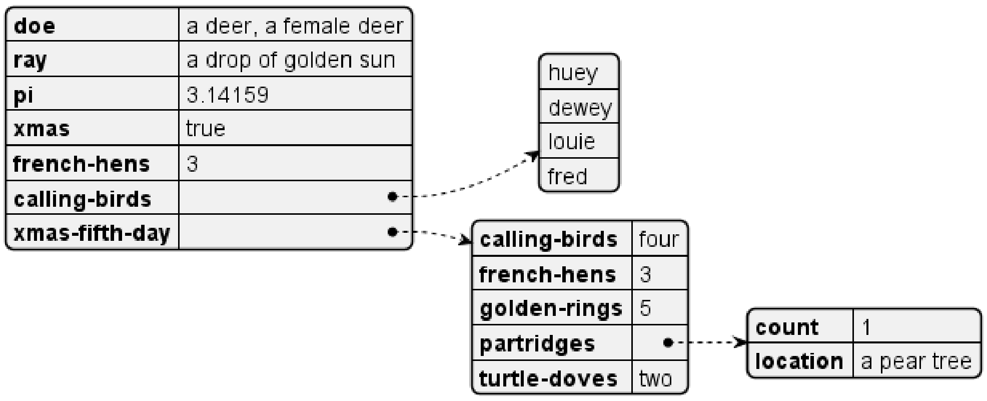
- 以关键字 @startyaml 开始
- 以关键字 @endyaml 结束.

```
@startyaml
fruit: Apple
size: Large
color:
  - Red
  - Green
@endyaml
```



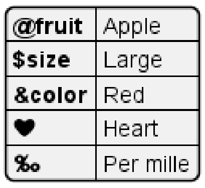
12.1 复杂例子

```
@startyaml
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



12.2 特定的 key (使用 symbols 或者 unicode)

```
@startyaml
@fruit: Apple
$size: Large
&color: Red
 : Heart
‰: Per mille
@endyaml
```



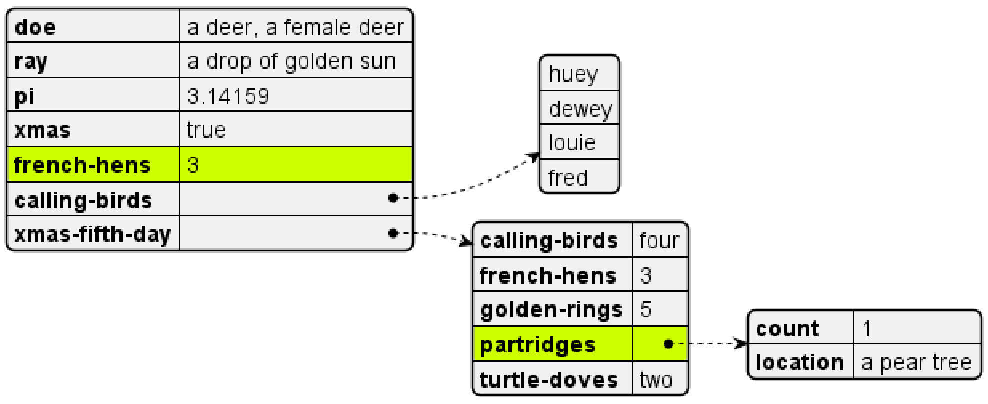
[Ref. QA-13376]

12.3 强调

12.3.1 正常样式

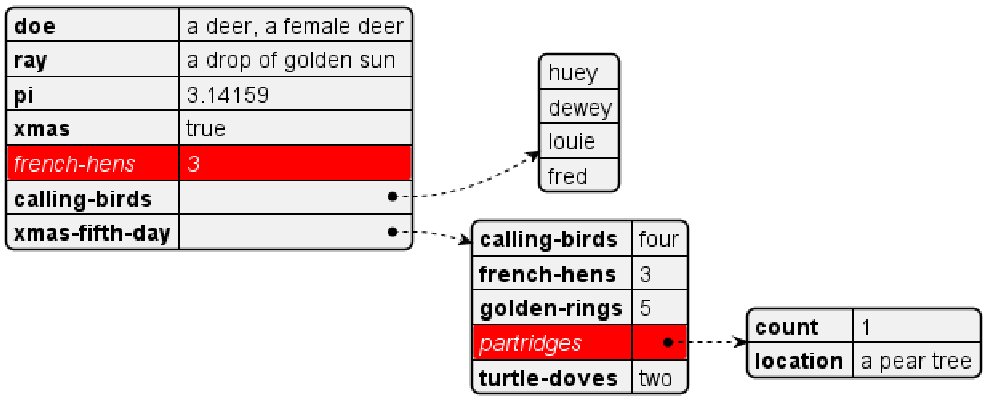
```
@startyaml
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
```

```
@endyaml
```



12.3.2 自定义样式

```
@startyaml
<style>
yamlDiagram {
    highlight {
      BackGroundColor red
      FontColor white
      FontStyle italic
    }
}
</style>
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```

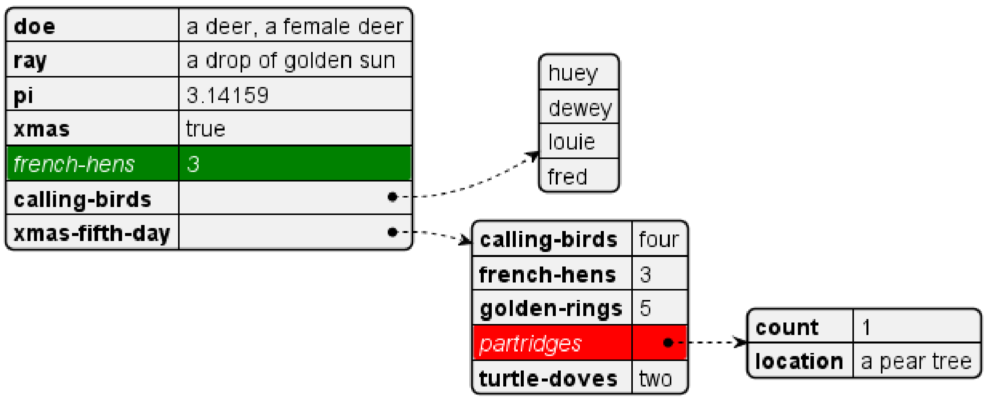


[Ref. QA-13288]

12.4 Using different styles for highlight

It is possible to have different styles for different highlights.

```
@startyaml
<style>
    .h1 {
      BackGroundColor green
      FontColor white
      FontStyle italic
    }
    .h2 {
      BackGroundColor red
      FontColor white
      FontStyle italic
    }
</style>
#highlight "french-hens" <<h1>>
#highlight "xmas-fifth-day" / "partridges" <<h2>>

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```

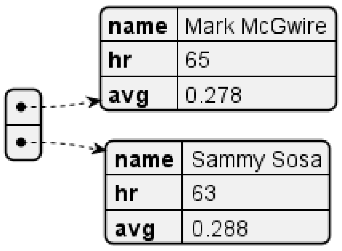


[Ref. QA-15756, GH-1393]

12.5 使用 (global) 全局样式

12.5.1 不带样式 *(默认)*

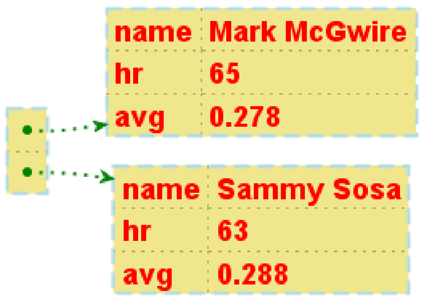
```
@startyaml
  -
    name: Mark McGwire
    hr:   65
    avg:  0.278
  -
    name: Sammy Sosa
    hr:   63
    avg:  0.288
@endyaml
```



12.5.2 使用样式

您可以使用 style 更改元素的显示.

```
@startyaml
<style>
yamlDiagram {
  node {
    BackGroundColor lightblue
    LineColor lightblue
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor Khaki
    RoundCorner 0
    LineThickness 2
    LineStyle 10-5
    separator {
      LineThickness 0.5
      LineColor black
      LineStyle 1-5
    }
  }
  arrow {
    BackGroundColor lightblue
    LineColor green
    LineThickness 2
    LineStyle 2-5
  }
}
</style>
  -
    name: Mark McGwire
    hr:   65
    avg:  0.278
  -
    name: Sammy Sosa
    hr:   63
    avg:  0.288
@endyaml
```



[Ref. QA-13123]

13 网络图（nwdiag）

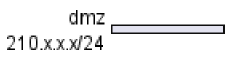
nwdiag 是一位名叫 Takeshi Komiya 的人创建的，这使得我们可以快速绘制网络拓扑图。让我们为此感谢他的创作！

由于语法非常清晰和简单，该功能已经集成在 PlantUML 中。我们在这里展示 Takeshi 归档整理的示例。

13.1 简单图示

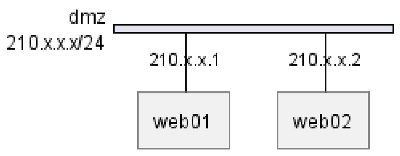
13.1.1 定义一个网络

```
@startuml
nwdiag {
  network dmz {
      address = "210.x.x.x/24"
  }
}
@enduml
```



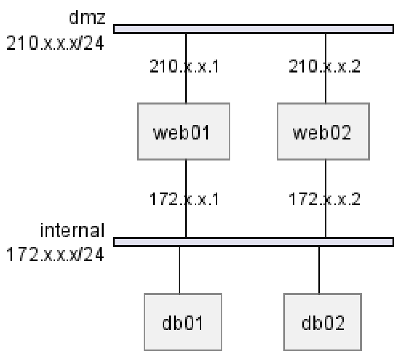
13.1.2 定义网络中的一些元素或服务器

```
@startuml
nwdiag {
  network dmz {
      address = "210.x.x.x/24"

      web01 [address = "210.x.x.1"];
      web02 [address = "210.x.x.2"];
  }
}
@enduml
```



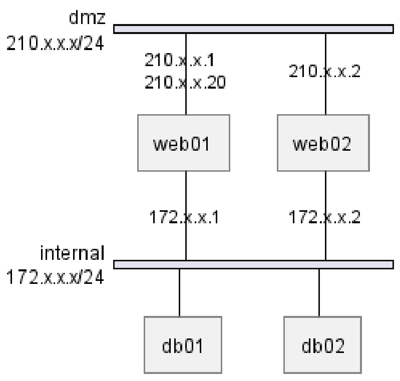
13.1.3 完整的例子

```
@startuml
nwdiag {
  network dmz {
      address = "210.x.x.x/24"

      web01 [address = "210.x.x.1"];
      web02 [address = "210.x.x.2"];
  }
  network internal {
      address = "172.x.x.x/24";

      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01;
      db02;
  }
}
@enduml
```



13.2 定义多个地址

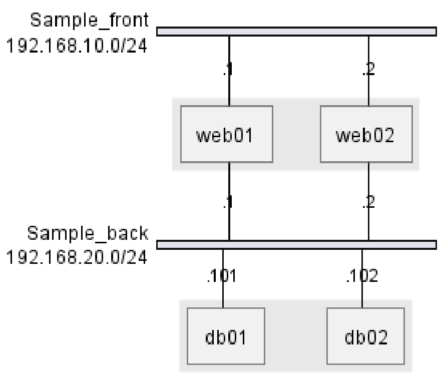
```
@startuml
nwdiag {
  network dmz {
      address = "210.x.x.x/24"

      // set multiple addresses (using comma)
      web01 [address = "210.x.x.1, 210.x.x.20"];
      web02 [address = "210.x.x.2"];
  }
  network internal {
      address = "172.x.x.x/24";

      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01;
      db02;
  }
}
@enduml
```



13.3 群节点

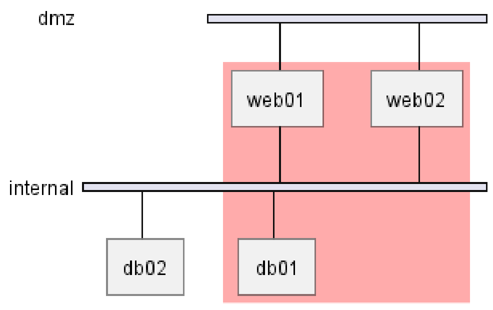
13.3.1 在网络定义中定义组

```
@startuml
nwdiag {
  network Sample_front {
    address = "192.168.10.0/24";

    // define group
    group web {
      web01 [address = ".1"];
      web02 [address = ".2"];
    }
  }
  network Sample_back {
    address = "192.168.20.0/24";
    web01 [address = ".1"];
    web02 [address = ".2"];
    db01 [address = ".101"];
    db02 [address = ".102"];

    // define network using defined nodes
    group db {
      db01;
      db02;
    }
  }
}
@enduml
```



13.3.2 在网络定义之外定义组

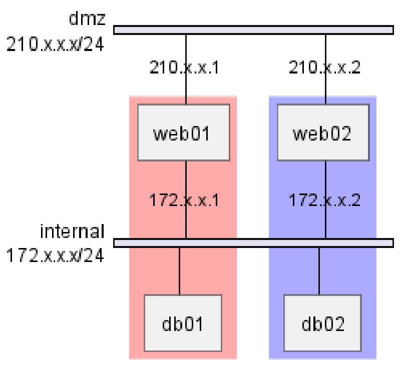
```
@startuml
nwdiag {
  // define group outside of network definitions
  group {
    color = "#FFAAAA";

    web01;
    web02;
    db01;
  }

  network dmz {
    web01;
    web02;
  }
  network internal {
    web01;
    web02;
    db01;
    db02;
  }
}
@enduml
```



13.3.3 在同一个网络上定义不同的一些组

13.3.4 一个定义了两个组的例子

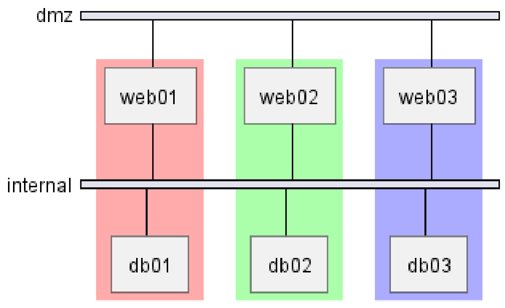
```
@startuml
nwdiag {
  group {
    color = "#FFaaaa";
    web01;
    db01;
  }
  group {
    color = "#aaaaFF";
    web02;
    db02;
  }
  network dmz {
      address = "210.x.x.x/24"

      web01 [address = "210.x.x.1"];
      web02 [address = "210.x.x.2"];
  }
  network internal {
      address = "172.x.x.x/24";

      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01 ;
      db02 ;
  }
}
@enduml
```



[Ref. QA-12663]

13.3.5 定义了三个组的例子

```
@startuml
nwdiag {
  group {
    color = "#FFaaaa";
    web01;
    db01;
  }
  group {
    color = "#aaFFaa";
    web02;
    db02;
  }
  group {
    color = "#aaaaFF";
    web03;
    db03;
  }

  network dmz {
      web01;
      web02;
      web03;
  }
  network internal {
      web01;
      db01 ;
      web02;
      db02 ;
      web03;
      db03;
  }
}
@enduml
```



[Ref. QA-13138]

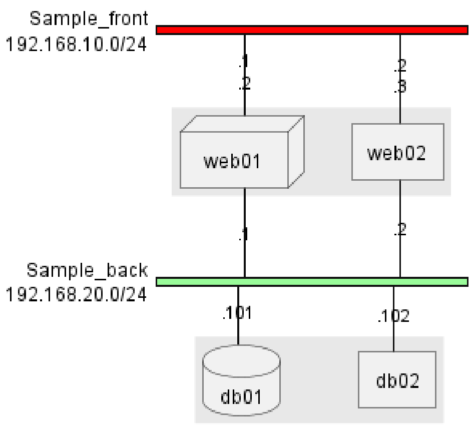
13.4 拓展语法 (适用于组或者网络)

13.4.1 网络

用于网络或者网络的组成部分，你可以添加或者修改:

- 地址 *(使用, 来分隔)*;
- 颜色;
- 描述;
- 形状.

```
@startuml
nwdiag {
  network Sample_front {
    address = "192.168.10.0/24"
    color = "red"

    // define group
    group web {
      web01 [address = ".1, .2", shape = "node"]
      web02 [address = ".2, .3"]
    }
  }
  network Sample_back {
    address = "192.168.20.0/24"
    color = "palegreen"
    web01 [address = ".1"]
    web02 [address = ".2"]
    db01 [address = ".101", shape = database ]
    db02 [address = ".102"]

    // define network using defined nodes
    group db {
      db01;
      db02;
    }
  }
}
@enduml
```

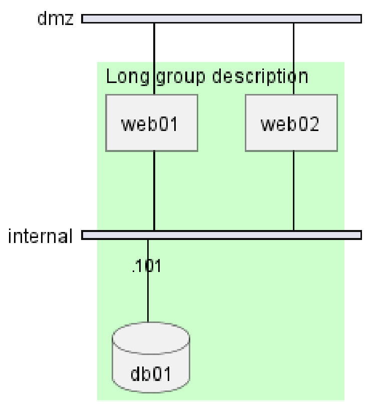


13.4.2 组

对于组，你可以添加或修改:

- 颜色;
- 描述.

```
@startuml
nwdiag {
  group {
    color = "#CCFFCC";
    description = "Long group description";

    web01;
    web02;
    db01;
  }

  network dmz {
    web01;
    web02;
  }
  network internal {
    web01;
    web02;
    db01 [address = ".101", shape = database];
  }
}
@enduml
```



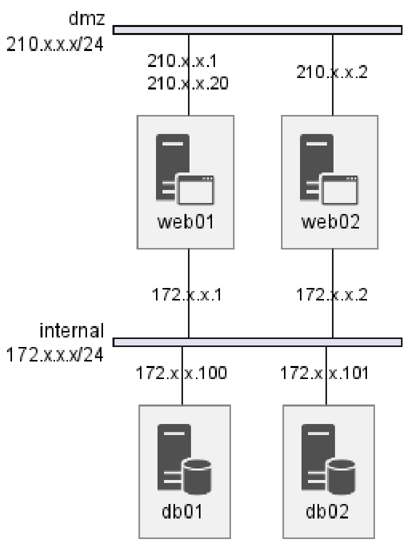
[Ref. QA-12056]

13.5 Using Sprites

You can use all sprites (icons) from the Standard Library or any other library.

Use the notation **<$sprite>** to use a sprite, to make a new line, or any other Creole syntax.

```
@startuml
!include <office/Servers/application_server>
!include <office/Servers/database_server>

nwdiag {
  network dmz {
      address = "210.x.x.x/24"

      // set multiple addresses (using comma)
      web01 [address = "210.x.x.1, 210.x.x.20",  description = "<$application_server>\n web01"]
      web02 [address = "210.x.x.2",  description = "<$application_server>\n web02"];
  }
  network internal {
      address = "172.x.x.x/24";

      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01 [address = "172.x.x.100",  description = "<$database_server>\n db01"];
      db02 [address = "172.x.x.101",  description = "<$database_server>\n db02"];
  }
}
@enduml
```



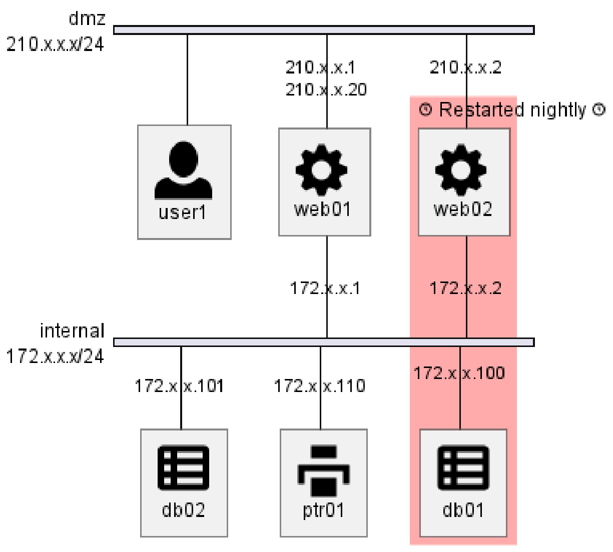
[Ref. QA-11862]

13.6 Using OpenIconic

You can also use the icons from OpenIconic in network or node descriptions.

Use the notation `<&icon>` to make an icon, `<&icon*n>` to multiply the size by a factor `n`, and to make a newline:

```
@startuml

nwdiag {
  group nightly {
    color = "#FFAAAA";
    description = "<&clock> Restarted nightly <&clock>";
    web02;
    db01;
  }
  network dmz {
      address = "210.x.x.x/24"

      user [description = "<&person*4.5>\n user1"];
      // set multiple addresses (using comma)
      web01 [address = "210.x.x.1, 210.x.x.20",  description = "<&cog*4>\nweb01"]
      web02 [address = "210.x.x.2",  description = "<&cog*4>\nweb02"];

  }
  network internal {
      address = "172.x.x.x/24";

      web01 [address = "172.x.x.1"];
      web02 [address = "172.x.x.2"];
      db01 [address = "172.x.x.100",  description = "<&spreadsheet*4>\n db01"];
      db02 [address = "172.x.x.101",  description = "<&spreadsheet*4>\n db02"];
      ptr  [address = "172.x.x.110",  description = "<&print*4>\n ptr01"];
  }
}
@enduml
```

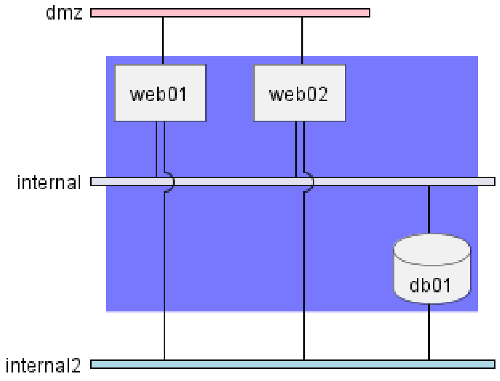


13.7 Same nodes on more than two networks

You can use same nodes on different networks (more than two networks); *nwdiag* use in this case *'jump line'* over networks.

```
@startuml
nwdiag {
  // define group at outside network definitions
  group {
    color = "#7777FF";

    web01;
    web02;
    db01;
  }

  network dmz {
    color = "pink"

    web01;
    web02;
  }

  network internal {
    web01;
    web02;
    db01 [shape = database ];
  }

  network internal2 {
    color = "LightBlue";

    web01;
    web02;
    db01;
  }

}
```

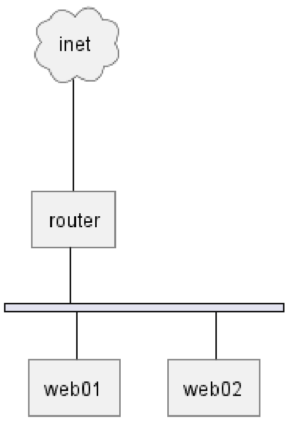
```
@enduml
```



13.8 Peer networks

Peer networks are simple connections between two nodes, for which we don't use a horizontal "busbar" network

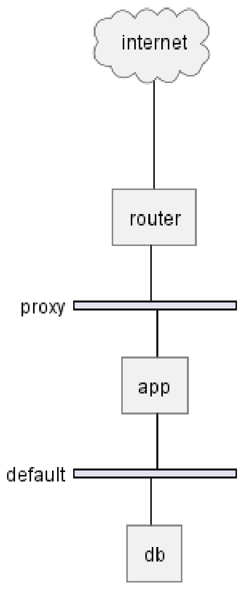
```
@startuml
nwdiag {
  inet [shape = cloud];
  inet -- router;

  network {
    router;
    web01;
    web02;
  }
}
@enduml
```



13.9 Peer networks and group

13.9.1 Without group

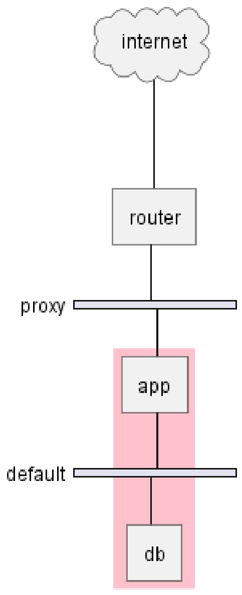
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
     app;
        db;
    }
}
@enduml
```



13.9.2 Group on first

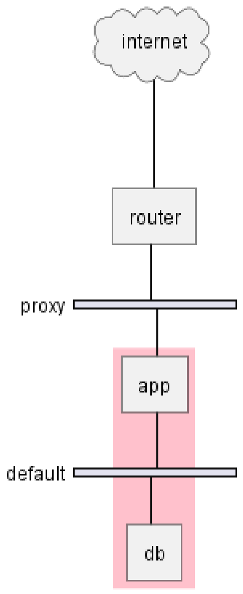
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    group {
      color = "pink";
      app;
      db;
    }

    network proxy {
        router;
        app;
    }

    network default {
     app;
        db;
    }
}
```

```
@enduml
```



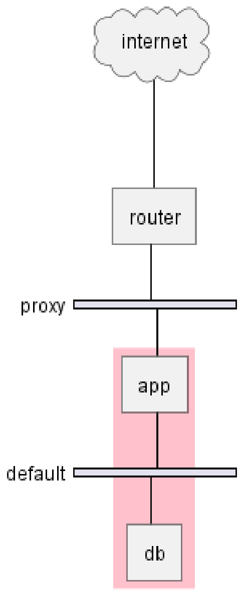
13.9.3 Group on second

```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }

    group {
      color = "pink";
      app;
      db;
    }

    network default {
     app;
        db;
    }
}
@enduml
```



13.9.4 Group on third

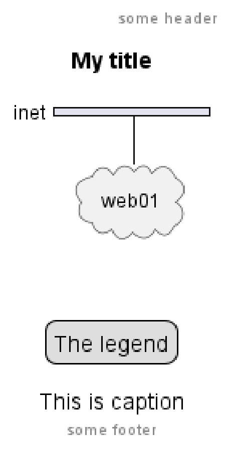
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
     app;
        db;
    }
    group {
      color = "pink";
      app;
      db;
    }
}
@enduml
```



[Ref. Issue#408 and QA-12655]

13.10 Add title, caption, header, footer or legend on network diagram

```
@startuml

header some header

footer some footer

title My title

nwdiag {
  network inet {
      web01 [shape = cloud]
  }
}

legend
The legend
end legend

caption This is caption
@enduml
```

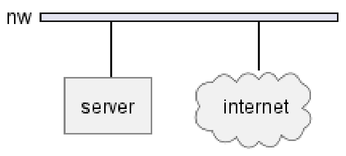


[Ref. QA-11303 and Common commands]

13.11 With or without shadow

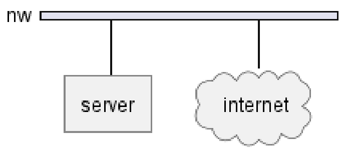
13.11.1 With shadow (by default)

```
@startuml
nwdiag {
  network nw {
    server;
    internet;
  }
  internet [shape = cloud];
}
@enduml
```



13.11.2 Without shadow

```
@startuml
<style>
root {
 shadowing 0
}
</style>
nwdiag {
  network nw {
    server;
    internet;
  }
  internet [shape = cloud];
}
@enduml
```



[Ref. QA-14516]

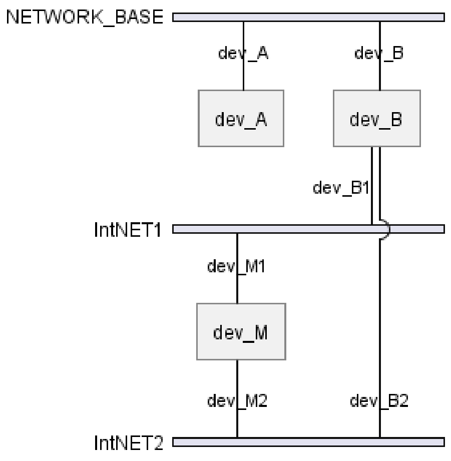
13.12 Change width of the networks

You can change the width of the networks, especially in order to have the same full width for only some or all networks.

Here are some examples, with all the possibilities:

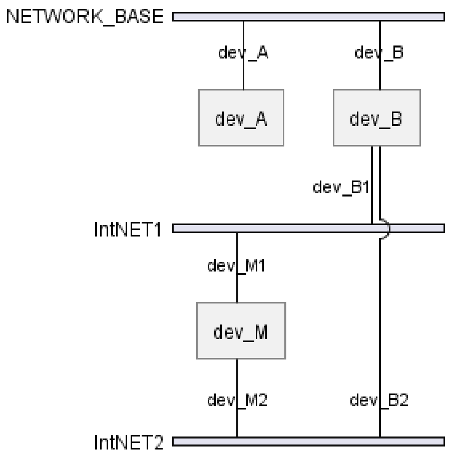
- without

```
@startuml
nwdiag {
  network NETWORK_BASE {
   dev_A [address = "dev_A" ]
   dev_B [address = "dev_B" ]
  }
  network IntNET1 {
   dev_B [address = "dev_B1" ]
   dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
   dev_B [address = "dev_B2" ]
   dev_M [address = "dev_M2" ]
 }
}
@enduml
```



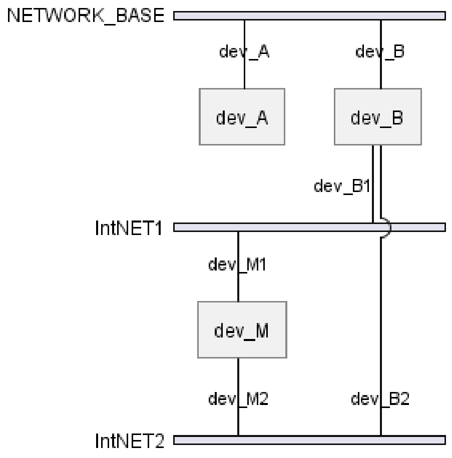
- only the first

```
@startuml
nwdiag {
  network NETWORK_BASE {
   width = full
   dev_A [address = "dev_A" ]
   dev_B [address = "dev_B" ]
  }
  network IntNET1 {
   dev_B [address = "dev_B1" ]
   dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
   dev_B [address = "dev_B2" ]
   dev_M [address = "dev_M2" ]
 }
}
@enduml
```



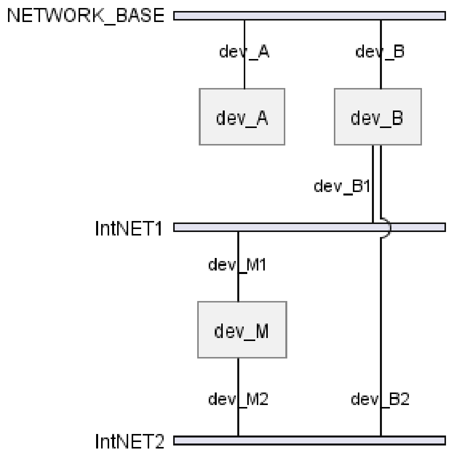
- the first and the second

```
@startuml
nwdiag {
  network NETWORK_BASE {
   width = full
   dev_A [address = "dev_A" ]
   dev_B [address = "dev_B" ]
  }
  network IntNET1 {
   width = full
   dev_B [address = "dev_B1" ]
   dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
   dev_B [address = "dev_B2" ]
   dev_M [address = "dev_M2" ]
 }
}
@enduml
```



- all the network (with same full width)

```
@startuml
nwdiag {
  network NETWORK_BASE {
   width = full
   dev_A [address = "dev_A" ]
   dev_B [address = "dev_B" ]
  }
  network IntNET1 {
   width = full
   dev_B [address = "dev_B1" ]
   dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
   width = full
   dev_B [address = "dev_B2" ]
   dev_M [address = "dev_M2" ]
 }
}
@enduml
```

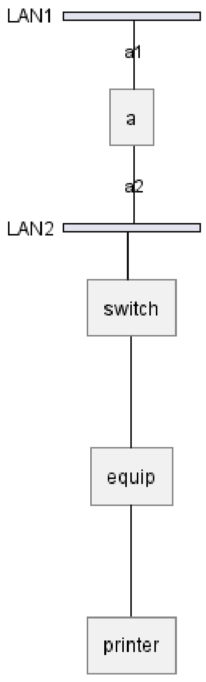


13.13 Other internal networks

You can define other internal networks (TCP/IP, USB, SERIAL,...).

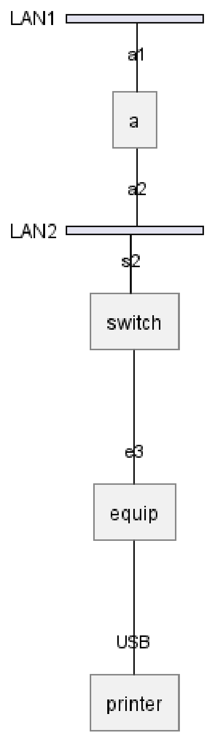
- Without address or type

```
@startuml
nwdiag {
  network LAN1 {
     a [address = "a1"];
  }
  network LAN2 {
     a [address = "a2"];
     switch;
  }
  switch -- equip;
  equip -- printer;
}
@enduml
```



- With address or type

```
@startuml
nwdiag {
  network LAN1 {
     a [address = "a1"];
  }
  network LAN2 {
     a [address = "a2"];
     switch [address = "s2"];
  }
  switch -- equip;
  equip [address = "e3"];
  equip -- printer;
  printer [address = "USB"];
}
@enduml
```

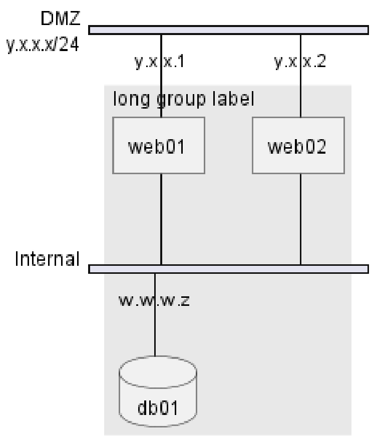


[Ref. QA-12824]

13.14 Using (global) style

13.14.1 Without style *(by default)*

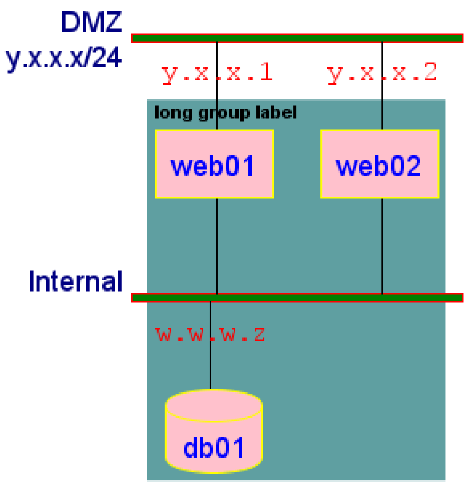
```
@startuml
nwdiag {
  network DMZ {
      address = "y.x.x.x/24"
      web01 [address = "y.x.x.1"];
      web02 [address = "y.x.x.2"];
  }

   network Internal {
    web01;
    web02;
    db01 [address = "w.w.w.z", shape = database];
  }

    group {
    description = "long group label";
    web01;
    web02;
    db01;
  }
}
@enduml
```



13.14.2 With style

You can use style to change rendering of elements.

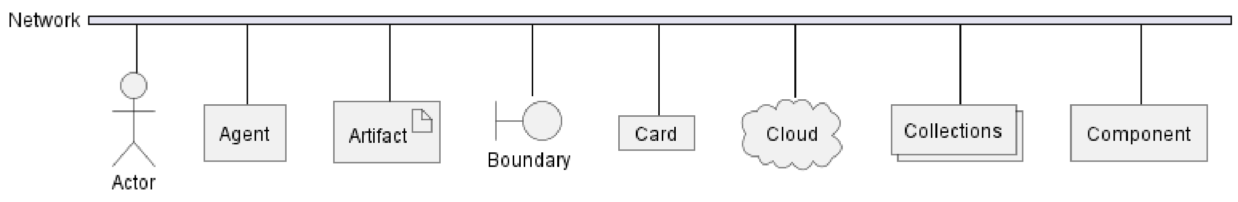
```
@startuml
<style>
nwdiagDiagram {
  network {
    BackGroundColor green
    LineColor red
    LineThickness 1.0
    FontSize 18
    FontColor navy
  }
  server {
    BackGroundColor pink
    LineColor yellow
    LineThickness 1.0
    ' FontXXX only for description or label
    FontSize 18
    FontColor #blue
  }
  arrow {
    ' FontXXX only for address
    FontSize 17
    FontColor #red
    FontName Monospaced
    LineColor black
  }
  group {
    BackGroundColor cadetblue
    LineColor black
    LineThickness 2.0
    FontSize 11
    FontStyle bold
    Margin 5
    Padding 5
  }
}
</style>
nwdiag {
  network DMZ {
      address = "y.x.x.x/24"
      web01 [address = "y.x.x.1"];
      web02 [address = "y.x.x.2"];
  }

   network Internal {
    web01;
    web02;
    db01 [address = "w.w.w.z", shape = database];
  }

    group {
    description = "long group label";
    web01;
    web02;
    db01;
  }
}
@enduml
```



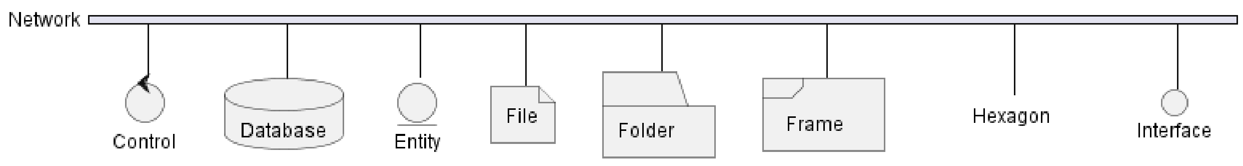
[Ref. QA-14479]

13.15 Appendix: Test of all shapes on Network diagram (nwdiag)

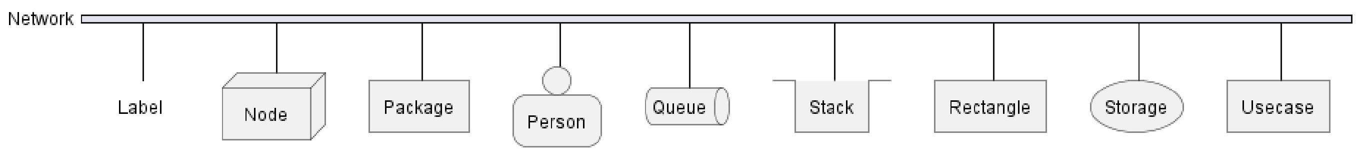
```
@startuml
nwdiag {
  network Network {
    Actor       [shape = actor]
    Agent       [shape = agent]
    Artifact    [shape = artifact]
    Boundary    [shape = boundary]
    Card        [shape = card]
    Cloud       [shape = cloud]
    Collections [shape = collections]
    Component   [shape = component]
  }
}
@enduml
```



```
@startuml
nwdiag {
  network Network {
    Control     [shape = control]
    Database    [shape = database]
    Entity      [shape = entity]
    File        [shape = file]
    Folder      [shape = folder]
    Frame       [shape = frame]
    Hexagon     [shape = hexagon]
    Interface   [shape = interface]
  }
}
@enduml
```

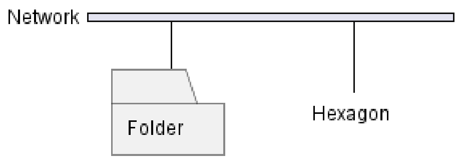


```
@startuml
nwdiag {
  network Network {
    Label       [shape = label]
    Node        [shape = node]
    Package     [shape = package]
    Person      [shape = person]
    Queue       [shape = queue]
    Stack       [shape = stack]
    Rectangle   [shape = rectangle]
    Storage     [shape = storage]
    Usecase     [shape = usecase]
  }
}
@enduml
```

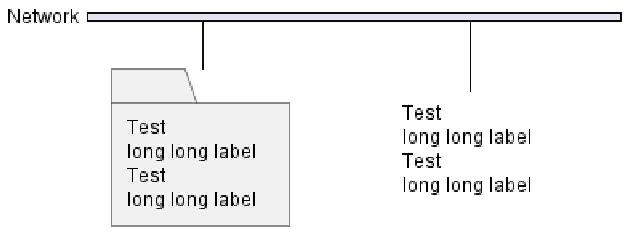


TODO: FIXME ol olli level 0 Overlap of label for folder olli olli level 0 Hexagon shape is missing olli ol

```
@startuml
nwdiag {
network Network {
Folder [shape = folder]
Hexagon [shape = hexagon]
}
}
@enduml
```



```
@startuml
nwdiag {
network Network {
Folder [shape = folder, description = "Test, long long label\nTest, long long label"]
Hexagon [shape = hexagon, description = "Test, long long label\nTest, long long label"]
}
}
@enduml
```



TODO: FIXME

14 Salt（线框图）

Salt 是 PlantUML 下面的子项目用来帮助用户来设计图形界面或是网页线框图/页面简图。

本工具的目标是实现简单的示例窗口。

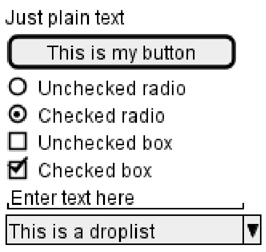
可以用 `@startsalt` 关键字，或者使用 `@startuml` 紧接着下一行使用 `salt` 关键字.

14.1 基本部件

一个窗口必须以中括号开头和结尾。接着可以这样定义:

- 按钮用 `[` 和 `]`。
- 单选按钮用 `(` 和 `)`。
- 复选框用 `[` 和 `]`。
- 用户文字域用 `"`。

```
@startsalt
{
  Just plain text
  [This is my button]
  ()  Unchecked radio
  (X) Checked radio
  []  Unchecked box
  [X] Checked box
  "Enter text here   "
  ^This is a droplist^
}
@endsalt
```

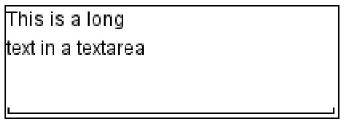


这个工具是用来讨论简单的示例窗口。

14.2 Text area

Here is an attempt to create a text area:

```
@startsalt
{+
   This is a long
   text in a textarea
   .
   "                          "
}
@endsalt
```



Note:

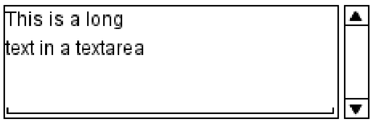
- the dot (.) to fill up vertical space;

- the last line of space ("   ") to make the area wider.

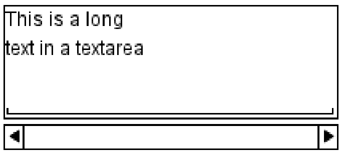
[Ref. QA-14765]

Then you can add scroll bar:

```
@startsalt
{SI
   This is a long
   text in a textarea
   .
   "                          "
}
@endsalt
```



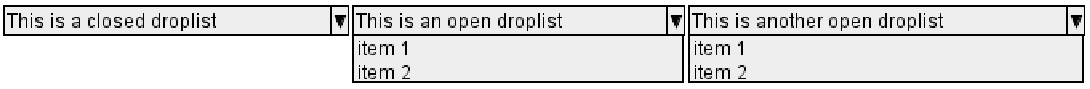
```
@startsalt
{S-
   This is a long
   text in a textarea
   .
   "                          "
}
@endsalt
```



14.3 Open, close droplist

You can open a droplist, by adding values enclosed by ^, as:

```
@startsalt
{
  ^This is a closed droplist^ |
  ^This is an open droplist^^ item 1^^ item 2^ |
  ^This is another open droplist^ item 1^ item 2^
}
@endsalt
```



[Ref. QA-4184]

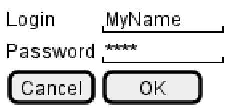
14.4 使用表格

当在输入关键词 {后，会自动建立一个表格

当输入 | 说明一个单元格

例子如下

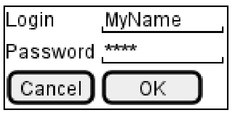
```
@startsalt
{
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



在启用关键词后，你可以使用以下字符来绘制表格中的线及列:

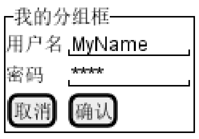
Symbol	Result
#	显示所有垂直水平线
!	显示所有垂直线
-	显示所有水平线
+	显示外框线

```
@startsalt
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



14.5 分组框

```
@startsalt
{^"我的分组框"
  用户名 | "MyName   "
  密码 | "****     "
  [取消] | [确认]
}
@endsalt
```

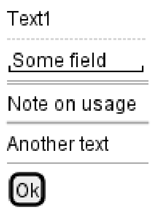


[参见 QA-5840]

14.6 使用分隔符

你可以使用几条横线表示分隔符

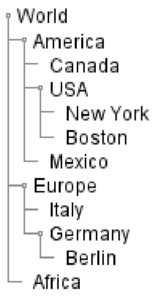
```
@startuml
salt
{
  Text1
  ..
  "Some field"
  ==
  Note on usage
  ~~
  Another text
  --
  [Ok]
}
@enduml
```



14.7 树形外挂

使用树结构，你必须要以 {T 进行起始，然后使用 + 定义层次。

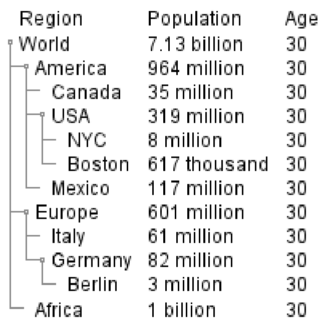
```
@startsalt
{
{T
 + World
 ++ America
 +++ Canada
 +++ USA
 ++++ New York
 ++++ Boston
 +++ Mexico
 ++ Europe
 +++ Italy
 +++ Germany
 ++++ Berlin
 ++ Africa
}
}
@endsalt
```



14.8 Tree table [T]

You can combine trees with tables.

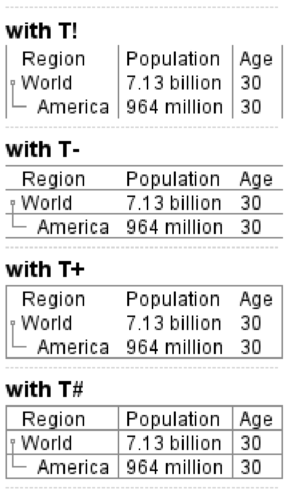
```
@startsalt
{
{T
+Region         | Population    | Age
+ World         | 7.13 billion  | 30
++ America      | 964 million   | 30
+++ Canada      | 35 million    | 30
+++ USA         | 319 million   | 30
++++ NYC        | 8 million     | 30
++++ Boston     | 617 thousand  | 30
+++ Mexico      | 117 million   | 30
++ Europe       | 601 million    | 30
+++ Italy       | 61 million     | 30
+++ Germany     | 82 million     | 30
++++ Berlin     | 3 million      | 30
++ Africa       | 1 billion      | 30
}
}
@endsalt
```



Region	Population	Age
World	7.13 billion	30
America	964 million	30
Canada	35 million	30
USA	319 million	30
NYC	8 million	30
Boston	617 thousand	30
Mexico	117 million	30
Europe	601 million	30
Italy	61 million	30
Germany	82 million	30
Berlin	3 million	30
Africa	1 billion	30

And add lines.

```
@startsalt
{
..
== with T!
{T!
+Region         | Population     | Age
+ World         | 7.13 billion   | 30
++ America      | 964 million    | 30
}
..
== with T-
{T-
+Region         | Population     | Age
+ World         | 7.13 billion   | 30
++ America      | 964 million    | 30
}
..
== with T+
{T+
+Region         | Population     | Age
+ World         | 7.13 billion   | 30
++ America      | 964 million    | 30
}
..
== with T#
{T#
+Region         | Population     | Age
+ World         | 7.13 billion   | 30
++ America      | 964 million    | 30
}
..
}
@endsalt
```



with T!

Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T-

Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T+

Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T#

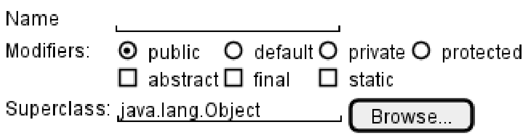
Region	Population	Age
World	7.13 billion	30
America	964 million	30

[Ref. QA-1265]

14.9 Enclosing brackets [{, }]

You can define subelements by opening a new opening bracket.

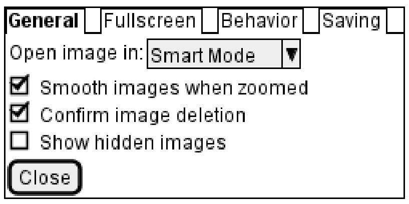
```
@startsalt
{
Name          | "                 "
Modifiers:    | { (X) public | () default | () private | () protected
                [] abstract | [] final   | [] static }
Superclass:   | { "java.lang.Object " | [Browse...] }
}
@endsalt
```



14.10 添加选项卡

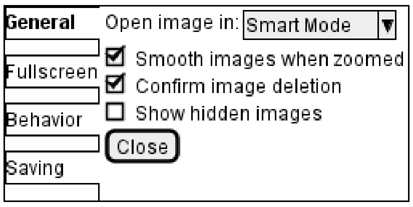
你可以通过 {/ 标记增加对应的选项卡。注意：可以使用 HTML 代码来增加粗体效果。

```
@startsalt
{+
{/ <b>General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



可以定义垂直选项卡，如下:

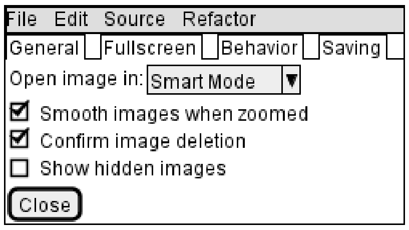
```
@startsalt
{+
{/ <b>General
Fullscreen
Behavior
Saving } |
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
[Close]
}
}
@endsalt
```



14.11 使用菜单

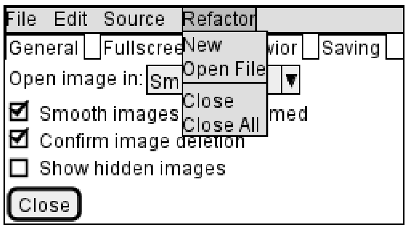
你可以使用记号 {* 来添加菜单。

```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



你也可以打开一个菜单:

```
@startsalt
{+
{* File | Edit | Source | Refactor
 Refactor | New | Open File | - | Close | Close All }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```


File Edit Source Refactor
General Fullscree New vior Saving
Open image in: Sm Open File
Close
Close All
Smooth images med
Confirm image deletion
Show hidden images
Close

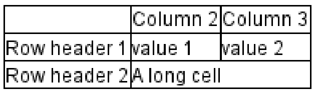
[Ref. [QA-4184](https://forum.plantuml.net/4184)]

14.12 高级表格

对于表格有两种特殊的标记:

- `*` 单元格同时具备 span 和 left 两个属性
- `.` 是空白单元格

```
@startsalt
{#
. | Column 2 | Column 3
Row header 1 | value 1 | value 2
Row header 2 | A long cell | *
}
@endsalt
```



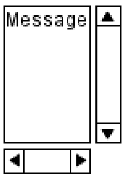
	Column 2	Column 3
Row header 1	value 1	value 2
Row header 2	A long cell	

14.13 Scroll Bars [S, SI, S-]

You can use `{S` notation for scroll bar like in following examples:

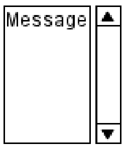
- `{S`: for horizontal and vertical scrollbars

```
@startsalt
{S
Message
.
.
.
.
}
@endsalt
```



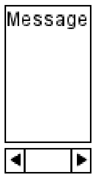
- `{SI` : for vertical scrollbar only

```
@startsalt
{SI
Message
.
.
.
.
}
@endsalt
```



- `{S-` : for horizontal scrollbar only

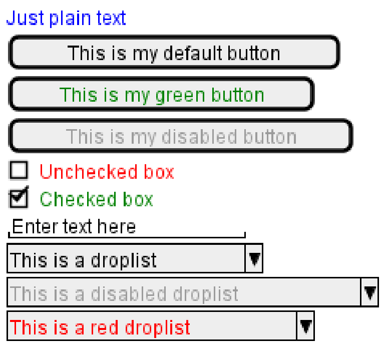
```
@startsalt
{S-
Message
.
.
.
.
}
@endsalt
```



14.14 Colors

It is possible to change text color of widget.

```
@startsalt
{
  <color:Blue>Just plain text
  [This is my default button]
  [<color:green>This is my green button]
  [<color:#9a9a9a>This is my disabled button]
  []  <color:red>Unchecked box
  [X] <color:green>Checked box
  "Enter text here   "
  ^This is a droplist^
  ^<color:#9a9a9a>This is a disabled droplist^
  ^<color:red>This is a red droplist^
}
@endsalt
```

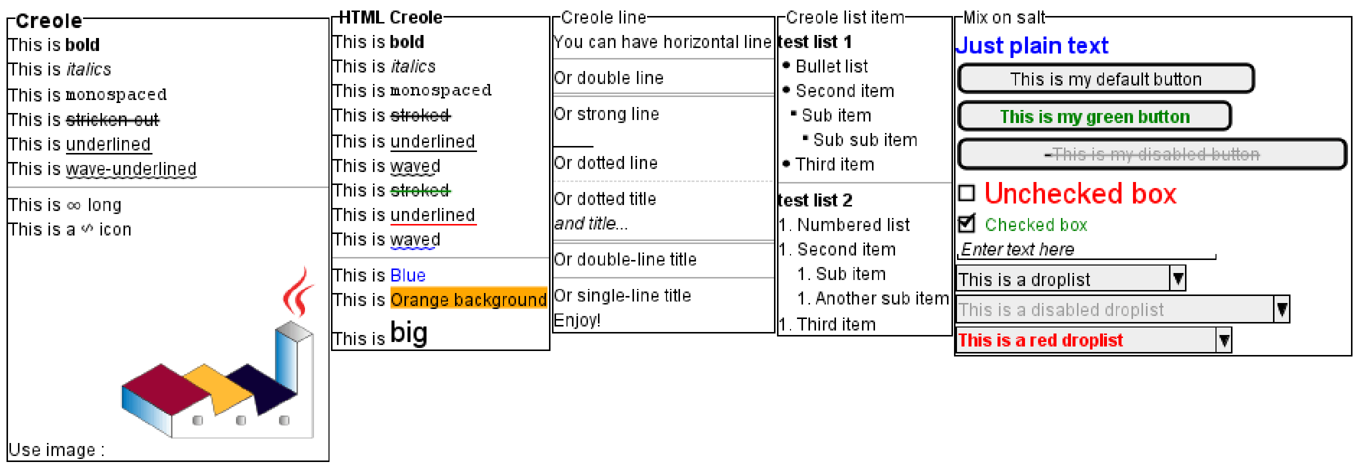


[Ref. QA-12177]

14.15 Creole on Salt

You can use Creole or HTML Creole on salt:

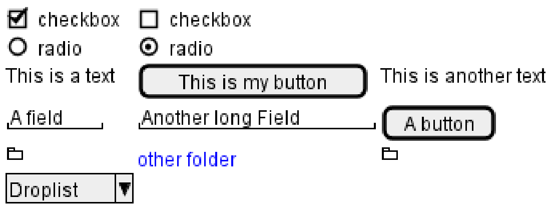
```
@startsalt
{{^==Creole
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
}|
{^<b>HTML Creole
 This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
  -- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
}|
{^Creole line
You can have horizontal line
----
Or double line
====
Or strong line

____
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}|
{^Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}|
{^Mix on salt
  ==<color:Blue>Just plain text
  [This is my default button]
  [<b><color:green>This is my green button]
  [ ---<color:#9a9a9a>This is my disabled button-- ]
  []  <size:20><color:red>Unchecked box
  [X] <color:green>Checked box
  "//Enter text here//   "
  ^This is a droplist^
  ^<color:#9a9a9a>This is a disabled droplist^
  ^<b><color:red>This is a red droplist^
}}
@endsalt
```



14.16 Pseudo sprite [«, »]

Using << and >> you can define a pseudo-sprite or sprite-like drawing and reusing it latter.

```
@startsalt
 {
 [X] checkbox|[] checkbox
 () radio | (X) radio
 This is a text|[This is my button]|This is another text
 "A field"|"Another long Field"|[A button]
 <<folder
 ............
 .XXXXX......
 .X...X......
 .XXXXXXXXXX.
 .X........X.
 .X........X.
 .X........X.
 .X........X.
 .XXXXXXXXXX.
 ............
 >>|<color:blue>other folder|<<folder>>
^Droplist^
}
@endsalt
```



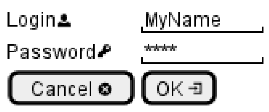
[Ref. QA-5849]

14.17 OpenIconic

OpenIconic is an very nice open source icon set. Those icons have been integrated into the creole parser, so you can use them out-of-the-box.

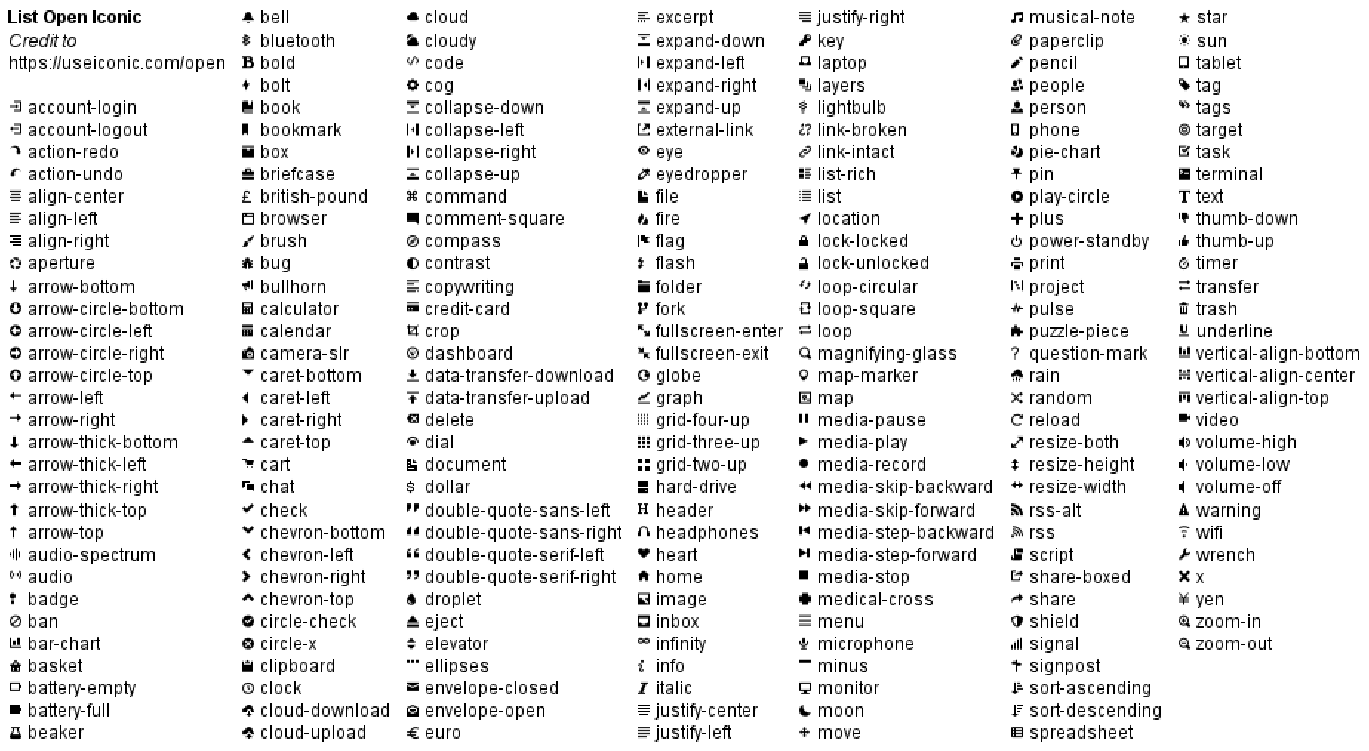
You can use the following syntax: `<&ICON_NAME>`.

```
@startsalt
{
  Login<&person> | "MyName   "
  Password<&key> | "****     "
  [Cancel <&circle-x>] | [OK <&account-login>]
}
@endsalt
```



The complete list is available on OpenIconic Website, or you can use the following special diagram:

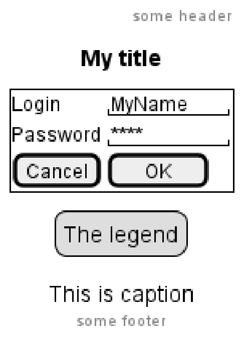
```
@startuml
listopeniconic
@enduml
```


List Open Iconic
Credit to
https://useiconic.com/open

account-login
account-logout
action-redo
action-undo
align-center
align-left
align-right
aperture
arrow-bottom
arrow-circle-bottom
arrow-circle-left
arrow-circle-right
arrow-circle-top
arrow-left
arrow-right
arrow-thick-bottom
arrow-thick-left
arrow-thick-right
arrow-thick-top
arrow-top
audio-spectrum
audio
badge
ban
bar-chart
basket
battery-empty
battery-full
beaker
bell
bluetooth
bold
bolt
book
bookmark
box
briefcase
british-pound
browser
brush
bug
bullhorn
calculator
calendar
camera-slr
caret-bottom
caret-left
caret-right
caret-top
cart
chat
check
chevron-bottom
chevron-left
chevron-right
chevron-top
circle-check
circle-x
clipboard
clock
cloud-download
cloud-upload
cloud
cloudy
code
cog
collapse-down
collapse-left
collapse-right
collapse-up
command
comment-square
compass
contrast
copywriting
credit-card
crop
dashboard
data-transfer-download
data-transfer-upload
delete
dial
document
dollar
double-quote-sans-left
double-quote-sans-right
double-quote-serif-left
double-quote-serif-right
droplet
eject
elevator
ellipses
envelope-closed
envelope-open
euro
excerpt
expand-down
expand-left
expand-right
expand-up
external-link
eye
eyedropper
file
fire
flag
flash
folder
fork
fullscreen-enter
fullscreen-exit
globe
graph
grid-four-up
grid-three-up
grid-two-up
hard-drive
header
headphones
heart
home
image
inbox
infinity
info
italic
justify-center
justify-left
justify-right
key
laptop
layers
lightbulb
link-broken
link-intact
list-rich
list
location
lock-locked
lock-unlocked
loop-circular
loop-square
loop
magnifying-glass
map-marker
map
media-pause
media-play
media-record
media-skip-backward
media-skip-forward
media-step-backward
media-step-forward
media-stop
medical-cross
menu
microphone
minus
monitor
moon
move
musical-note
paperclip
pencil
people
person
phone
pie-chart
pin
play-circle
plus
power-standby
print
project
pulse
puzzle-piece
question-mark
rain
random
reload
resize-both
resize-height
resize-width
rss-alt
rss
script
share-boxed
share
shield
signal
signpost
sort-ascending
sort-descending
spreadsheet
star
sun
tablet
tag
tags
target
task
terminal
text
thumb-down
thumb-up
timer
transfer
trash
underline
vertical-align-bottom
vertical-align-center
vertical-align-top
video
volume-high
volume-low
volume-off
warning
wifi
wrench
x
yen
zoom-in
zoom-out

14.18 Add title, header, footer, caption or legend

```
@startsalt
title My title
header some header
footer some footer
caption This is caption
legend
The legend
end legend

{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}

@endsalt
```

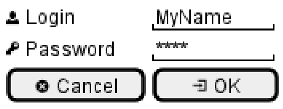


(See also: Common commands)

14.19 Zoom, DPI

14.19.1 Whitout zoom (by default)

```
@startsalt
{
  <&person> Login  | "MyName   "
  <&key> Password  | "****     "
  [<&circle-x> Cancel ] | [ <&account-login> OK   ]
}
@endsalt
```

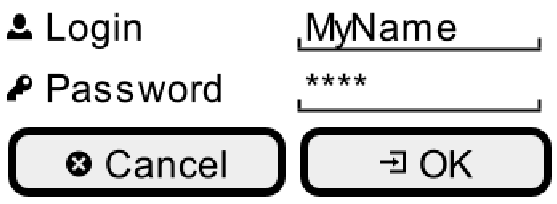


14.19.2 Scale

You can use the `scale` command to zoom the generated image.

You can use either a number or a fraction to define the scale factor. You can also specify either width or height (in pixel). And you can also give both width and height: the image is scaled to fit inside the specified dimension.

```
@startsalt
scale 2
{
  <&person> Login  | "MyName   "
  <&key> Password  | "****     "
  [<&circle-x> Cancel ] | [ <&account-login> OK   ]
}
@endsalt
```

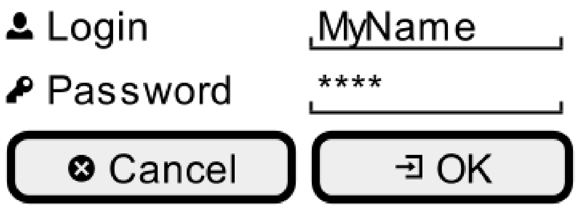


(See also: Zoom on Common commands)

14.19.3 DPI

You can also use the `skinparam dpi`command to zoom the generated image.

```
@startsalt
skinparam dpi 200
{
  <&person> Login  | "MyName   "
  <&key> Password  | "****     "
  [<&circle-x> Cancel ] | [ <&account-login> OK   ]
}
@endsalt
```

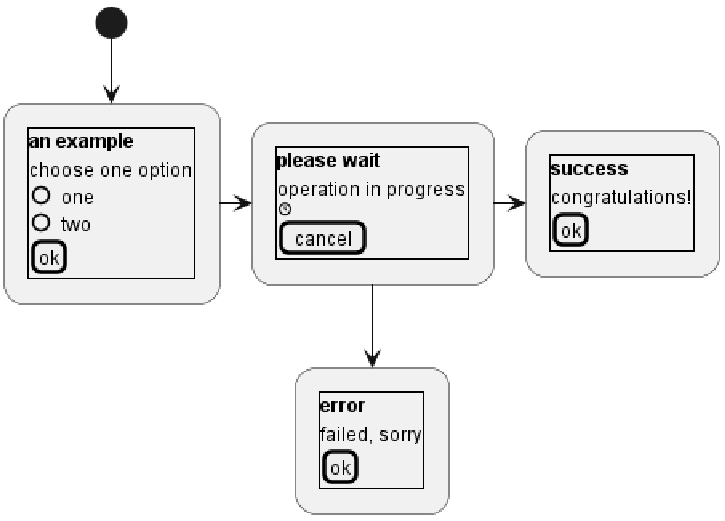


14.20 Include Salt "on activity diagram"

You can read the following explanation.

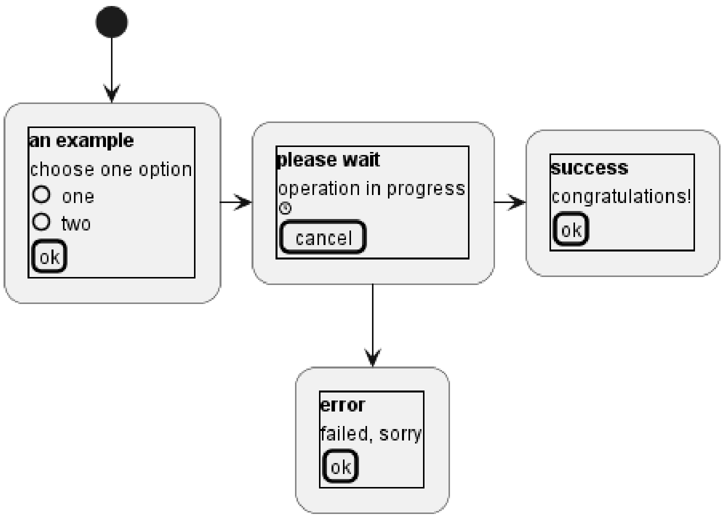
```
@startuml
(*) --> "
{{
salt
{+
<b>an example
choose one option
()one
()two
[ok]
}
}}
" as choose

choose -right-> "
{{
salt
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
}}
" as wait
wait -right-> "
{{
salt
{+
<b>success
congratulations!
[ok]
}
}}
" as success

wait -down-> "
{{
salt
{+
<b>error
failed, sorry
[ok]
}
}}
"
```

```
@enduml
```



It can also be combined with define macro.

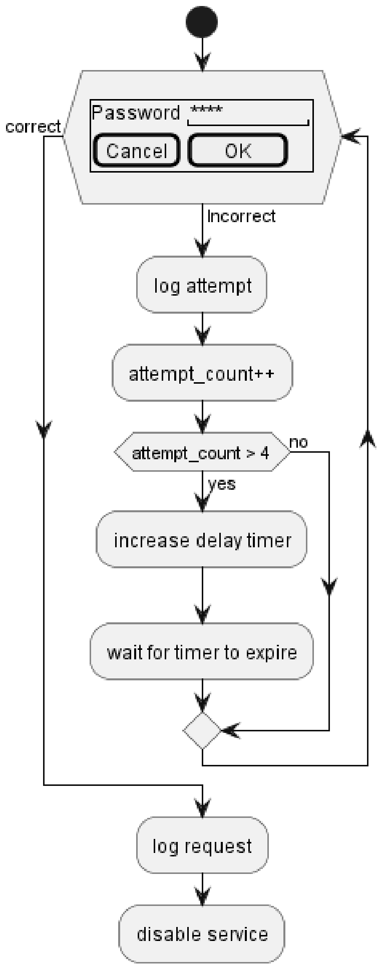
```
@startuml
!unquoted procedure SALT($x)
"{{
salt
%invoke_procedure("_"+$x)
}}" as $x
!endprocedure

!procedure _choose()
{+
<b>an example
choose one option
()one
()two
[ok]
}
!endprocedure

!procedure _wait()
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!endprocedure

!procedure _success()
{+
<b>success
congratulations!
[ok]
}
!endprocedure

!procedure _error()
{+
<b>error
failed, sorry
[ok]
}
!endprocedure

(*) --> SALT(choose)
-right-> SALT(wait)
wait -right-> SALT(success)
wait -down-> SALT(error)
@enduml
```



14.21 Include salt ”on while condition of activity diagram”

You can include `salt` on while condition of activity diagram.

```
@startuml
start
while (\n{{\nsalt\n{+\nPassword | "****     "\n[Cancel] | [  OK   ]}\n}}\n) is (Incorrect)
  :log attempt;
  :attempt_count++;
  if (attempt_count > 4) then (yes)
    :increase delay timer;
    :wait for timer to expire;
  else (no)
  endif
endwhile (correct)
:log request;
:disable service;
@enduml
```

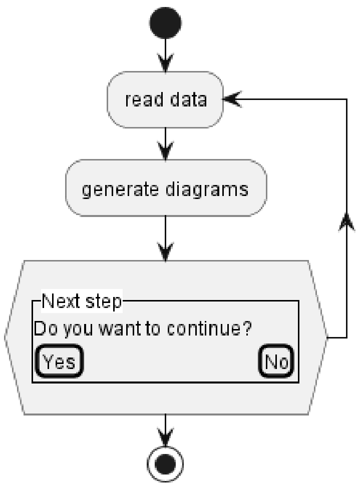


[Ref. QA-8547]

14.22 Include salt ”on repeat while condition of activity diagram”

You can include `salt` on ’repeat while’ condition of activity diagram.

```
@startuml
start
repeat :read data;
  :generate diagrams;
repeat while (\n{{\nsalt\n{^"Next step"\n  Do you want to continue? \n[Yes]|[No]\n}\n}}\n)
stop
@enduml
```



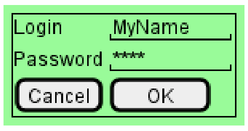
[Ref. QA-14287]

14.23 Skinparam

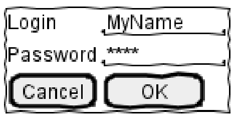
You can use [**only**] some skinparam command to change the skin of the drawing.

Some example:

```
@startsalt
skinparam Backgroundcolor palegreen
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```

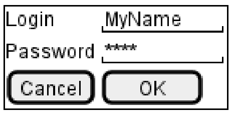


```
@startsalt
skinparam handwritten true
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



TODO: FIXME FYI, some other skinparam does not work with salt, as:

```
@startsalt
skinparam defaultFontName monospaced
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```

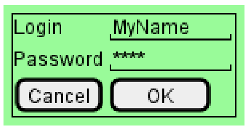


14.24 Style

You can use [**only**] some style command to change the skin of the drawing.

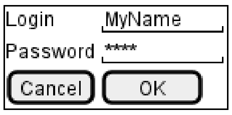
Some example:

```
@startsalt
<style>
saltDiagram {
  BackgroundColor palegreen
}
</style>
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



TODO: FIXME FYI, some other style does not work with salt, as:

```
@startsalt
<style>
saltDiagram {
  Fontname Monospaced
  FontSize 10
  FontStyle italic
  LineThickness 0.5
  LineColor red
}
</style>
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



[Ref. QA-13460]

15 Archimate Diagram

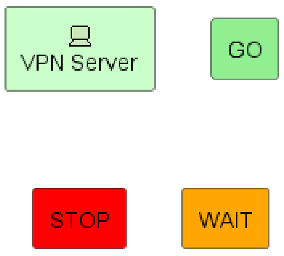
This is only a proposal and subject to change.

You are very welcome to create a new discussion on this future syntax. Your feedbacks, ideas and suggestions help us to find the right solution.

15.1 Archimate keyword

You can use the `archimate` keyword to define an element. Stereotype can optionally specify an additional icon. Some colors (`Business`, `Application`, `Motivation`, `Strategy`, `Technology`, `Physical`, `Implementation`) are also available.

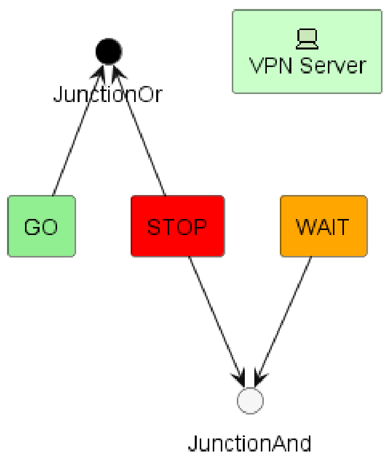
```
@startuml
archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```



15.2 定义结点

使用 `circle` 关键字和预处理程序，你也可以创建结点。

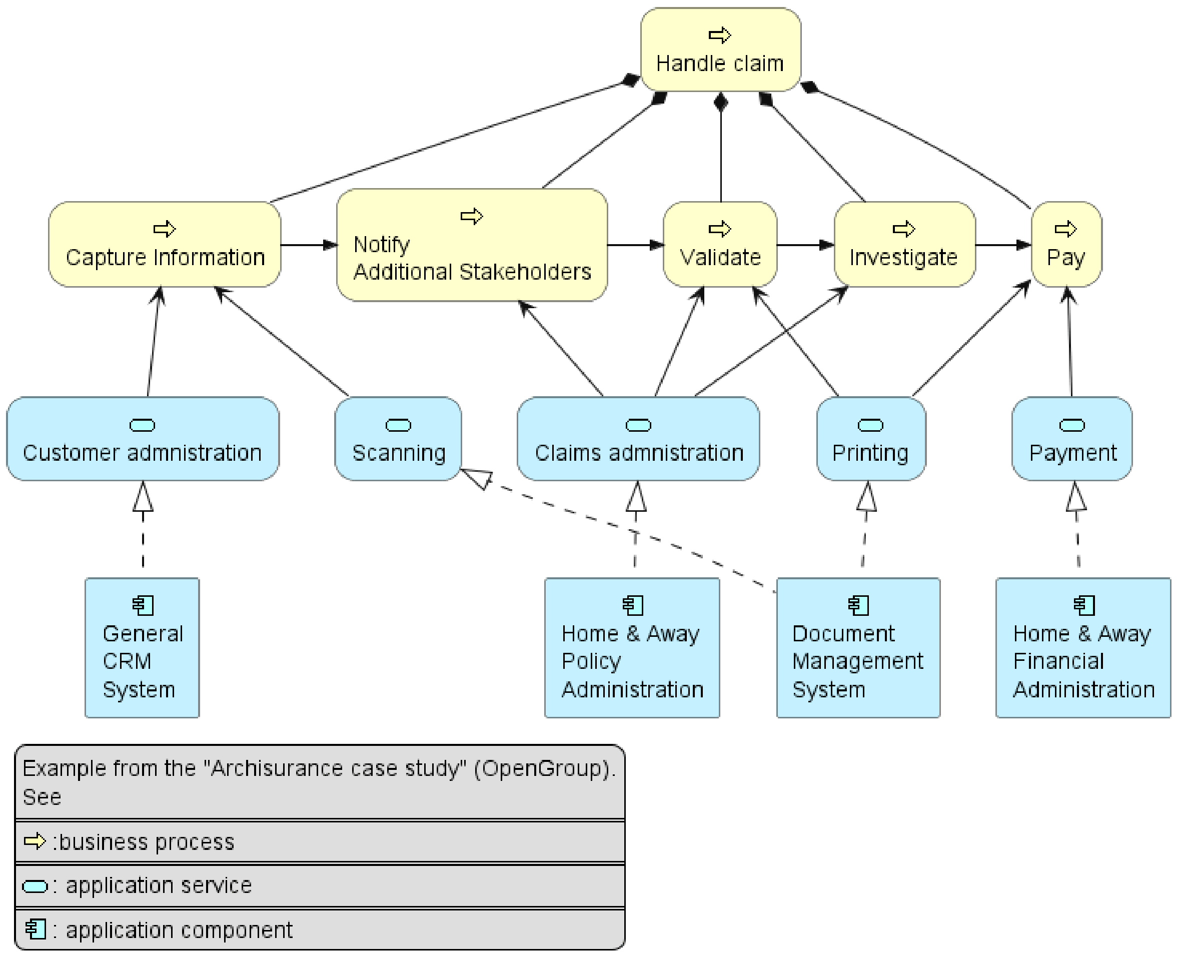
```
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

Junction_And JunctionAnd
Junction_Or JunctionOr

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
GO -up-> JunctionOr
STOP -up-> JunctionOr
STOP -down-> JunctionAnd
WAIT -down-> JunctionAnd
@enduml
```



15.3 例 1

```
@startuml
skinparam rectangle<<behavior>> {
roundCorner 25
}
sprite $bProcess jar:archimate/business-process
sprite $aService jar:archimate/application-service
sprite $aComponent jar:archimate/application-component

rectangle "Handle claim"  as HC <<$bProcess>><<behavior>> #Business
rectangle "Capture Information"  as CI <<$bProcess>><<behavior>> #Business
rectangle "Notify\nAdditional Stakeholders" as NAS <<$bProcess>><<behavior>> #Business
rectangle "Validate" as V <<$bProcess>><<behavior>> #Business
rectangle "Investigate" as I <<$bProcess>><<behavior>> #Business
rectangle "Pay" as P <<$bProcess>><<behavior>> #Business

HC *-down- CI
HC *-down- NAS
HC *-down- V
HC *-down- I
HC *-down- P

CI -right->> NAS
NAS -right->> V
V -right->> I
I -right->> P

rectangle "Scanning" as scanning <<$aService>><<behavior>> #Application
rectangle "Customer admnistration" as customerAdministration <<$aService>><<behavior>> #Application
rectangle "Claims admnistration" as claimsAdministration <<$aService>><<behavior>> #Application
rectangle Printing <<$aService>><<behavior>> #Application
rectangle Payment <<$aService>><<behavior>> #Application

scanning -up-> CI
customerAdministration  -up-> CI
claimsAdministration -up-> NAS
claimsAdministration -up-> V
claimsAdministration -up-> I
Payment -up-> P

Printing -up-> V
Printing -up-> P
```

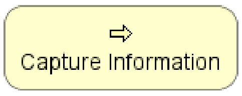
```
rectangle "Document\nManagement\nSystem" as DMS <<$aComponent>> #Application
rectangle "General\nCRM\nSystem" as CRM <<$aComponent>>  #Application
rectangle "Home & Away\nPolicy\nAdministration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\nFinancial\nAdministration" as HFPA <<$aComponent>>  #Application

DMS .up.|> scanning
DMS .up.|> Printing
CRM .up.|> customerAdministration
HAPA .up.|> claimsAdministration
HFPA .up.|> Payment

legend left
Example from the "Archisurance case study" (OpenGroup).
See
====
<$bProcess> :business process
====
<$aService> : application service
====
<$aComponent> : application component
endlegend
@enduml
```



15.4 Example 2

```
@startuml
```

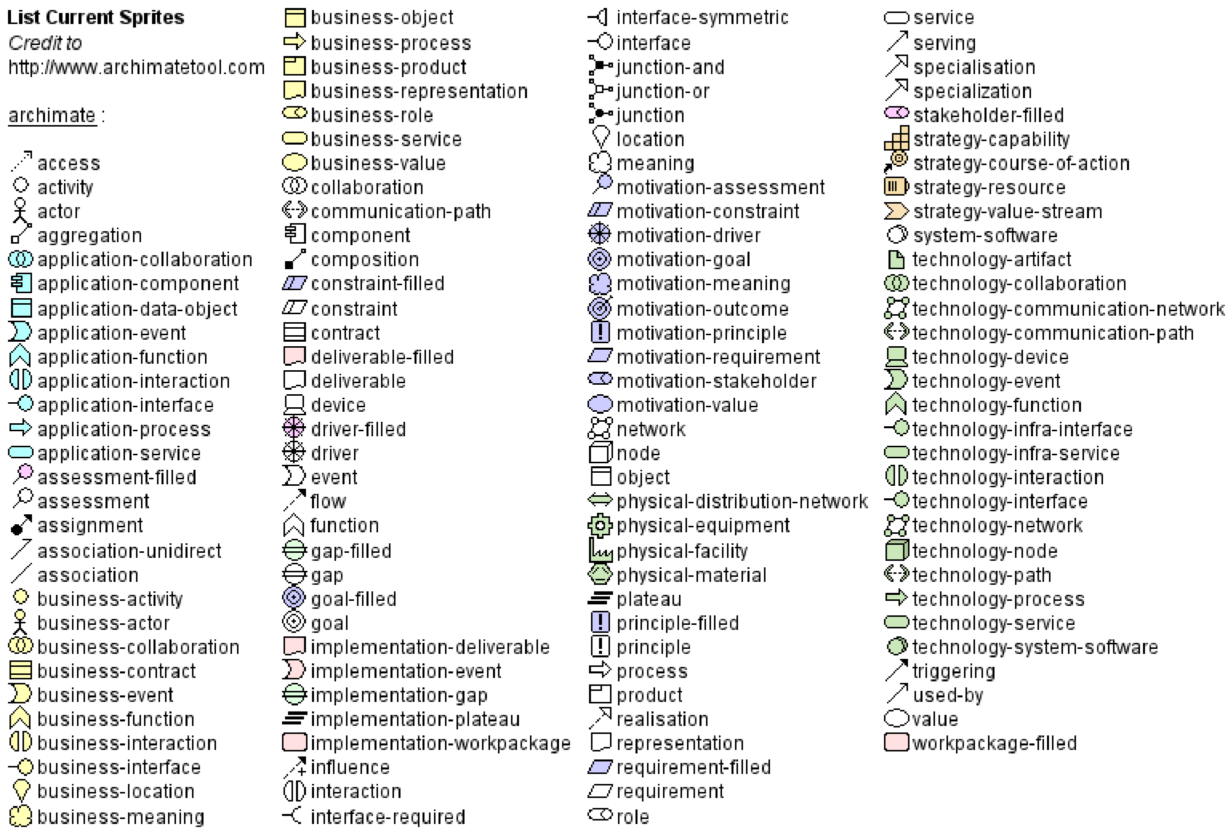
```
skinparam roundcorner 25
rectangle "Capture Information"  as CI <<$archimate/business-process>> #Business
@enduml
```



15.5 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```



List Current Sprites
Credit to
http://www.archimatetool.com

archimate :

access
activity
actor
aggregation
application-collaboration
application-component
application-data-object
application-event
application-function
application-interaction
application-interface
application-process
application-service
assessment-filled
assessment
assignment
association-unidirect
association
business-activity
business-actor
business-collaboration
business-contract
business-event
business-function
business-interaction
business-interface
business-location
business-meaning
business-object
business-process
business-product
business-representation
business-role
business-service
business-value
collaboration
communication-path
component
composition
constraint-filled
constraint
contract
deliverable-filled
deliverable
device
driver-filled
driver
event
flow
function
gap-filled
gap
goal-filled
goal
implementation-deliverable
implementation-event
implementation-gap
implementation-plateau
implementation-workpackage
influence
interaction
interface-required
interface-symmetric
interface
junction-and
junction-or
junction
location
meaning
motivation-assessment
motivation-constraint
motivation-driver
motivation-goal
motivation-meaning
motivation-outcome
motivation-principle
motivation-requirement
motivation-stakeholder
motivation-value
network
node
object
physical-distribution-network
physical-equipment
physical-facility
physical-material
plateau
principle-filled
principle
process
product
realisation
representation
requirement-filled
requirement
role
service
serving
specialisation
specialization
stakeholder-filled
strategy-capability
strategy-course-of-action
strategy-resource
strategy-value-stream
system-software
technology-artifact
technology-collaboration
technology-communication-network
technology-communication-path
technology-device
technology-event
technology-function
technology-infra-interface
technology-infra-service
technology-interaction
technology-interface
technology-network
technology-node
technology-path
technology-process
technology-service
technology-system-software
triggering
used-by
value
workpackage-filled

15.6 ArchiMate Macros

15.6.1 Archimate Macros and Library

A list of Archimate macros are defined Archimate-PlantUML here which simplifies the creation of ArchiMate diagrams, and Archimate is natively on the Standard Library of PlantUML.

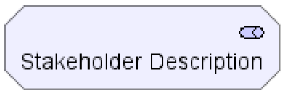
15.6.2 Archimate elements

Using the macros, creation of ArchiMate elements are done using the following format: `Category_ElementName(nameOfThe` `"description")`

For example:

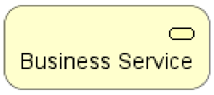
- To define a *Stakeholder* element, which is part of Motivation category, the syntax will be `Motivation_Stakeholder(S` `"Stakeholder Description")`:

```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
@enduml
```



- To define a *Business Service* element, `Business_Service(BService, "Business Service")`:

```
@startuml
!include <archimate/Archimate>
Business_Service(BService, "Business Service")
@enduml
```



15.6.3 Archimate relationships

The ArchiMate relationships are defined with the following pattern: `Rel_RelationType(fromElement, toElement, "description")` and to define the direction/orientation of the two elements: `Rel_RelationType_Direction toElement, "description")`

The `RelationTypes` supported are:

- Access
- Aggregation
- Assignment
- Association
- Composition
- Flow
- Influence
- Realization
- Serving
- Specialization
- Triggering

The `Directions` supported are:

- Up
- Down
- Left
- Right

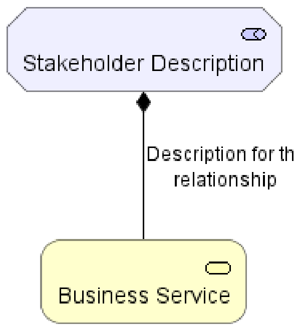
For example:

- To denote a composition relationship between the *Stakeholder* and *Business Service* defined above, the syntax will be

```
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
```

```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
```

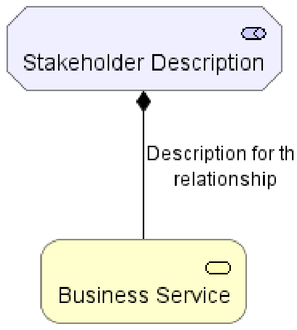
```
Business_Service(BService, "Business Service")
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml
```



- Unordered List ItemTo orient the two elements in top - down position, the syntax will be

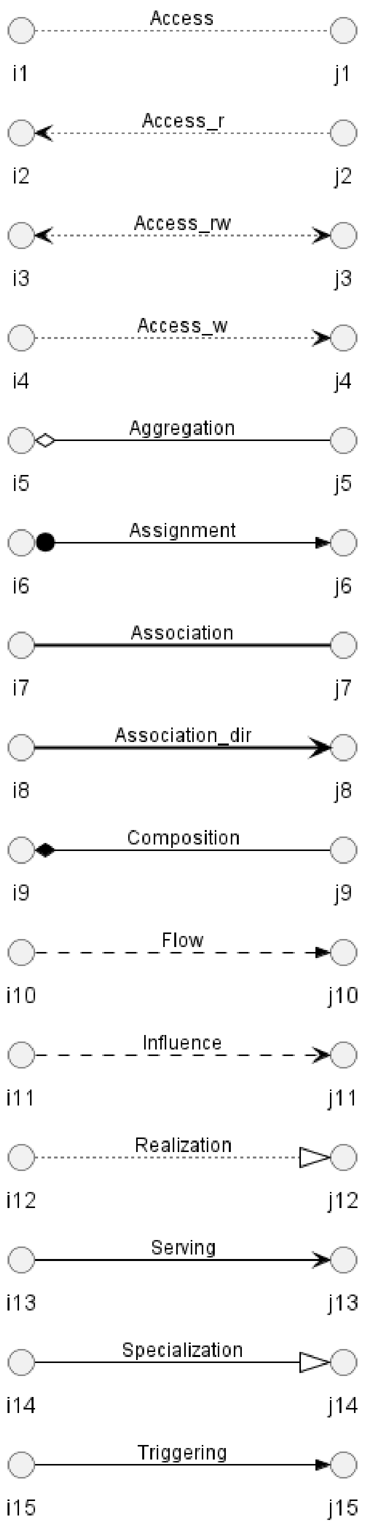
```
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
```

```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
@enduml
```

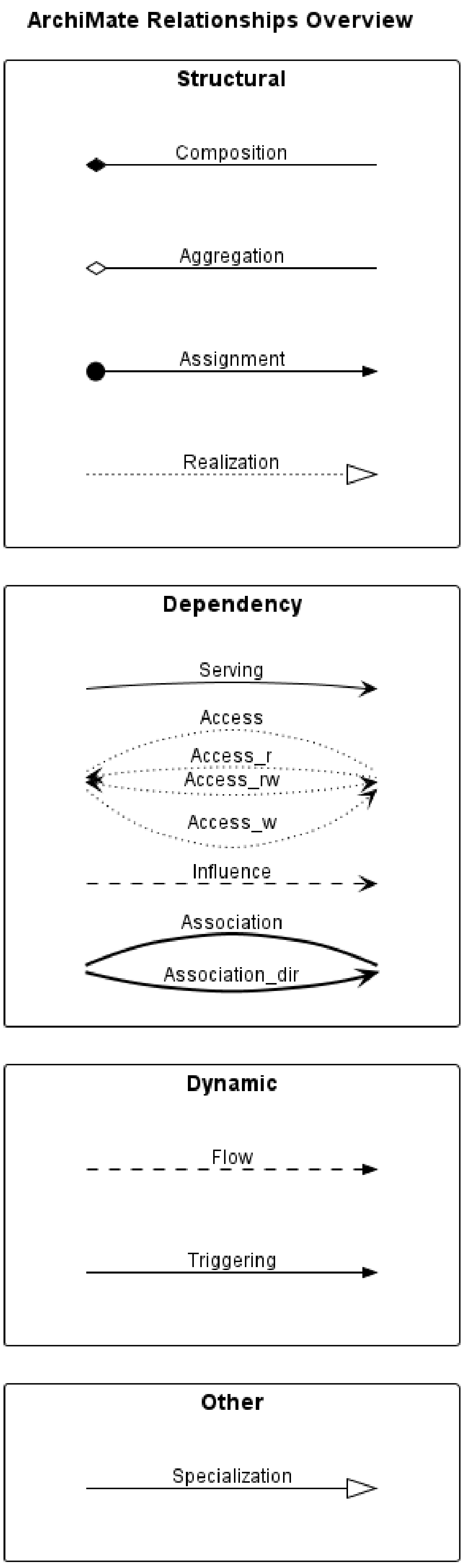


15.6.4 Appendice: Examples of all Archimate RelationTypes

```
@startuml
left to right direction
skinparam nodesep 4
!include <archimate/Archimate>
Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i8, j8, Association_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i3, j3, Access_rw)
Rel_Access_r(i2, j2, Access_r)
Rel_Access(i1, j1, Access)
@enduml
```



```
@startuml
title ArchiMate Relationships Overview
skinparam nodesep 5
<style>
interface {
    shadowing 0
    backgroundcolor transparent
    linecolor transparent
    FontColor transparent
}
</style>
!include <archimate/Archimate>
left to right direction

rectangle Other {
() i14
() j14
}


rectangle Dynamic {
() i10
() j10
() i15
() j15
}

rectangle Dependency {
() i13
() j13
() i4
() j4
() i11
() j11
() i7
() j7
}

rectangle Structural {
() i9
() j9
() i5
() j5
() i6
() j6
() i12
() j12
}

Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i7, j7, \nAssociation_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i4, j4, Access_rw)
Rel_Access_r(i4, j4, Access_r)
Rel_Access(i4, j4, Access)
@enduml
```



[Adapted from Archimate PR#25]

16 甘特图

甘特图是用自然语言描述的，使用非常简单的句子（主语-动词-补语）。

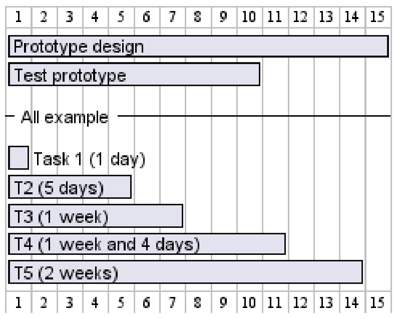
16.1 声明任务

使用中括号来定义任务。

16.1.1 时长

通过试用动词 `last` 来定义时长。

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
-- All example --
[Task 1 (1 day)] lasts 1 day
[T2 (5 days)] lasts 5 days
[T3 (1 week)] lasts 1 week
[T4 (1 week and 4 days)] lasts 1 week and 4 days
[T5 (2 weeks)] lasts 2 weeks
@endgantt
```

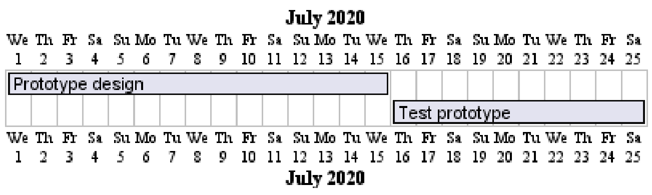


所谓的一周，是指一周内非关闭日（即一周内的工作日）。如果你指定了周六、周日不上工，那么一周就等于 5 天。

16.1.2 开始

通过 `start` 动词来定义开始。

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days

Project starts 2020-07-01
[Prototype design] starts 2020-07-01
[Test prototype] starts 2020-07-16
@endgantt
```

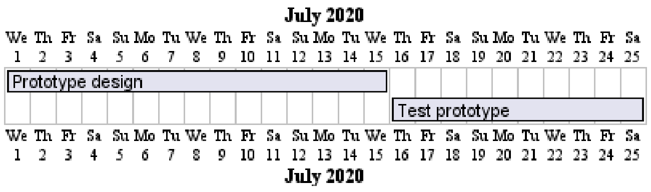


16.1.3 结束

通过 `end` 动词来定义结束。

```
@startgantt
```

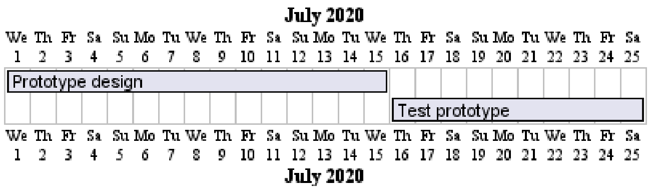
```
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days

Project starts 2020-07-01
[Prototype design] ends 2020-07-15
[Test prototype] ends 2020-07-25

@endgantt
```



16.1.4 开始/结束

可以通过指定具体的日期来定义开始、结束点。

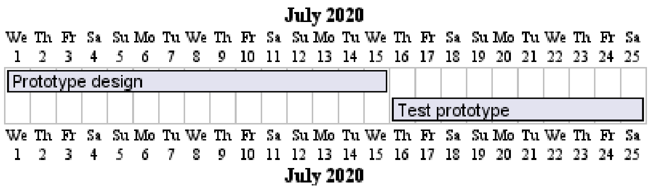
```
@startgantt

Project starts 2020-07-01
[Prototype design] starts 2020-07-01
[Test prototype] starts 2020-07-16
[Prototype design] ends 2020-07-15
[Test prototype] ends 2020-07-25

@endgantt
```



16.2 单行声明（用 and 连词）

可以将声明与 and 连词组合在一行。

```
@startgantt
Project starts 2020-07-01
[Prototype design] starts 2020-07-01 and ends 2020-07-15
[Test prototype] starts 2020-07-16 and lasts 10 days
@endgantt
```

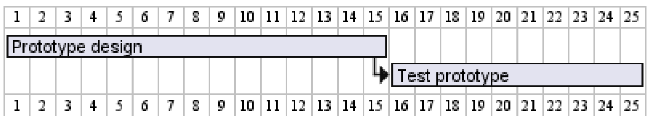


16.3 添加约束

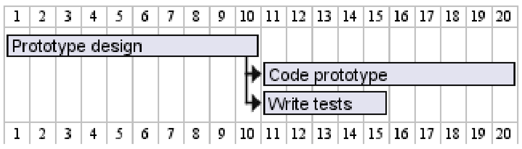
可以在任务之间添加约束。

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
[Test prototype] starts at [Prototype design]'s end
```

```
@endgantt
```



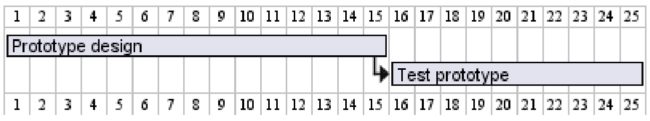
```
@startgantt
[Prototype design] lasts 10 days
[Code prototype] lasts 10 days
[Write tests] lasts 5 days
[Code prototype] starts at [Prototype design]'s end
[Write tests] starts at [Code prototype]'s start
@endgantt
```



16.4 简称

可以通过使用 as 关键字给任务设置简称。

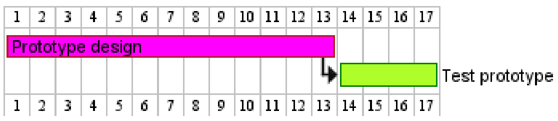
```
@startgantt
[Prototype design] as [D] lasts 15 days
[Test prototype] as [T] lasts 10 days
[T] starts at [D]'s end
@endgantt
```



16.5 自定义配色

可以通过使用 is colored in 来自定义颜色。

```
@startgantt
[Prototype design] lasts 13 days
[Test prototype] lasts 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



16.6 完成状态

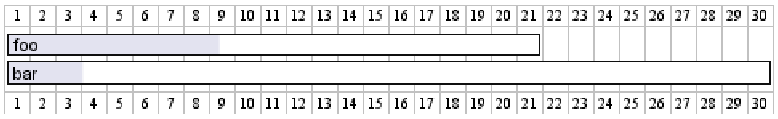
16.6.1 添加完成度百分比

可以给任务设置完成度状态，通过：

- is xx% completed
- is xx% complete

```
@startgantt
[foo] lasts 21 days
```

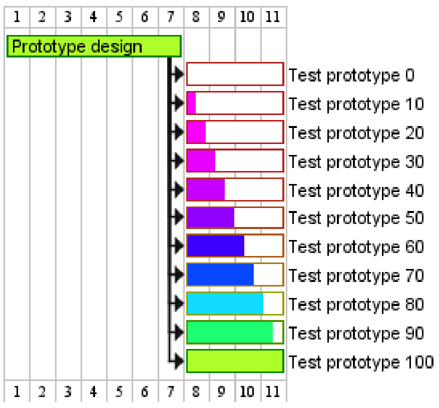
```
[foo] is 40% completed
[bar] lasts 30 days and is 10% complete
@endgantt
```



16.6.2 改变完成颜色 (通过样式)

```
@startgantt

<style>
ganttDiagram {
  task {
    BackGroundColor GreenYellow
    LineColor Green
    unstarted {
      BackGroundColor Fuchsia
      LineColor FireBrick
    }
  }
}
</style>

[Prototype design] lasts 7 days
[Test prototype 0] lasts 4 days
[Test prototype 10] lasts 4 days
[Test prototype 20] lasts 4 days
[Test prototype 30] lasts 4 days
[Test prototype 40] lasts 4 days
[Test prototype 50] lasts 4 days
[Test prototype 60] lasts 4 days
[Test prototype 70] lasts 4 days
[Test prototype 80] lasts 4 days
[Test prototype 90] lasts 4 days
[Test prototype 100] lasts 4 days

[Test prototype 0] starts at [Prototype design]'s end
[Test prototype 10] starts at [Prototype design]'s end
[Test prototype 20] starts at [Prototype design]'s end
[Test prototype 30] starts at [Prototype design]'s end
[Test prototype 40] starts at [Prototype design]'s end
[Test prototype 50] starts at [Prototype design]'s end
[Test prototype 60] starts at [Prototype design]'s end
[Test prototype 70] starts at [Prototype design]'s end
[Test prototype 80] starts at [Prototype design]'s end
[Test prototype 90] starts at [Prototype design]'s end
[Test prototype 100] starts at [Prototype design]'s end

[Test prototype 0] is 0% complete
[Test prototype 10] is 10% complete
[Test prototype 20] is 20% complete
[Test prototype 30] is 30% complete
[Test prototype 40] is 40% complete
[Test prototype 50] is 50% complete
[Test prototype 60] is 60% complete
[Test prototype 70] is 70% complete
```

```
[Test prototype 80] is 80% complete
[Test prototype 90] is 90% complete
[Test prototype 100] is 100% complete

@endgantt
```



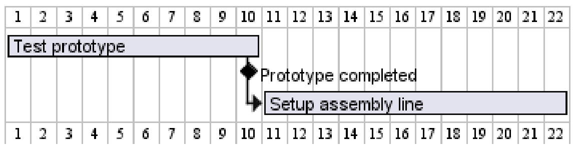
[Ref. QA-8297]

16.7 里程碑

通过使用 happen 动词设置里程碑.

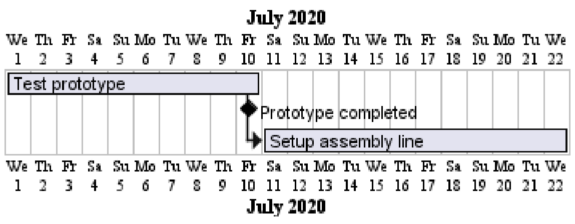
16.7.1 关联性里程碑 (使用约束)

```
@startgantt
[Test prototype] lasts 10 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] lasts 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



16.7.2 绝对里程碑 (使用固定日期)

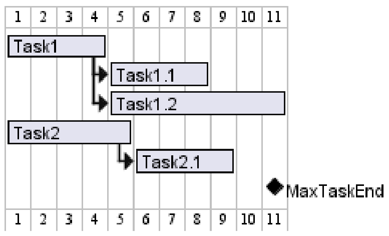
```
@startgantt
Project starts 2020-07-01
[Test prototype] lasts 10 days
[Prototype completed] happens 2020-07-10
[Setup assembly line] lasts 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



16.7.3 最晚截止任务的里程碑

```
@startgantt
```

```
[Task1] lasts 4 days
then [Task1.1] lasts 4 days
[Task1.2] starts at [Task1]'s end and lasts 7 days

[Task2] lasts 5 days
then [Task2.1] lasts 4 days

[MaxTaskEnd] happens at [Task1.1]'s end
[MaxTaskEnd] happens at [Task1.2]'s end
[MaxTaskEnd] happens at [Task2.1]'s end

@endgantt
```

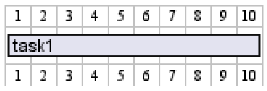


[Ref. QA-10764]

16.8 超链接

可以给任务添加超链接

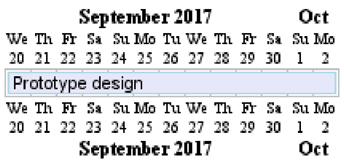
```
@startgantt
[task1] lasts 10 days
[task1] links to [[http://plantuml.com]]
@endgantt
```



16.9 日历

可以为整个项目指定开始日期。默认情况下，是第一个任务的开始日期。

```
@startgantt
Project starts the 20th of september 2017
[Prototype design] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue
@endgantt
```

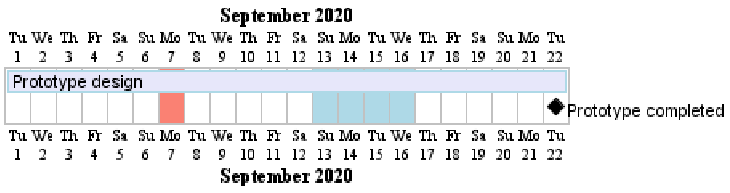


16.10 日期上色

可以对某些日期 [上色](上色)。

```
@startgantt
Project starts the 2020/09/01

2020/09/07 is colored in salmon
2020/09/13 to 2020/09/16 are colored in lightblue
```

```
[Prototype design] as [TASK1] lasts 22 days
[TASK1] is colored in Lavender/LightBlue
[Prototype completed] happens at [TASK1]'s end
@endgantt
```



16.11 改变比例

对于长期项目，可以改变比例，使用如下参数：

- printscale （图片比例）
- ganttscale （甘特图比例）
- projectscale （项目比例）

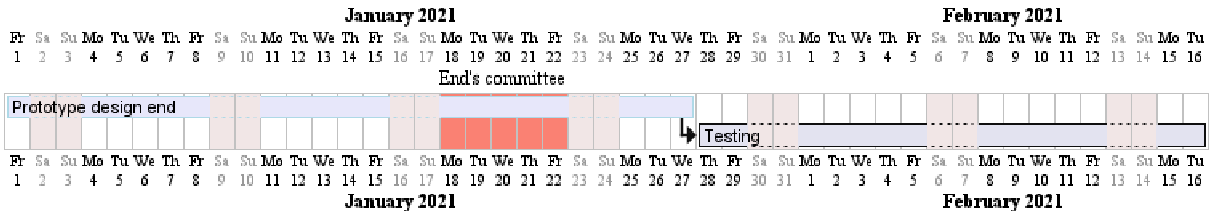
和如下值：

- daily *(by default)*
- weekly
- monthly
- quarterly
- yearly

(参考 QA-11272, QA-9041 和 QA-10948)

16.11.1 每天 Daily *(默认)*

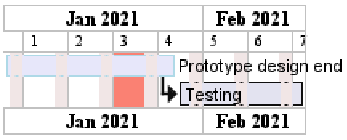
```
@startgantt
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



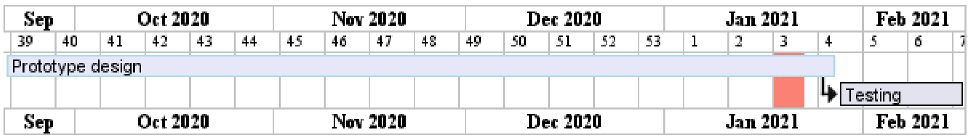
16.11.2 每周 Weekly

```
@startgantt
printscale weekly
saturday are closed
```

```
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```

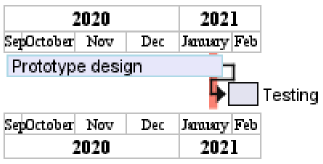


```
@startgantt
printscale weekly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.11.3 每月 Monthly

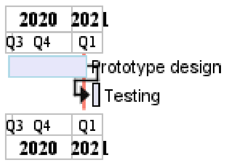
```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



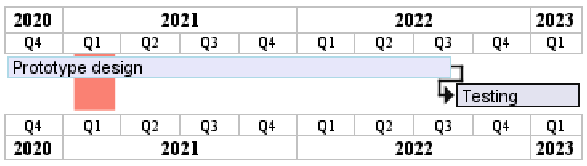
16.11.4 每季度 Quarterly

```
@startgantt
projectscale quarterly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
```

```
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```

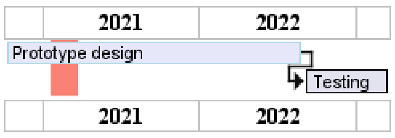


```
@startgantt
projectscale quarterly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt
```



16.11.5 每年 Yearly

```
@startgantt
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt
```



16.12 缩放 (例子对所有的比例)

可以改变缩放比例，通过使用参数:

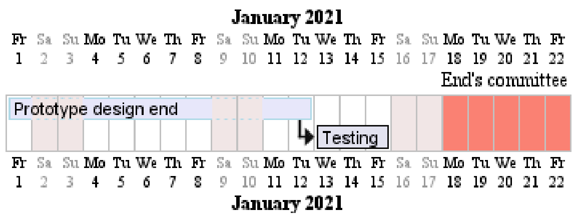
- `zoom <integer>`

16.12.1 日比例缩放

16.12.2 无缩放

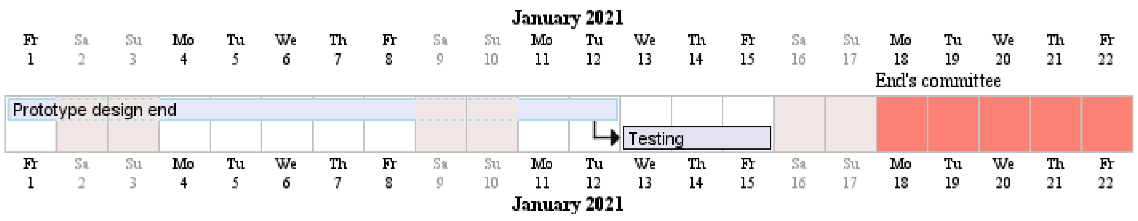
```
@startgantt
printscale daily
saturday are closed
```

```
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 8 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 3 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.12.3 有缩放

```
@startgantt
printscale daily zoom 2
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 8 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 3 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



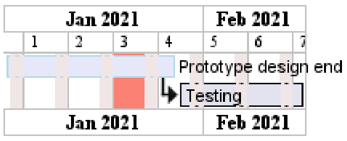
[Ref. QA-13725]

16.12.4 周比例缩放

16.12.5 无缩放

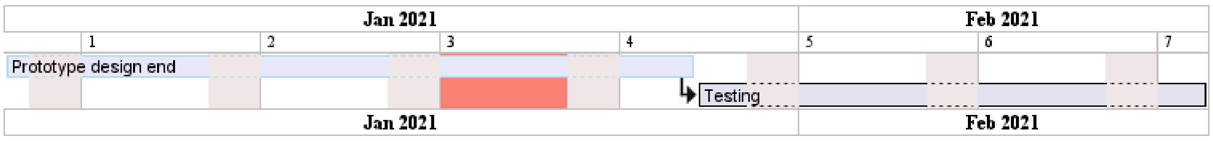
```
@startgantt
printscale weekly
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]
```

```
2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.12.6 有缩放

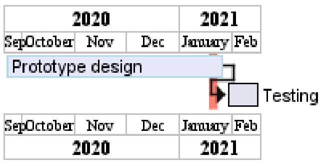
```
@startgantt
printscale weekly zoom 4
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.12.7 月比例缩放

16.12.8 无缩放

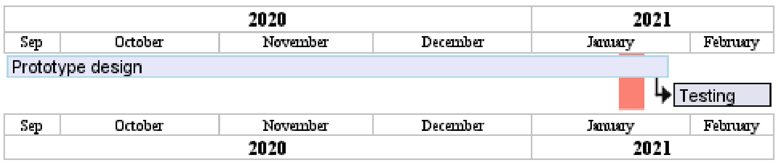
```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.12.9 有缩放

```
@startgantt
projectscale monthly zoom 3
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
```

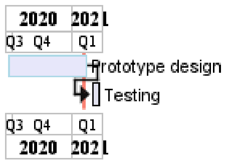
```
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.12.10 季度比例缩放

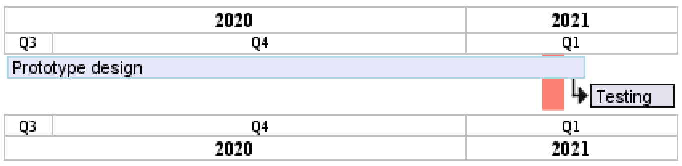
16.12.11 无缩放

```
@startgantt
projectscale quarterly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.12.12 有缩放

```
@startgantt
projectscale quarterly zoom 7
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```

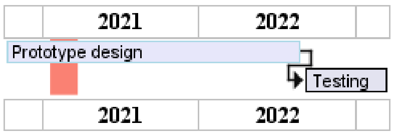


16.12.13 年比例缩放

16.12.14 无缩放

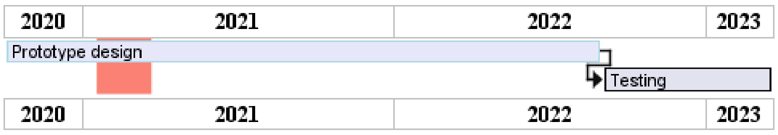
```
@startgantt
```

```
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt
```



16.12.15 有缩放

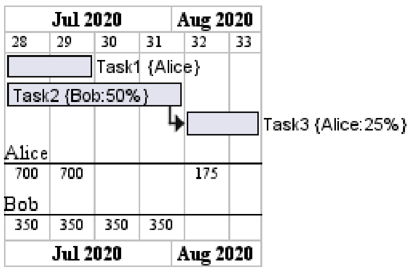
```
@startgantt
projectscale yearly zoom 2
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt
```



16.13 Weekscale with Weeknumbers or Calendar Date

16.13.1 With Weeknumbers *(by default)*

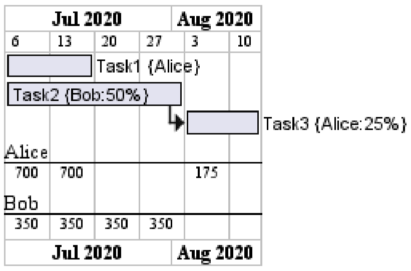
```
@startgantt
printscale weekly
Project starts the 6th of July 2020
[Task1] on {Alice} lasts 2 weeks
[Task2] on {Bob:50%} lasts 2 weeks
then [Task3] on {Alice:25%} lasts 3 days
@endgantt
```



16.13.2 With Calendar Date

```
@startgantt
printscale weekly with calendar date
Project starts the 6th of July 2020
[Task1] on {Alice} lasts 2 weeks
```

```
[Task2] on {Bob:50%} lasts 2 weeks
then [Task3] on {Alice:25%} lasts 3 days
@endgantt
```

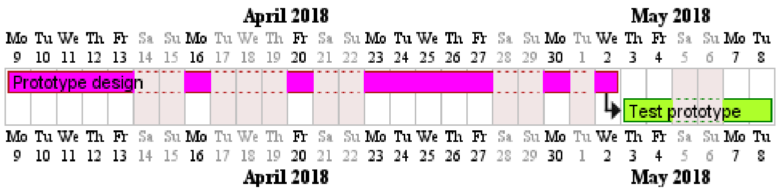


[Ref. QA-11630]

16.14 Close day

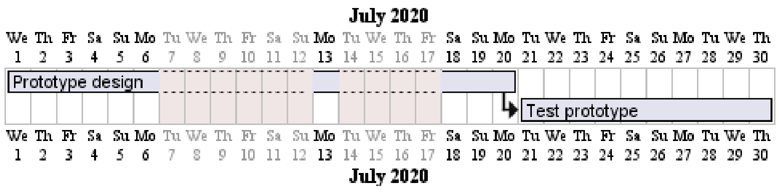
It is possible to close some day.

```
@startgantt
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] lasts 14 days
[Test prototype] lasts 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



Then it is possible to open some closed day.

```
@startgantt
2020-07-07 to 2020-07-17 is closed
2020-07-13 is open

Project starts the 2020-07-01
[Prototype design] lasts 10 days
Then [Test prototype] lasts 10 days
@endgantt
```

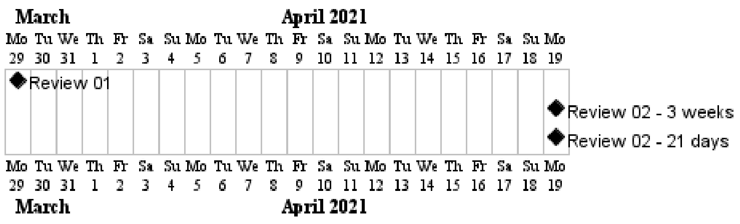


16.15 Definition of a week depending of closed days

A **week** is a synonym for how many non-closed days are in a week, as:

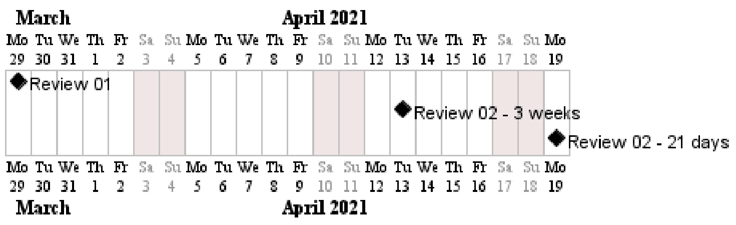
```
@startgantt
```

```
Project starts 2021-03-29
[Review 01] happens at 2021-03-29
[Review 02 - 3 weeks] happens on 3 weeks after [Review 01]'s end
[Review 02 - 21 days] happens on 21 days after [Review 01]'s end
@endgantt
```



So if you specify *Saturday* and *Sunday* as closed, a **week** will be equivalent to 5 days, as:

```
@startgantt
Project starts 2021-03-29
saturday are closed
sunday are closed
[Review 01] happens at 2021-03-29
[Review 02 - 3 weeks] happens on 3 weeks after [Review 01]'s end
[Review 02 - 21 days] happens on 21 days after [Review 01]'s end
@endgantt
```

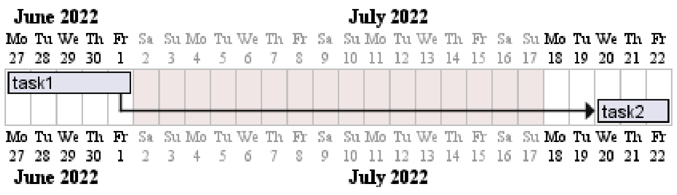


[Ref. QA-13434]

16.16 Working days

It is possible to manage working days.

```
@startgantt

saturday are closed
sunday are closed
2022-07-04 to 2022-07-15 is closed

Project starts 2022-06-27
[task1] starts at 2022-06-27 and lasts 1 week
[task2] starts 2 working days after [task1]'s end and lasts 3 days

@endgantt
```

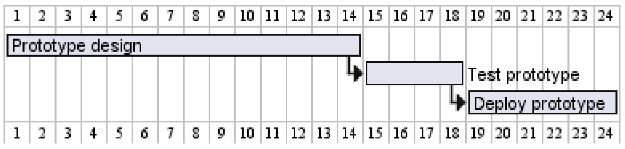


[Ref. QA-16188]

16.17 Simplified task succession

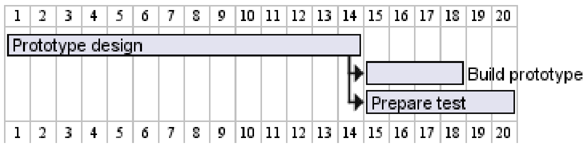
It's possible to use the `then` keyword to denote consecutive tasks.

```
@startgantt
[Prototype design] lasts 14 days
then [Test prototype] lasts 4 days
then [Deploy prototype] lasts 6 days
@endgantt
```



You can also use arrow `->`

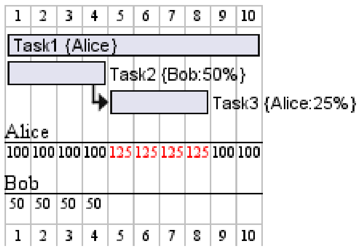
```
@startgantt
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -> [Build prototype]
[Prototype design] -> [Prepare test]
@endgantt
```



16.18 Working with resources

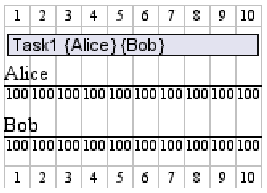
You can affect tasks on resources using the `on` keyword and brackets for resource name.

```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
@endgantt
```



Multiple resources can be assigned to a task:

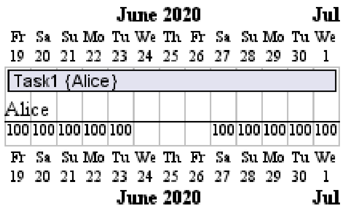
```
@startgantt
[Task1] on {Alice} {Bob} lasts 20 days
@endgantt
```



Resources can be marked as off on specific days:

```
@startgantt
project starts on 2020-06-19
[Task1] on {Alice} lasts 10 days
```

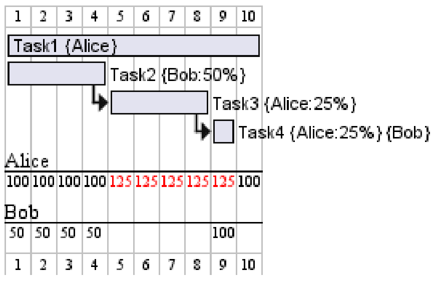
```
{Alice} is off on 2020-06-24 to 2020-06-26
@endgantt
```



16.19 Hide resources

16.19.1 Without any hiding (by default)

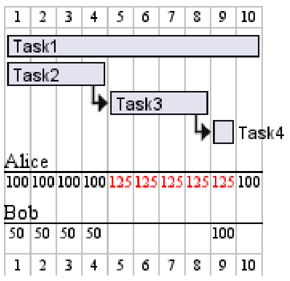
```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```



16.19.2 Hide resources names

You can hide ressources names and percentage, on tasks, using the `hide ressources names` keywords.

```
@startgantt
hide ressources names
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```

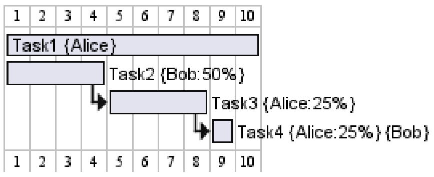


16.19.3 Hide resources footbox

You can also hide ressources names on bottom of the diagram using the `hide ressources footbox` keywords.

```
@startgantt
hide ressources footbox
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
```

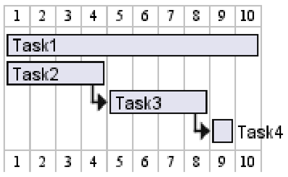
```
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```



16.19.4 Hide the both (resources names and resources footbox)

You can also hide the both.

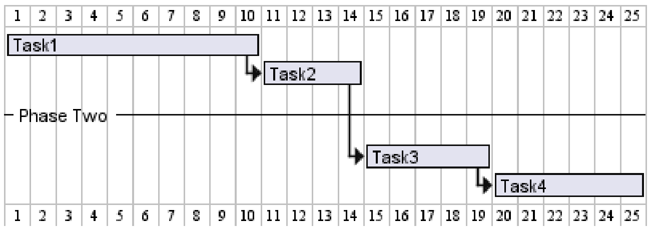
```
@startgantt
hide ressources names
hide ressources footbox
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```



16.20 分隔符

使用-- 分割不同的任务组

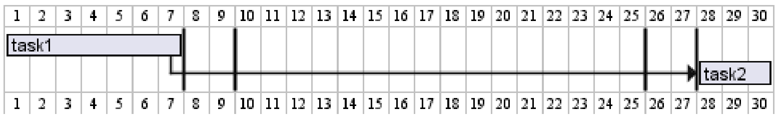
```
@startgantt
[Task1] lasts 10 days
then [Task2] lasts 4 days
-- Phase Two --
then [Task3] lasts 5 days
then [Task4] lasts 6 days
@endgantt
```



16.21 Vertical Separator

You can add Vertical Separators with the syntax: `Separator just [at]`.

```
@startgantt
[task1] lasts 1 week
[task2] starts 20 days after [task1]'s end and lasts 3 days

Separator just at [task1]'s end
Separator just 2 days after [task1]'s end

Separator just at [task2]'s start
Separator just 2 days before [task2]'s start
@endgantt
```



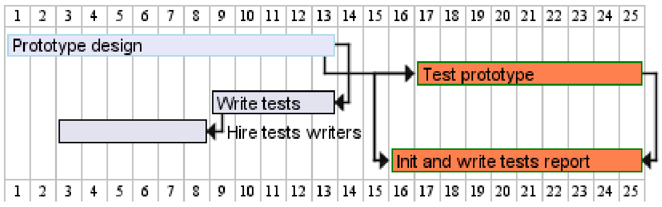
[Ref. QA-16247]

16.22 Complex example

It also possible to use the `and` conjunction.

You can also add delays in constraints.

```
@startgantt
[Prototype design] lasts 13 days and is colored in Lavender/LightBlue
[Test prototype] lasts 9 days and is colored in Coral/Green and starts 3 days after [Prototype desig
[Write tests] lasts 5 days and ends at [Prototype design]'s end
[Hire tests writers] lasts 6 days and ends at [Write tests]'s start
[Init and write tests report] is colored in Coral/Green
[Init and write tests report] starts 1 day before [Test prototype]'s start and ends at [Test prototy
@endgantt
```

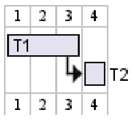


16.23 Comments

As is mentioned on Common Commands page: blockquote Everything that starts with `simple quote` `'` is a comment.

You can also put comments on several lines using `/'` to start and `'/` to end. blockquote *(i.e.: the first character (except space character) of a comment line must be a `simple quote '`)*

```
@startgantt
' This is a comment

[T1] lasts 3 days

/' this comment
is on several lines '/

[T2] starts at [T1]'s end and lasts 1 day
@endgantt
```

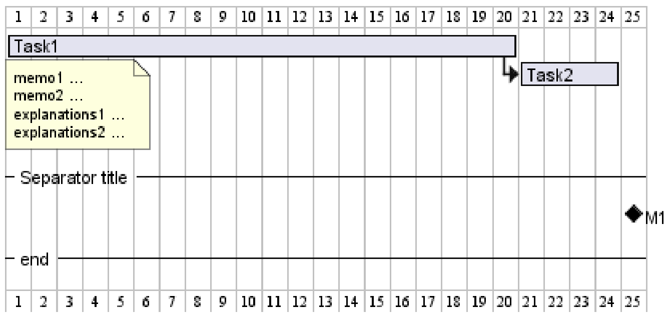


16.24 Using style

16.24.1 Without style (by default)

```
@startgantt
[Task1] lasts 20 days
note bottom
```

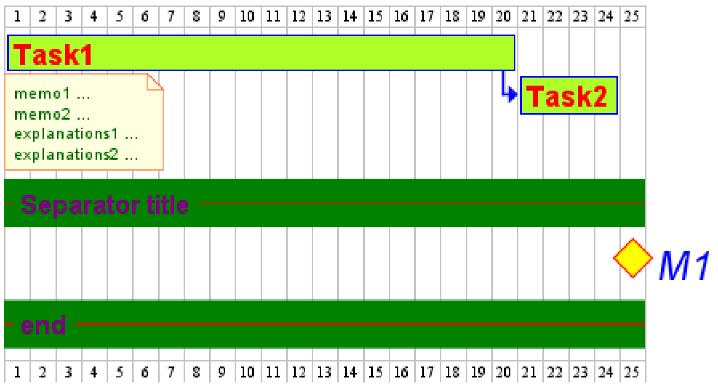
```
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note
[Task2] lasts 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt
```



16.24.2 With style

You can use style to change rendering of elements.

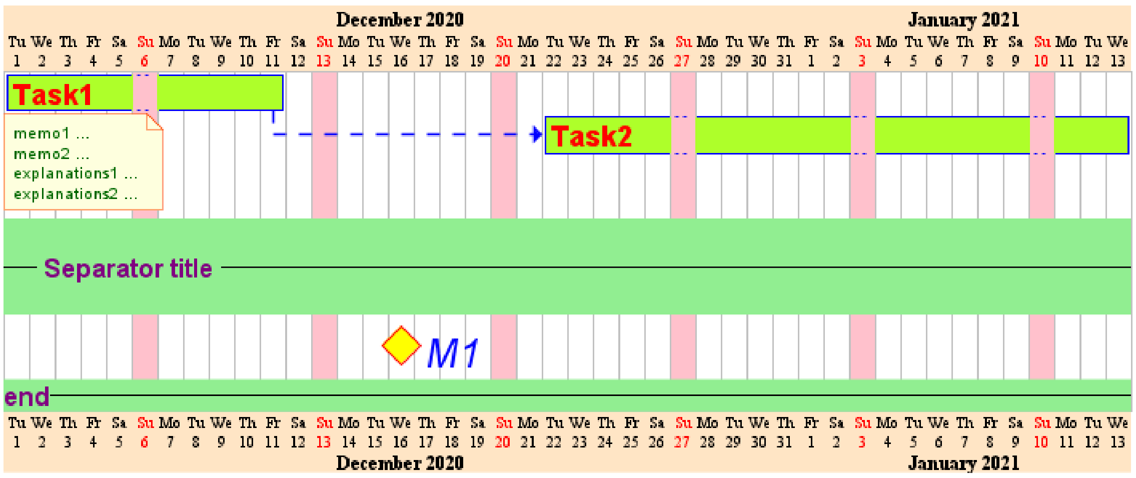
```
@startgantt
<style>
ganttDiagram {
task {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
}
milestone {
FontColor blue
FontSize 25
FontStyle italic
BackGroundColor yellow
LineColor red
}
note {
FontColor DarkGreen
FontSize 10
LineColor OrangeRed
}
arrow {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
}
separator {
LineColor red
BackGroundColor green
FontSize 16
FontStyle bold
FontColor purple
}
}
</style>
[Task1] lasts 20 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note
[Task2] lasts 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt
```



[Ref. QA-10835, QA-12045, QA-11877 and PR-438]

16.24.3 With style (full example)

```
@startgantt
<style>
ganttDiagram {
task {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
}
milestone {
FontColor blue
FontSize 25
FontStyle italic
BackGroundColor yellow
LineColor red
}
note {
FontColor DarkGreen
FontSize 10
LineColor OrangeRed
}
arrow {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
LineStyle 8.0;13.0
LineThickness 3.0
}
separator {
BackgroundColor lightGreen
LineStyle 8.0;3.0
LineColor red
LineThickness 1.0
FontSize 16
FontStyle bold
FontColor purple
Margin 5
Padding 20
}
timeline {
    BackgroundColor Bisque
}
closed {
BackgroundColor pink
FontColor red
}
}
</style>
Project starts the 2020-12-01

[Task1] lasts 10 days
sunday are closed

note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

[Task2] lasts 20 days
[Task2] starts 10 days after [Task1]'s end
-- Separator title --
[M1] happens on 5 days after [Task1]'s end

<style>
separator {
    LineColor black
Margin 0
Padding 0
}
</style>

-- end --
@endgantt
```



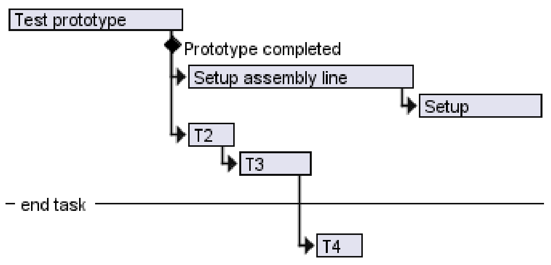
[Ref. QA-13570, QA-13672]

TODO: DONE *Thanks for style for Separator and all style for Arrow (thickness...)*

16.24.4 Clean style

With style, you can also clean a Gantt diagram *(showing tasks, dependencies and relative durations only - but no actual start date and no actual scale)*:

```
@startgantt
<style>
ganttDiagram {
  timeline {
    LineColor transparent
    FontColor transparent
 }
}
</style>

hide footbox
[Test prototype] lasts 7 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] lasts 9 days
[Setup assembly line] starts at [Test prototype]'s end
then [Setup] lasts 5 days
[T2] lasts 2 days and starts at [Test prototype]'s end
then [T3] lasts 3 days
-- end task --
then [T4] lasts 2 days
@endgantt
```

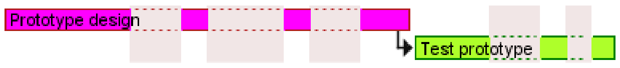


[Ref. QA-13971]

Or:

```
@startgantt
```

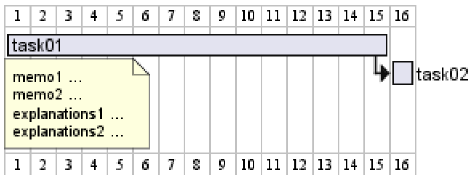
```
<style>
ganttDiagram {
  timeline {
    LineColor transparent
    FontColor transparent
  }
  closed {
    FontColor transparent
  }
}
</style>

hide footbox
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] lasts 9 days
[Test prototype] lasts 5 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



[Ref. QA-13464]

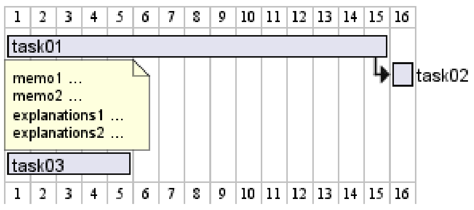
16.25 添加注释

```
@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

[task01] -> [task02]

@endgantt
```

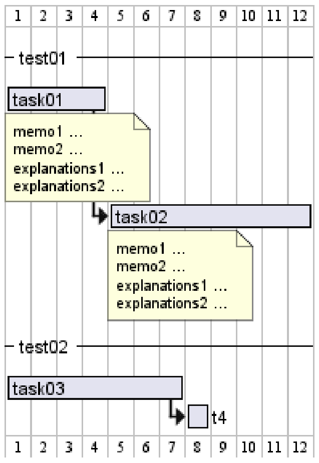


有重叠的例子。

```
@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

[task01] -> [task02]
[task03] lasts 5 days

@endgantt
```

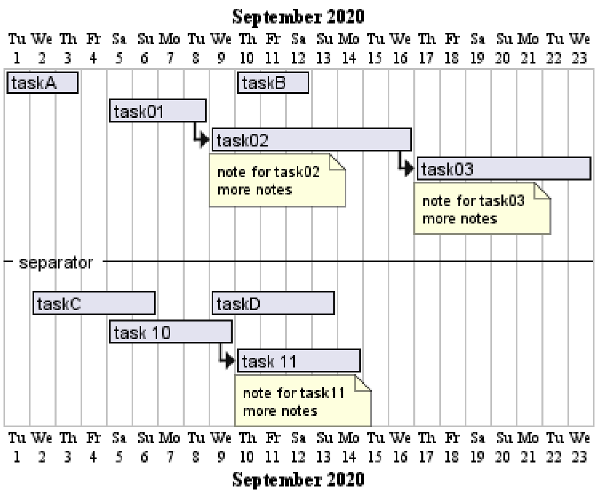


```
@startgantt

-- test01 --

[task01] lasts 4 days
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note

[task02] lasts 8 days
[task01] -> [task02]
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note
-- test02 --

[task03] as [t3] lasts 7 days
[t3] -> [t4]
@endgantt
```



TODO: 完成 谢谢你在重叠时的修正（*#386* 在 *v1.2020.18* 上)。

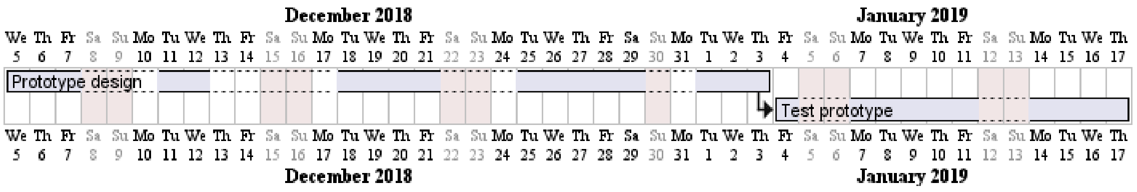
```
@startgantt

Project starts 2020-09-01

[taskA] starts 2020-09-01 and lasts 3 days
[taskB] starts 2020-09-10 and lasts 3 days
[taskB] displays on same row as [taskA]

[task01] starts 2020-09-05 and lasts 4 days

then [task02] lasts 8 days
note bottom
  note for task02
  more notes
end note

then [task03] lasts 7 days
note bottom
  note for task03
  more notes
end note

-- separator --

[taskC] starts 2020-09-02 and lasts 5 days
[taskD] starts 2020-09-09 and lasts 5 days
[taskD] displays on same row as [taskC]

[task 10] starts 2020-09-05 and lasts 5 days
then [task 11] lasts 5 days
note bottom
  note for task11
  more notes
end note
@endgantt
```



16.26 Pause tasks

```
@startgantt
Project starts the 5th of december 2018
saturday are closed
```

```
sunday are closed
2018/12/29 is opened
[Prototype design] lasts 17 days
[Prototype design] pauses on 2018/12/13
[Prototype design] pauses on 2018/12/14
[Prototype design] pauses on monday
[Test prototype] starts at [Prototype design]'s end and lasts 2 weeks
@endgantt
```

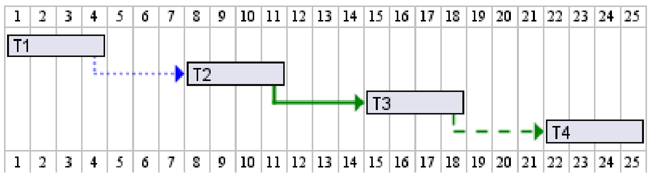


16.27 Change link colors

You can change link colors:

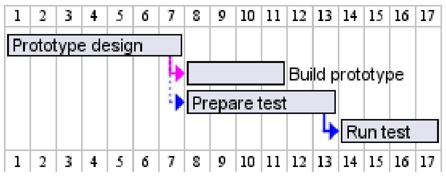
- with this syntax: `with <color> <style> link`

```
@startgantt
[T1] lasts 4 days
[T2] lasts 4 days and starts 3 days after [T1]'s end with blue dotted link
[T3] lasts 4 days and starts 3 days after [T2]'s end with green bold link
[T4] lasts 4 days and starts 3 days after [T3]'s end with green dashed link
@endgantt
```



- or directly by using arrow style

```
@startgantt
<style>
ganttDiagram {
arrow {
LineColor blue
}
}
</style>
[Prototype design] lasts 7 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -[#FF00FF]-> [Build prototype]
[Prototype design] -[dotted]-> [Prepare test]
Then [Run test]  lasts 4 days
@endgantt
```



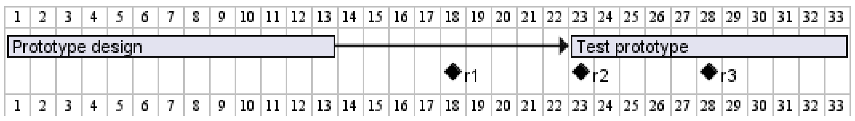
[Ref. QA-13693]

16.28 Tasks or Milestones on the same line

You can put Tasks or Milestones on the same line, with this syntax:

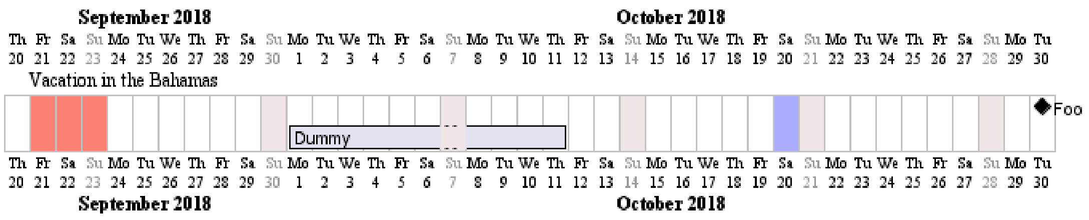
- `[T|M] displays on same row as [T|M]`

```
@startgantt
[Prototype design] lasts 13 days
[Test prototype] lasts 4 days and 1 week
[Test prototype] starts 1 week and 2 days after [Prototype design]'s end
[Test prototype] displays on same row as [Prototype design]
[r1] happens on 5 days after [Prototype design]'s end
[r2] happens on 5 days after [r1]'s end
[r3] happens on 5 days after [r2]'s end
[r2] displays on same row as [r1]
[r3] displays on same row as [r1]
@endgantt
```



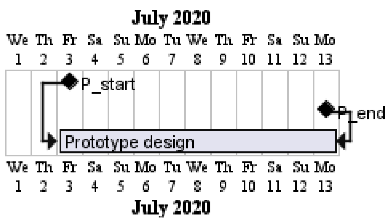
16.29 今天的亮点

```
@startgantt
Project starts the 20th of september 2018
sunday are close
2018/09/21 to 2018/09/23 are colored in salmon
2018/09/21 to 2018/09/30 are named [Vacation in the Bahamas]

today is 30 days after start and is colored in #AAF
[Foo] happens 40 days after start
[Dummy] lasts 10 days and starts 10 days after start

@endgantt
```



16.30 Task between two milestones

```
@startgantt
project starts on 2020-07-01
[P_start] happens 2020-07-03
[P_end]   happens 2020-07-13
[Prototype design] occurs from [P_start] to [P_end]
@endgantt
```

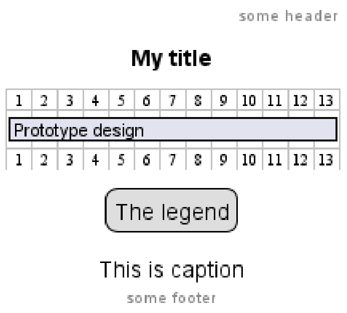


16.31 Grammar and verbal form

Verbal form	Example
[T] starts	
[M] happens	

16.32 Add title, header, footer, caption or legend

```
@startgantt

header some header

footer some footer

title My title

[Prototype design] lasts 13 days

legend
The legend
end legend

caption This is caption

@endgantt
```



(*See also: Common commands*)

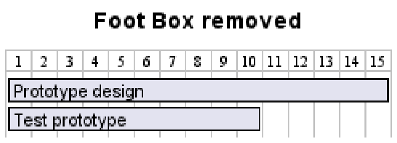
16.33 移除脚盒（所有比例的例子）

你可以使用 `hide footbox` 关键字来移除甘特图的脚盒 （与顺序图一样）。

上的例子。

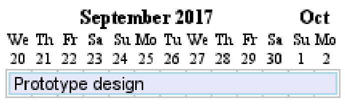
- 每日规模 *(没有项目开始)*

```
@startgantt

hide footbox
title Foot Box removed

[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
@endgantt
```



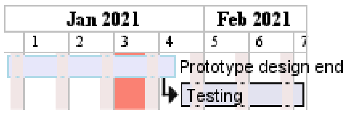
- 日线图

```
@startgantt

Project starts the 20th of september 2017
[Prototype design] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue

hide footbox
@endgantt
```



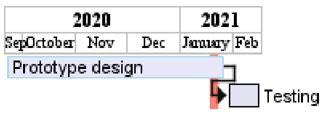
- 周尺度

```
@startgantt
hide footbox

printscale weekly
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



- 月度规模

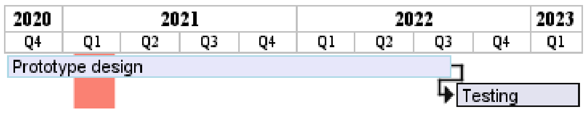
```
@startgantt

hide footbox

projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



- 季度表

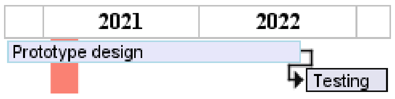
```
@startgantt
```

```
hide footbox

projectscale quarterly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt
```



- 年度规模

```
@startgantt

hide footbox

projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt
```

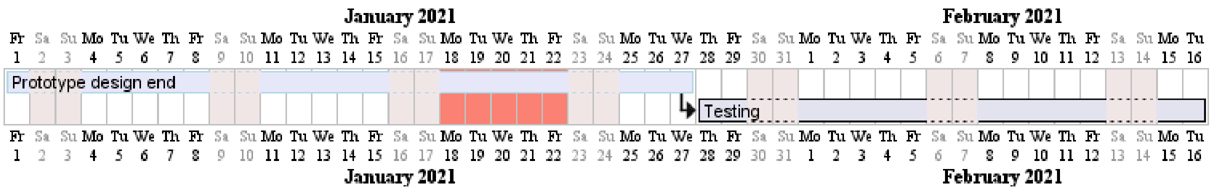


16.34 日历的语言

你可以选择甘特日历的语言，用 `language <xx>` 命令，其中 `<xx>` 是语言的 ISO 639 代码。

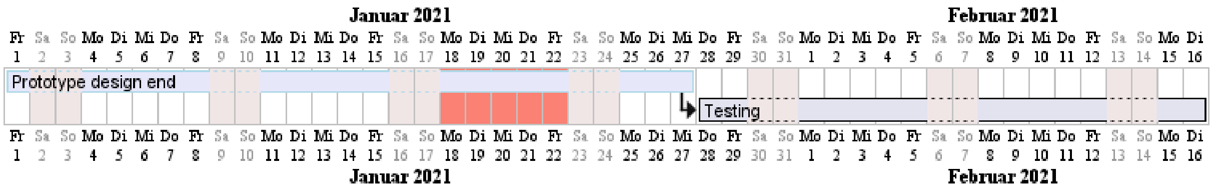
16.34.1 英语（*en*，默认）。

```
@startgantt
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



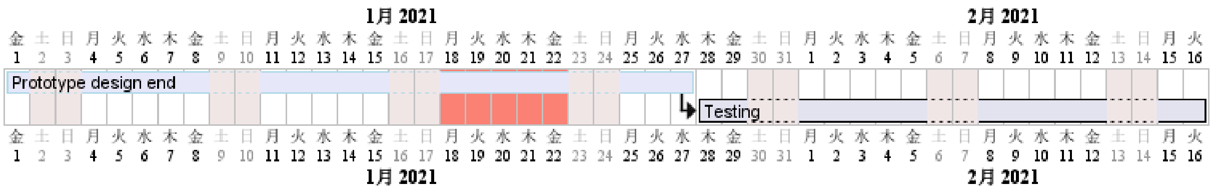
16.34.2 Deutsch (de)

```
@startgantt
language de
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.34.3 日语 (ja)

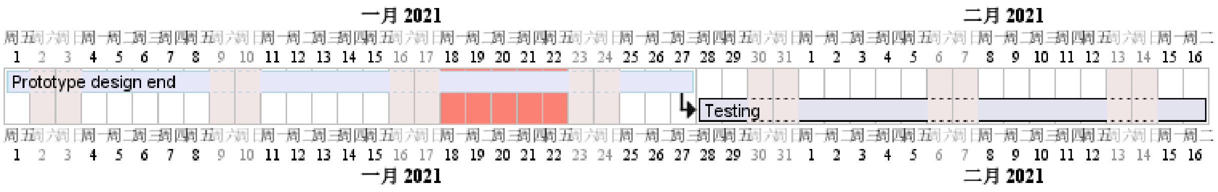
```
@startgantt
language ja
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.34.4 Chinese (zh)

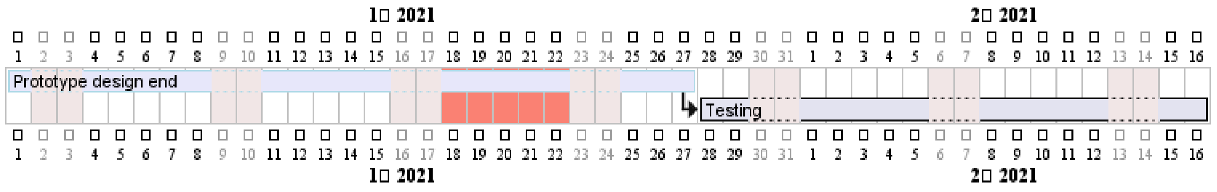
```
@startgantt
language zh
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
```

```
@endgantt
```



16.34.5 Korean (ko)

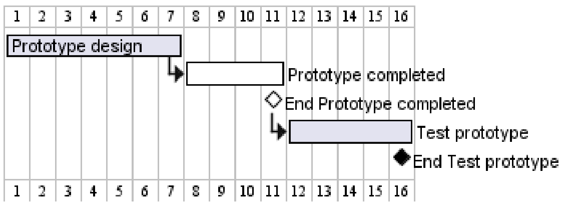
```
@startgantt
language ko
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.35 删除任务或里程碑

可以对任务和里程碑标记为 deleted 来代替通常的完成态，用以区别当前的任务、里程碑可能是被丢弃、推迟或其他情况。

```
@startgantt
[Prototype design] lasts 1 weeks
then [Prototype completed]  lasts 4 days
[End Prototype completed] happens at [Prototype completed]'s end
then [Test prototype] lasts 5 days
[End Test prototype] happens at [Test prototype]'s end

[Prototype completed] is deleted
[End Prototype completed] is deleted
@endgantt
```

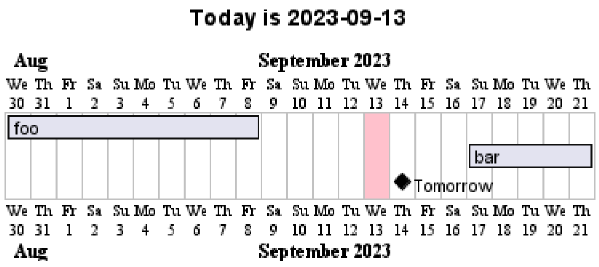


[Ref. QA-9129]

16.36 Start a project, a task or a milestone a number of days before or after today

You can start a project, a task or a milestone a number of days before or after today, using the builtin functions `%now` and `%date`:

```
@startgantt
title Today is %date("YYYY-MM-dd")
!$now = %now()
!$past = %date("YYYY-MM-dd", $now - 14*24*3600)
Project starts $past
today is colored in pink
[foo] lasts 10 days
[bar] lasts 5 days and starts %date("YYYY-MM-dd", $now + 4*24*3600)
[Tomorrow] happens %date("YYYY-MM-dd", $now + 1*24*3600)
@endgantt
```

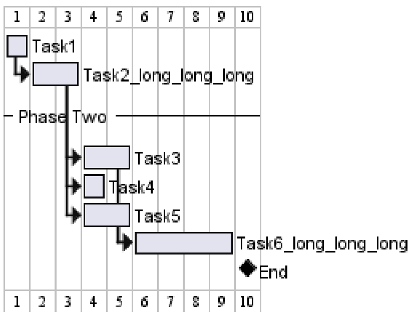


[Ref. QA-16285]

16.37 Change Label position

16.37.1 The labels are near elements *(by default)*

```
@startgantt
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```

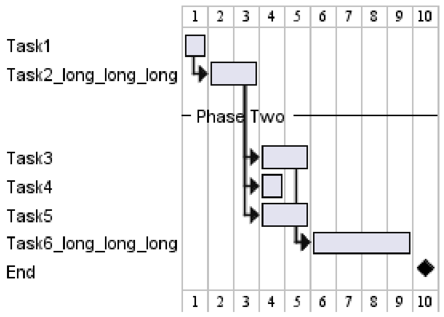


To change the label position, you can use the command `label`:

16.37.2 Label on first column

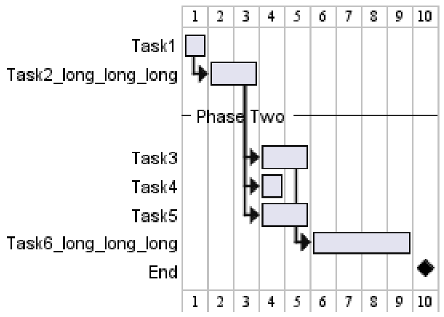
- Left aligned

```
@startgantt
Label on first column and left aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



- Right aligned

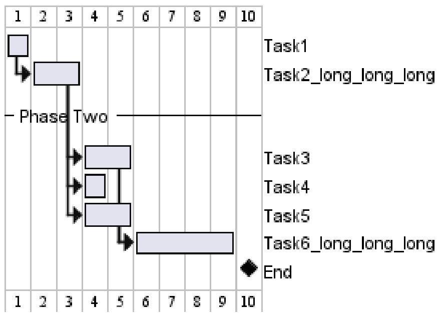
```
@startgantt
Label on first column and right aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



16.37.3 Label on last column

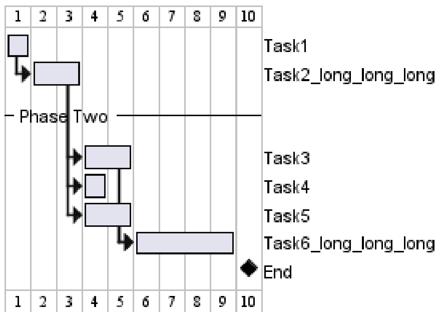
- Left aligned

```
@startgantt
Label on last column and left aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



- Right aligned

```
@startgantt
Label on last column and right aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



[Ref. QA-12433]

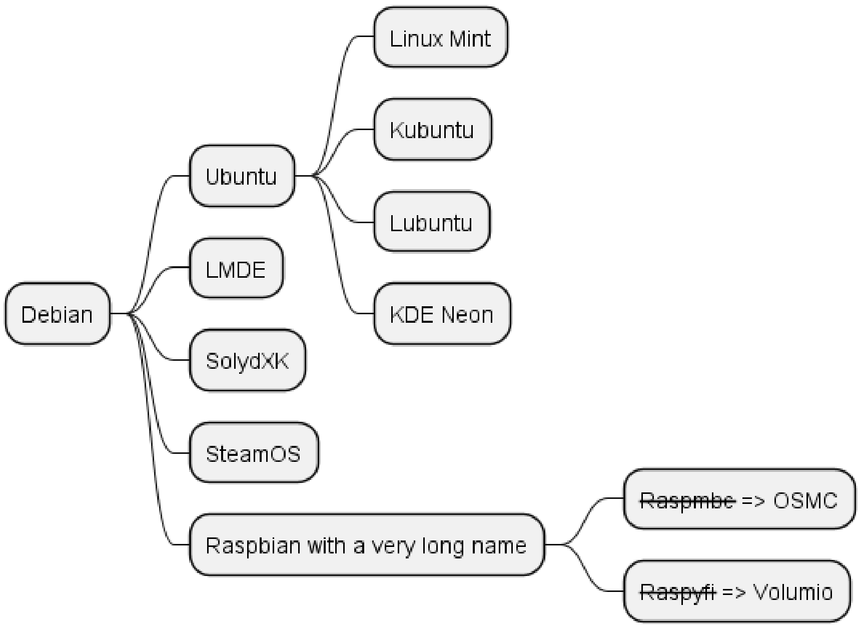
17 思维导图

于测试阶段：语法随时可能更改。

17.1 OrgMode 语法

同时兼容 OrgMode 语法。

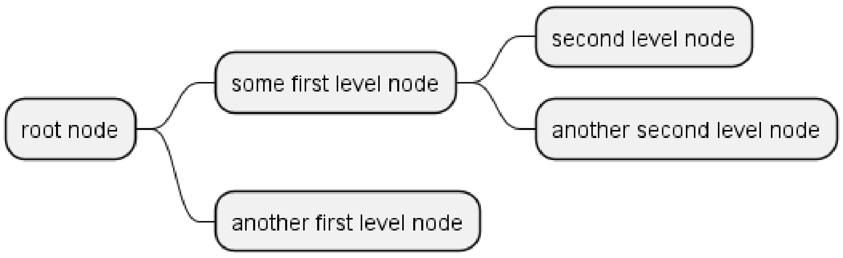
```
@startmindmap
* Debian
** Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** LMDE
** SolydXK
** SteamOS
** Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio
@endmindmap
```



17.2 Markdown 语法

同时兼容 Markdown 语法。

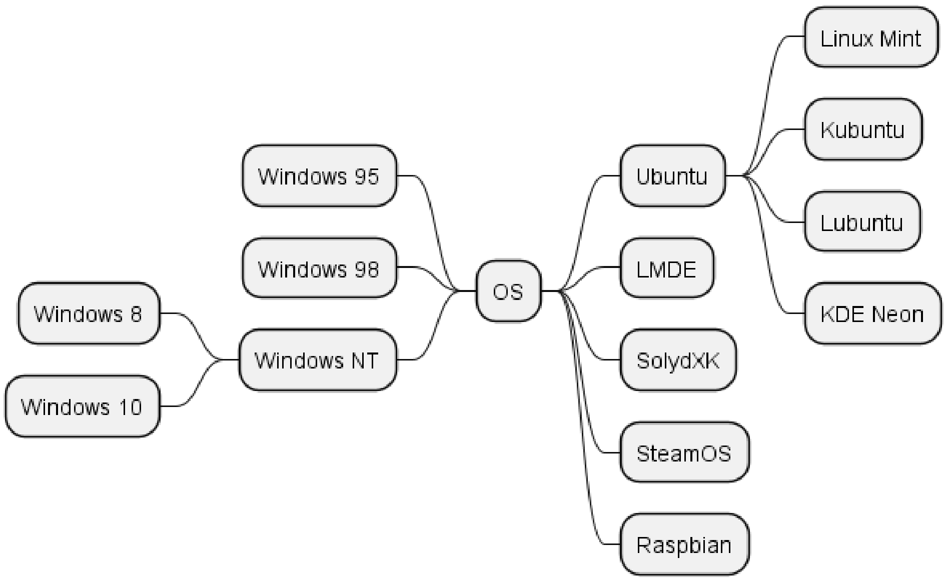
```
@startmindmap
* root node
* some first level node
* second level node
* another second level node
* another first level node
@endmindmap
```



17.3 运算符

你可以使用下面的运算符来决定图形方向。

```
@startmindmap
+ OS
++ Ubuntu
+++ Linux Mint
+++ Kubuntu
+++ Lubuntu
+++ KDE Neon
++ LMDE
++ SolydXK
++ SteamOS
++ Raspbian
-- Windows 95
-- Windows 98
-- Windows NT
--- Windows 8
--- Windows 10
@endmindmap
```

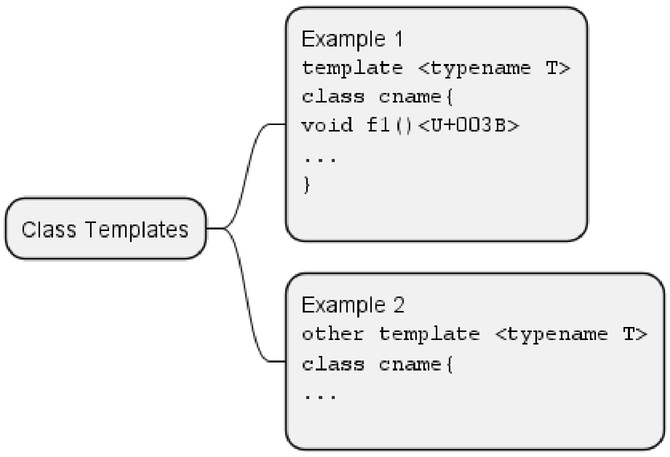


17.4 多行表示

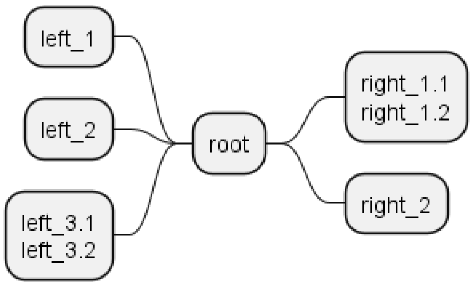
你可以用 : 和 ; 包围文字，来表示多行文本.

```
@startmindmap
* Class Templates
```

```
**:Example 1
<code>
template <typename T>
class cname{
void f1()<U+003B>
...
}
</code>
;
**:Example 2
<code>
other template <typename T>
class cname{
...
</code>
;
@endmindmap
```



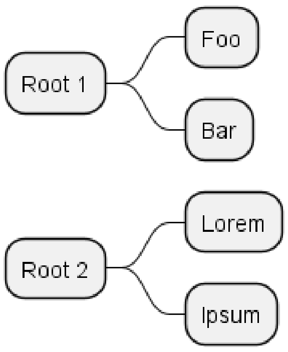
```
@startmindmap
+ root
**:right_1.1
right_1.2;
++ right_2

left side

-- left_1
-- left_2
**:left_3.1
left_3.2;
@endmindmap
```



17.5 Multiroot Mindmap

You can create multiroot mindmap, as:

```
@startmindmap
* Root 1
** Foo
** Bar
* Root 2
** Lorem
** Ipsum
@endmindmap
```



[Ref. QH-773]

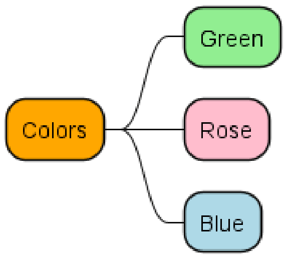
17.6 Colors

It is possible to change node color.

17.6.1 With inline color

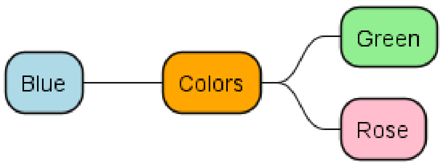
- OrgMode syntax mindmap

```
@startmindmap
*[#Orange] Colors
**[#lightgreen] Green
**[#FFBBCC] Rose
**[#lightblue] Blue
@endmindmap
```



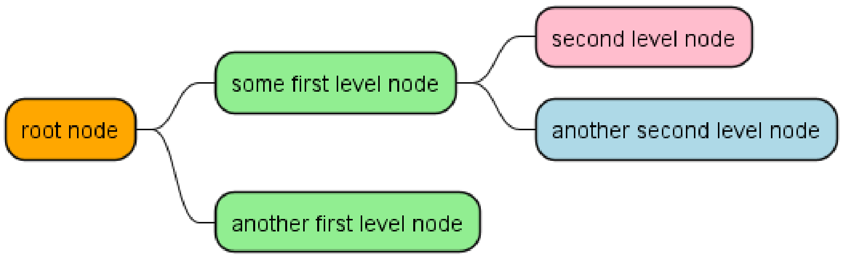
- Arithmetic notation syntax mindmap

```
@startmindmap
+[#Orange] Colors
++[#lightgreen] Green
++[#FFBBCC] Rose
--[#lightblue] Blue
@endmindmap
```



- Markdown syntax mindmap

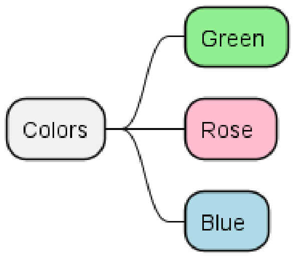
```
@startmindmap
*[#Orange] root node
 *[#lightgreen] some first level node
  *[#FFBBCC] second level node
  *[#lightblue] another second level node
 *[#lightgreen] another first level node
@endmindmap
```



17.6.2 With style color

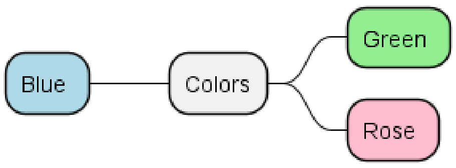
- OrgMode syntax mindmap

```
@startmindmap
<style>
mindmapDiagram {
  .green {
    BackgroundColor lightgreen
  }
  .rose {
    BackgroundColor #FFBBCC
  }
  .your_style_name {
    BackgroundColor lightblue
  }
}
</style>
* Colors
** Green <<green>>
** Rose <<rose>>
** Blue <<your_style_name>>
@endmindmap
```



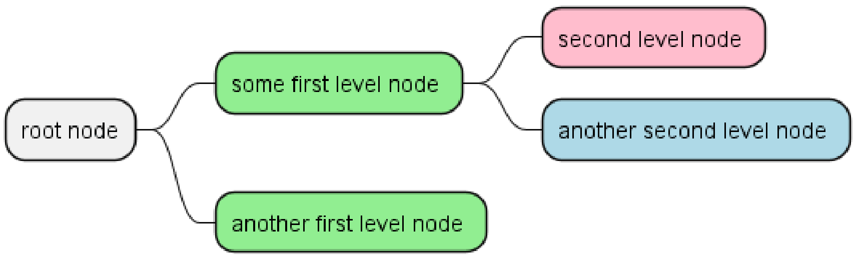
- Arithmetic notation syntax mindmap

```
@startmindmap
<style>
mindmapDiagram {
  .green {
    BackgroundColor lightgreen
  }
  .rose {
    BackgroundColor #FFBBCC
  }
  .your_style_name {
    BackgroundColor lightblue
  }
}
</style>
+ Colors
++ Green <<green>>
++ Rose <<rose>>
-- Blue <<your_style_name>>
@endmindmap
```



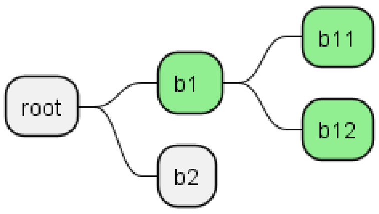
- Markdown syntax mindmap

```
@startmindmap
<style>
mindmapDiagram {
  .green {
    BackgroundColor lightgreen
  }
  .rose {
    BackgroundColor #FFBBCC
  }
  .your_style_name {
    BackgroundColor lightblue
  }
}
</style>
* root node
 * some first level node <<green>>
  * second level node <<rose>>
  * another second level node <<your_style_name>>
 * another first level node <<green>>
@endmindmap
```



- Apply style to a branch

```
@startmindmap
<style>
mindmapDiagram {
  .myStyle * {
    BackgroundColor lightgreen
  }
}
</style>
+ root
++ b1 <<myStyle>>
+++ b11
+++ b12
++ b2
@endmindmap
```

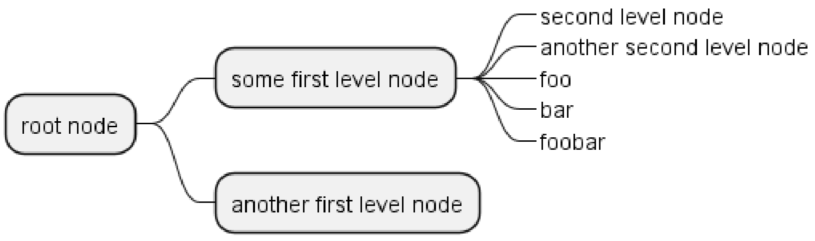


[Ref. GA-920]

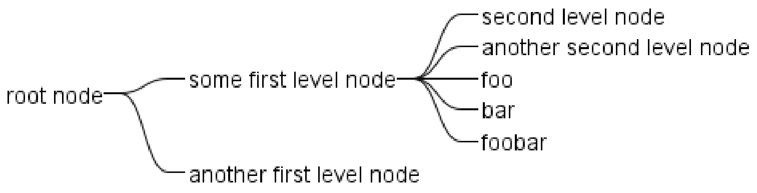
17.7 移除方框

你可以用下划线移除方框图。

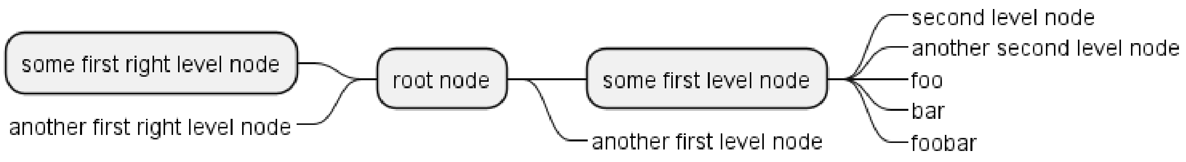
```
@startmindmap
* root node
** some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
** another first level node
@endmindmap
```



```
@startmindmap
*_ root node
**_ some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
**_ another first level node
@endmindmap
```



```
@startmindmap
+ root node
++ some first level node
+++_ second level node
+++_ another second level node
+++_ foo
+++_ bar
+++_ foobar
++_ another first level node
-- some first right level node
--_ another first right level node
@endmindmap
```

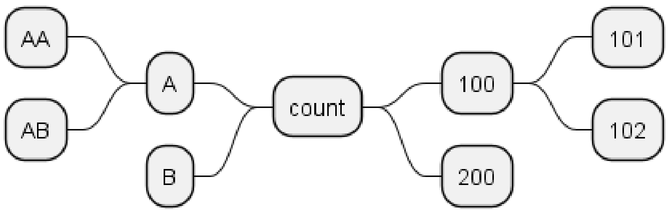


17.8 改变图形方向

你可以同时使用图形的左右两侧。

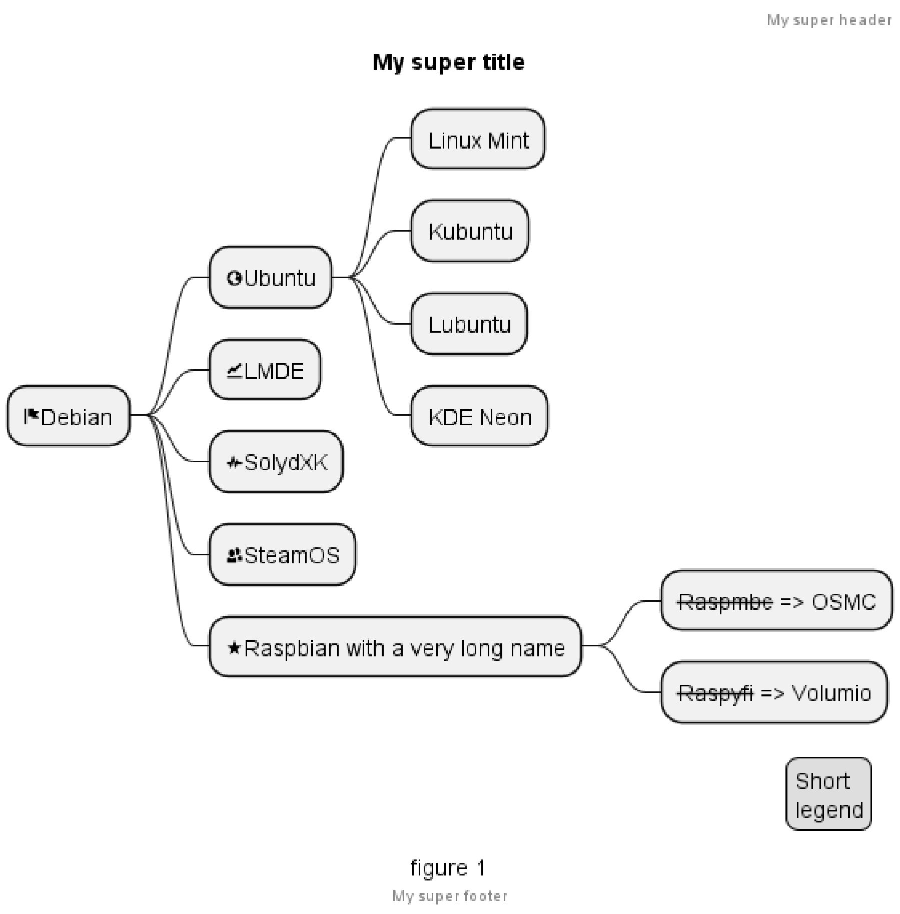
```
@startmindmap
* count
** 100
*** 101
*** 102
** 200
```

```
left side

** A
*** AA
*** AB
** B
@endmindmap
```



17.9 完整示例

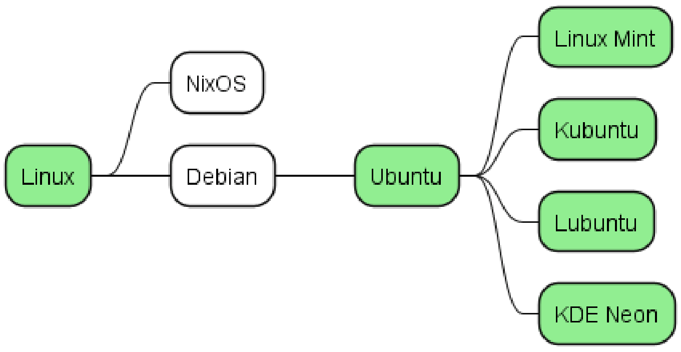
```
@startmindmap
caption figure 1
title My super title

* <&flag>Debian
** <&globe>Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** <&graph>LMDE
** <&pulse>SolydXK
** <&people>SteamOS
** <&star>Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio

header
My super header
endheader

center footer My super footer

legend right
  Short
  legend
endlegend
@endmindmap
```



17.10 改变风格

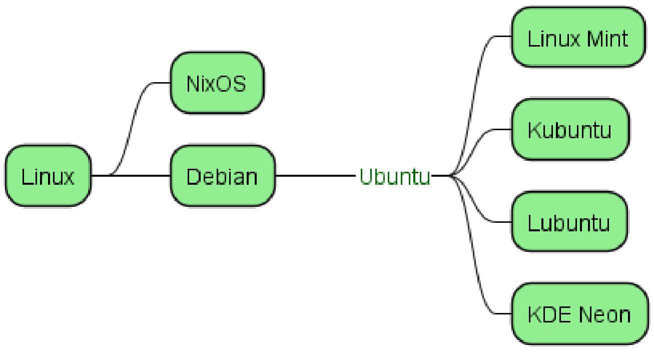
17.10.1 节点、深度

```
@startmindmap
<style>
mindmapDiagram {
    node {
        BackgroundColor lightGreen
    }
    :depth(1) {
      BackGroundColor white
    }
}
</style>
* Linux
** NixOS
** Debian
*** Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap
```



17.10.2 无盒

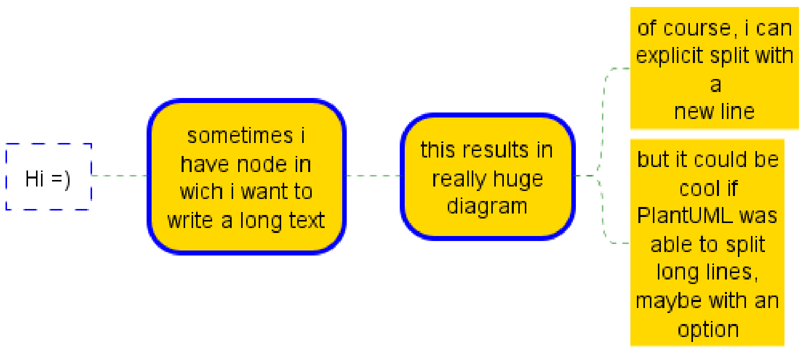
```
@startmindmap
<style>
mindmapDiagram {
  node {
    BackgroundColor lightGreen
  }
  boxless {
    FontColor darkgreen
  }
}
</style>
* Linux
** NixOS
** Debian
***_ Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap
```



17.11 Word Wrap

使用 `MaximumWidth` 设置，你可以控制自动换字。使用的单位是像素。

```
@startmindmap


<style>
node {
    Padding 12
    Margin 3
    HorizontalAlignment center
    LineColor blue
    LineThickness 3.0
    BackgroundColor gold
    RoundCorner 40
    MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in wich i want to write a long text
*** this results in really huge diagram
**** of course, i can explicit split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option

@endmindmap
```


Hi =)
sometimes i have node in wich i want to write a long text
this results in really huge diagram
of course, i can explicit split with a new line
but it could be cool if PlantUML was able to split long lines, maybe with an option

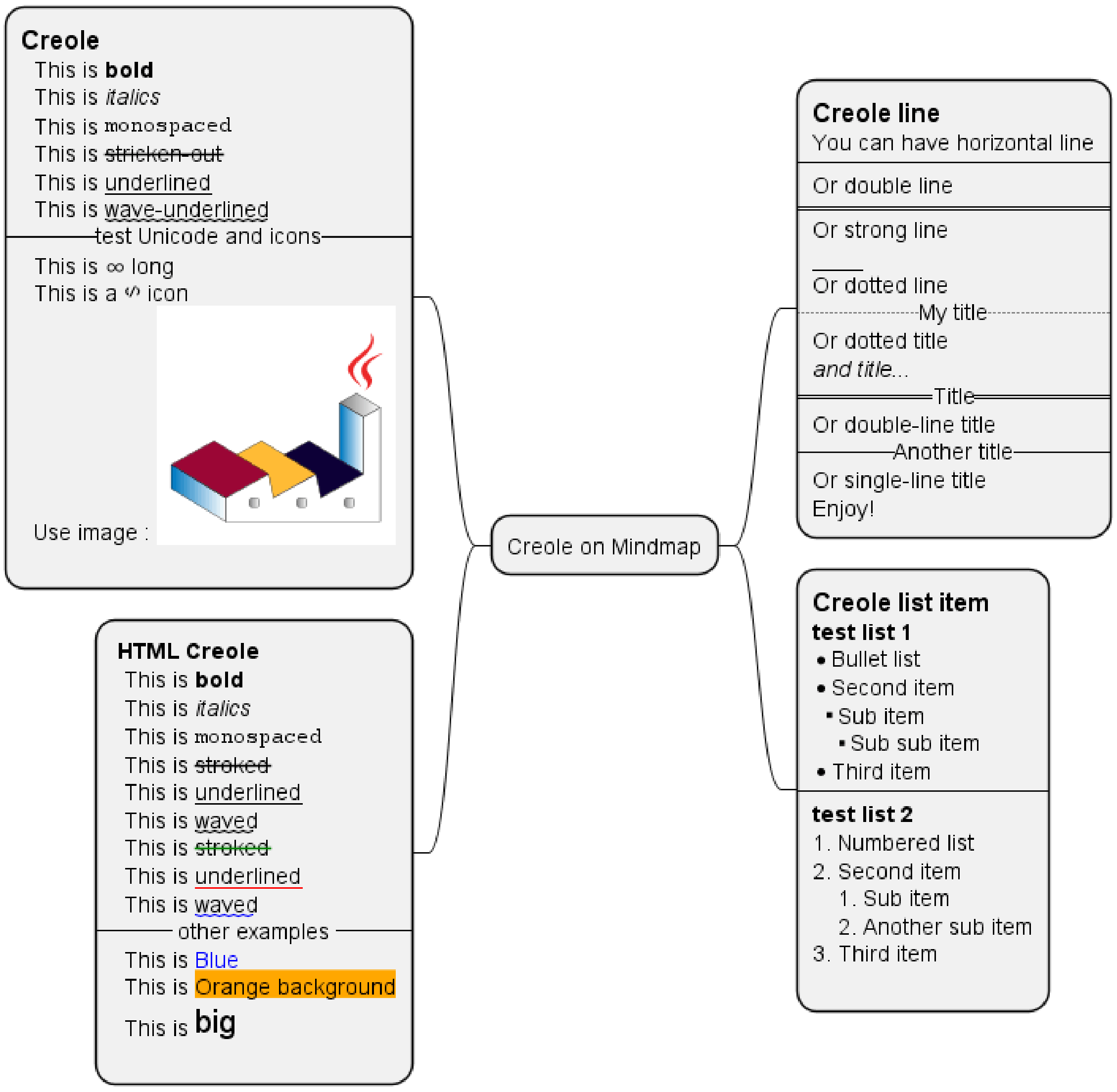
17.12 Creole on Mindmap diagram

You can use Creole or HTML Creole on Mindmap:

```
@startmindmap
* Creole on Mindmap
```

```
left side
**:==Creole
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
--test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
;
**: <b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
-- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
;
right side
**:==Creole line
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
**:==Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
```

```
# Third item
;
@endmindmap
```



[Ref. QA-17838]

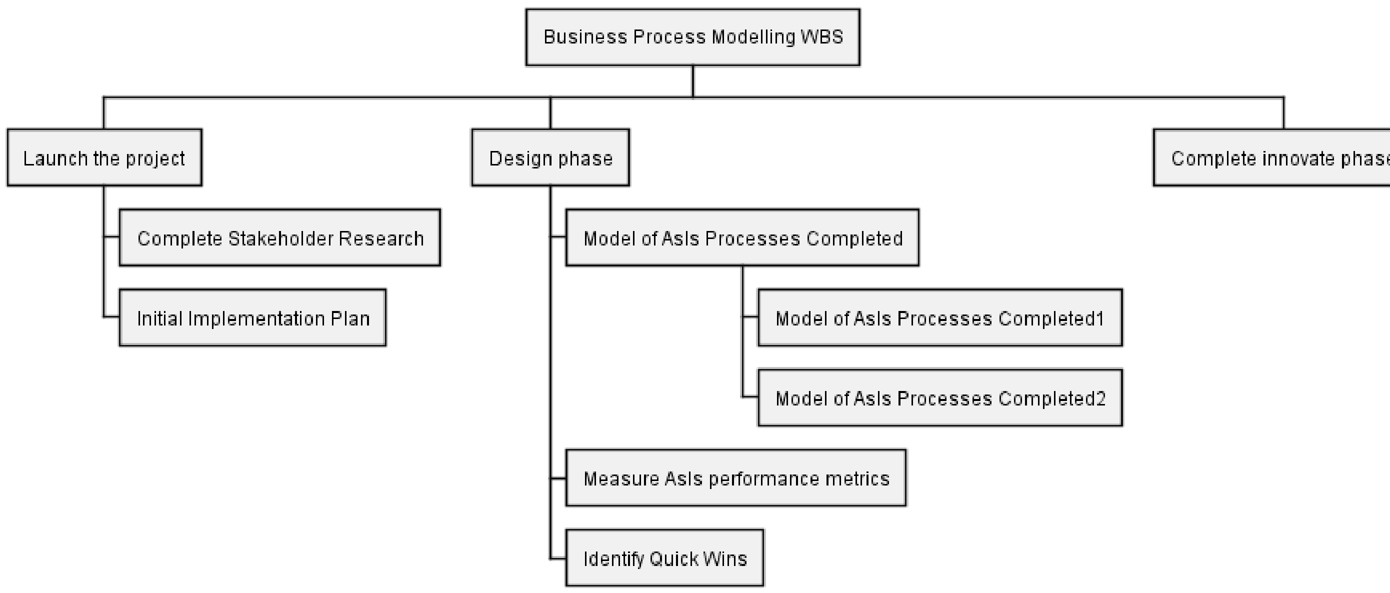
18 工作分解结构（WBS）

WBS 图仍处于测试阶段：语法可能会发生变化，恕不另行通知。

18.1 OrgMode syntax

This syntax is compatible with OrgMode

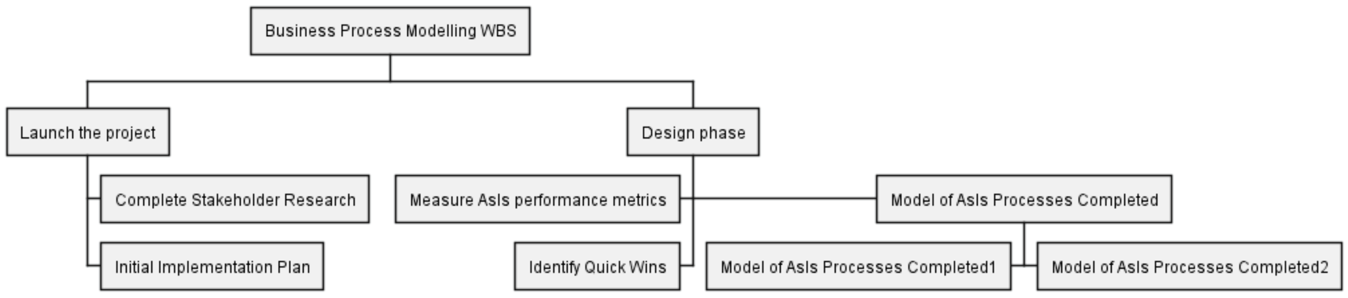
```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
*** Initial Implementation Plan
** Design phase
*** Model of AsIs Processes Completed
**** Model of AsIs Processes Completed1
**** Model of AsIs Processes Completed2
*** Measure AsIs performance metrics
*** Identify Quick Wins
** Complete innovate phase
@endwbs
```


Business Process Modelling WBS
Launch the project
Design phase
Complete innovate phas
Complete Stakeholder Research
Initial Implementation Plan
Model of AsIs Processes Completed
Model of AsIs Processes Completed1
Model of AsIs Processes Completed2
Measure AsIs performance metrics
Identify Quick Wins

18.2 Change direction

You can change direction using < and >

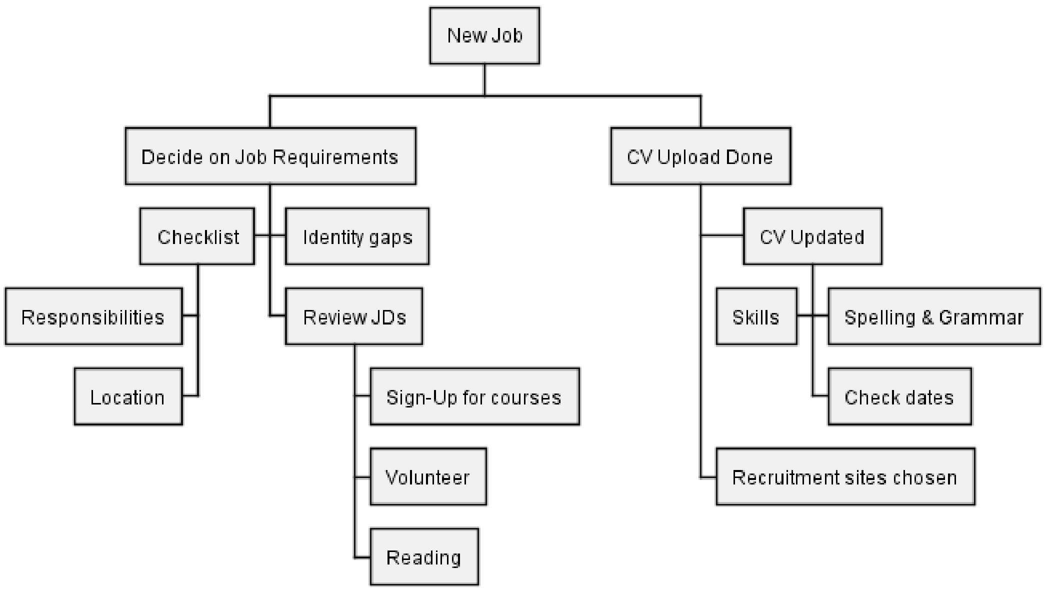
```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
*** Initial Implementation Plan
** Design phase
*** Model of AsIs Processes Completed
****< Model of AsIs Processes Completed1
****> Model of AsIs Processes Completed2
***< Measure AsIs performance metrics
***< Identify Quick Wins
@endwbs
```


Business Process Modelling WBS
Launch the project
Complete Stakeholder Research
Initial Implementation Plan
Design phase
Measure AsIs performance metrics
Identify Quick Wins
Model of AsIs Processes Completed
Model of AsIs Processes Completed1
Model of AsIs Processes Completed2

18.3 Arithmetic notation

You can use the following notation to choose diagram side.

```
@startwbs
+ New Job
++ Decide on Job Requirements
+++ Identity gaps
+++ Review JDs
++++ Sign-Up for courses
++++ Volunteer
++++ Reading
++- Checklist
+++- Responsibilities
+++- Location
++ CV Upload Done
+++ CV Updated
++++ Spelling & Grammar
++++ Check dates
---- Skills
+++ Recruitment sites chosen
@endwbs
```

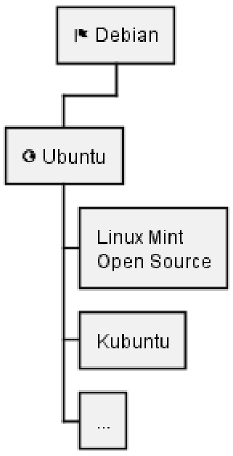


18.4 Multilines

You can use `:` and `;` to have multilines box, as on MindMap.

```
@startwbs
```

```
* <&flag> Debian
** <&globe> Ubuntu

***:Linux Mint
Open Source;

*** Kubuntu
*** ...
@endwbs
```



[Ref. QA-13945]

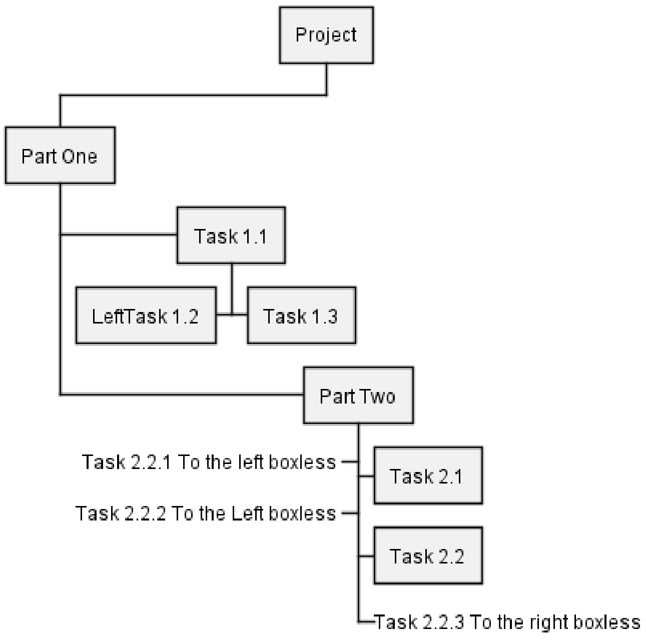
18.5 Removing box

You can use underscore _ to remove box drawing.

18.5.1 Boxless on Arithmetic notation

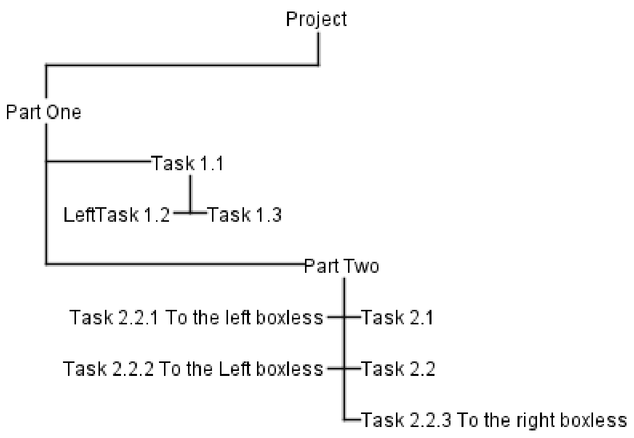
18.5.2 Several boxless node

```
@startwbs
+ Project
 + Part One
  + Task 1.1
   - LeftTask 1.2
   + Task 1.3
  + Part Two
   + Task 2.1
   + Task 2.2
   -_ Task 2.2.1 To the left boxless
   -_ Task 2.2.2 To the Left boxless
   +_ Task 2.2.3 To the right boxless
@endwbs
```


Project
Part One
Task 1.1
LeftTask 1.2
Task 1.3
Part Two
Task 2.2.1 To the left boxless
Task 2.1
Task 2.2.2 To the Left boxless
Task 2.2
Task 2.2.3 To the right boxless

18.5.3 All boxless node

```
@startwbs
+_ Project
 +_ Part One
  +_ Task 1.1
   -_ LeftTask 1.2
   +_ Task 1.3
  +_ Part Two
   +_ Task 2.1
   +_ Task 2.2
   -_ Task 2.2.1 To the left boxless
   -_ Task 2.2.2 To the Left boxless
   +_ Task 2.2.3 To the right boxless
@endwbs
```

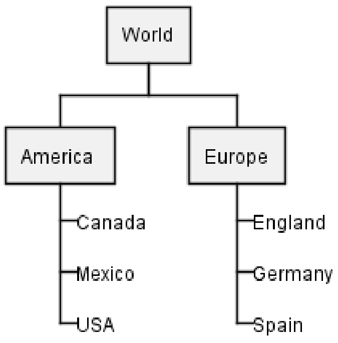

Project
Part One
Task 1.1
LeftTask 1.2
Task 1.3
Part Two
Task 2.2.1 To the left boxless
Task 2.1
Task 2.2.2 To the Left boxless
Task 2.2
Task 2.2.3 To the right boxless

18.5.4 Boxless on OrgMode syntax

18.5.5 Several boxless node

```
@startwbs
* World
** America
```

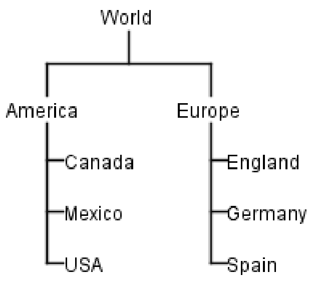
```
***_ Canada
***_ Mexico
***_ USA
** Europe
***_  England
***_  Germany
***_  Spain
@endwbs
```



[Ref. QA-13297]

18.5.6 All boxless node

```
@startwbs
*_ World
**_ America
***_ Canada
***_ Mexico
***_ USA
**_ Europe
***_  England
***_  Germany
***_  Spain
@endwbs
```



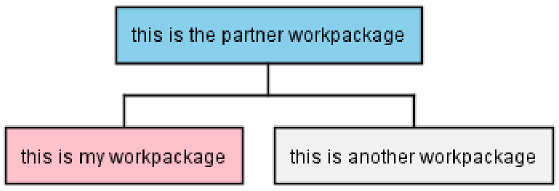
[Ref. QA-13355]

18.6 Colors (with inline or style color)

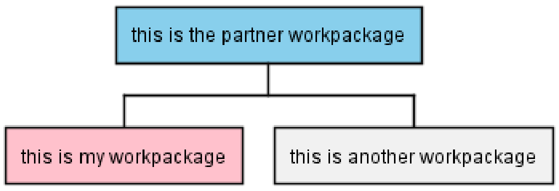
It is possible to change node color:

- with inline color

```
@startwbs
*[#SkyBlue] this is the partner workpackage
**[#pink] this is my workpackage
** this is another workpackage
@endwbs
```



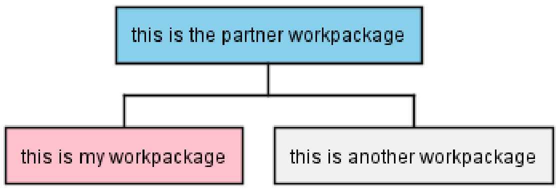
```
@startwbs
+[#SkyBlue] this is the partner workpackage
++[#pink] this is my workpackage
++ this is another workpackage
@endwbs
```



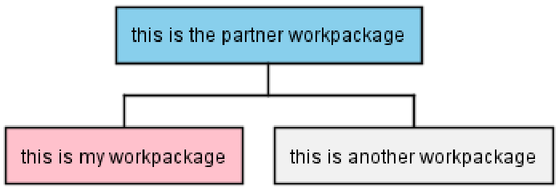
[Ref. QA-12374, only from v1.2020.20]

- with style color

```
@startwbs
<style>
wbsDiagram {
  .pink {
      BackgroundColor pink
  }
  .your_style_name {
      BackgroundColor SkyBlue
  }
}
</style>
* this is the partner workpackage <<your_style_name>>
** this is my workpackage <<pink>>
** this is another workpackage
@endwbs
```



```
@startwbs
<style>
wbsDiagram {
  .pink {
      BackgroundColor pink
  }
  .your_style_name {
      BackgroundColor SkyBlue
  }
}
</style>
+ this is the partner workpackage <<your_style_name>>
++ this is my workpackage <<pink>>
++ this is another workpackage
@endwbs
```

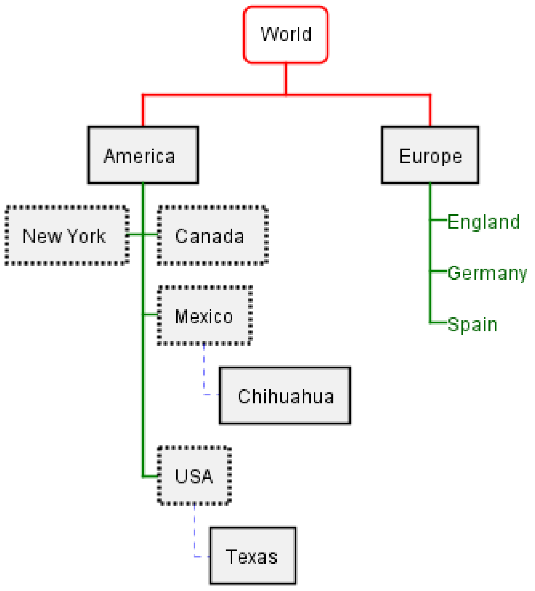


18.7 Using style

It is possible to change diagram style.

```
@startwbs
<style>
wbsDiagram {
  // all lines (meaning connector and borders, there are no other lines in WBS) are black by default
  Linecolor black
  arrow {
    // note that connector are actually "arrow" even if they don't look like as arrow
    // This is to be consistent with other UML diagrams. Not 100% sure that it's a good idea
    // So now connector are green
    LineColor green
  }
  :depth(0) {
      // will target root node
      BackgroundColor White
      RoundCorner 10
      LineColor red
      // Because we are targetting depth(0) for everything, border and connector for level 0 will be
  }
  arrow {
    :depth(2) {
      // Targetting only connector between Mexico-Chihuahua and USA-Texas
      LineColor blue
      LineStyle 4
      LineThickness .5
    }
  }
  node {
    :depth(2) {
      LineStyle 2
      LineThickness 2.5
    }
  }
  boxless {
    // will target boxless node with '_'
    FontColor darkgreen
  }
}
</style>
* World
** America
*** Canada
*** Mexico
**** Chihuahua
*** USA
**** Texas
```

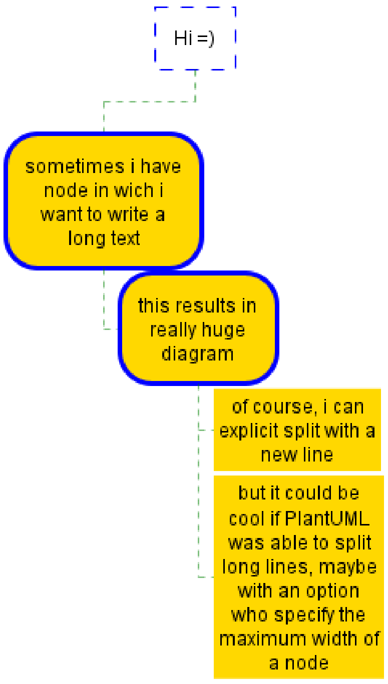
```
***< New York
** Europe
***_  England
***_  Germany
***_  Spain
@endwbs
```



18.8 Word Wrap

Using `MaximumWidth` setting you can control automatic word wrap. Unit used is pixel.

```
@startwbs


<style>
node {
    Padding 12
    Margin 3
    HorizontalAlignment center
    LineColor blue
    LineThickness 3.0
    BackgroundColor gold
    RoundCorner 40
    MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in wich i want to write a long text
*** this results in really huge diagram
**** of course, i can explicit split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option who specify

@endwbs
```

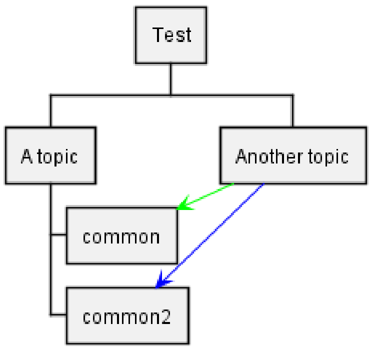

Hi =)
sometimes i have node in wich i want to write a long text
this results in really huge diagram
of course, i can explicit split with a new line
but it could be cool if PlantUML was able to split long lines, maybe with an option who specify the maximum width of a node

18.9 Add arrows between WBS elements

You can add arrows between WBS elements.

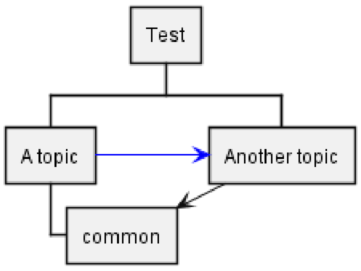
Using alias with as:

```
@startwbs
<style>
.foo {
  LineColor #00FF00;
}
</style>
* Test
** A topic
*** "common" as c1
*** "common2" as c2
** "Another topic" as t2
t2 -> c1 <<foo>>
t2 ..> c2 #blue
@endwbs
```



Using alias in parentheses:

```
@startwbs
* Test
**(b) A topic
***(c1) common
**(t2) Another topic
t2 --> c1
b -> t2 #blue
@endwbs
```

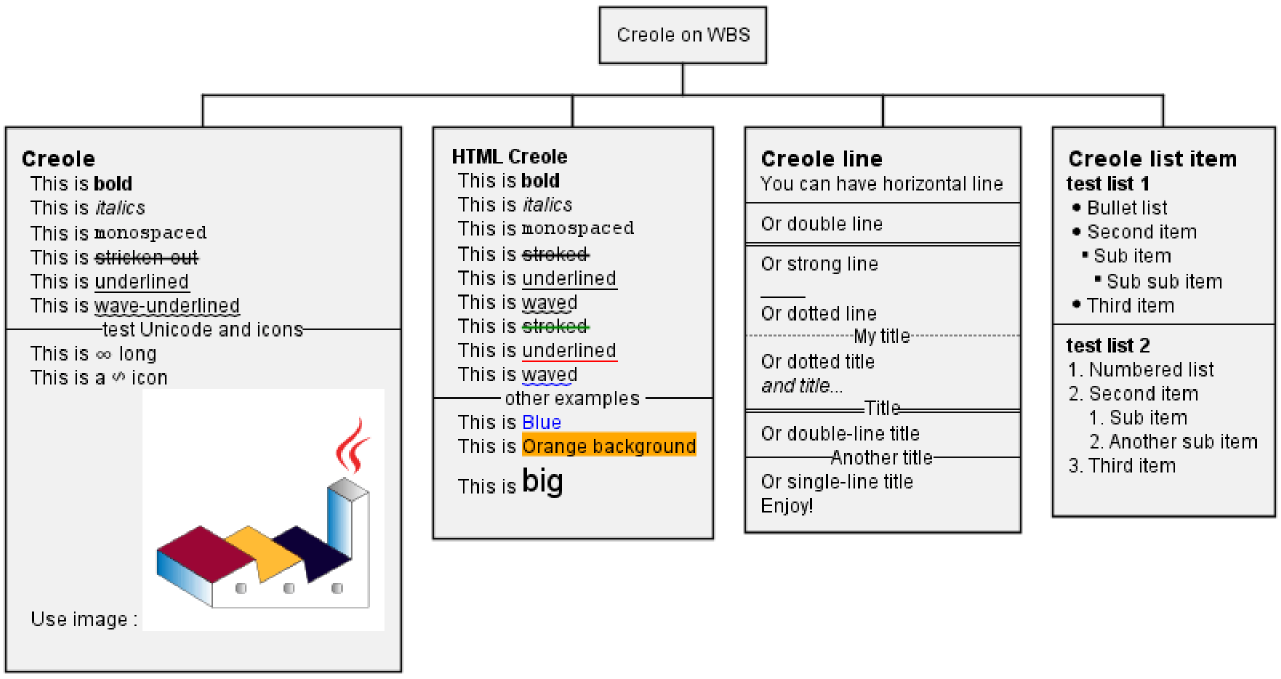


[Ref. QA-16251]

18.10 Creole on WBS diagram

You can use Creole or HTML Creole on WBS:

```
@startwbs
* Creole on WBS
**:==Creole
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
--test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
;
**: <b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
-- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
;
**:==Creole line
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
**:==Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
;
@endwbs
```

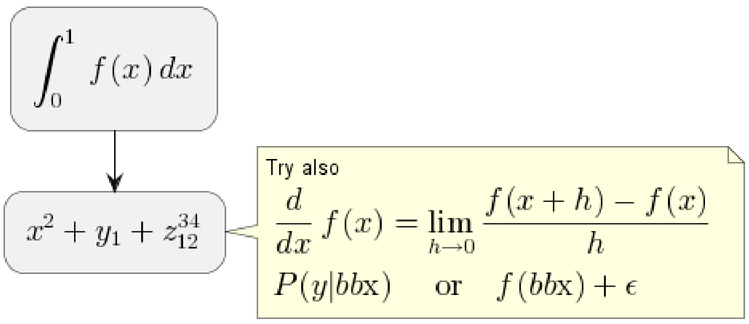


Creole on WBS
Creole
This is **bold**
This is *italics*
This is `monospaced`
This is ~~stricken-out~~
This is underlined
This is wave-underlined
test Unicode and icons
This is ∞ long
This is a icon
Use image :
HTML Creole
This is **bold**
This is *italics*
This is `monospaced`
This is ~~stroked~~
This is underlined
This is waved
This is ~~stroked~~
This is underlined
This is waved
other examples
This is Blue
This is Orange background
This is big
Creole line
You can have horizontal line
Or double line
Or strong line
Or dotted line
My title
Or dotted title
and title...
Title
Or double-line title
Another title
Or single-line title
Enjoy!
Creole list item
test list 1
• Bullet list
• Second item
▪ Sub item
▪ Sub sub item
• Third item
test list 2
1. Numbered list
2. Second item
1. Sub item
2. Another sub item
3. Third item

19 简介

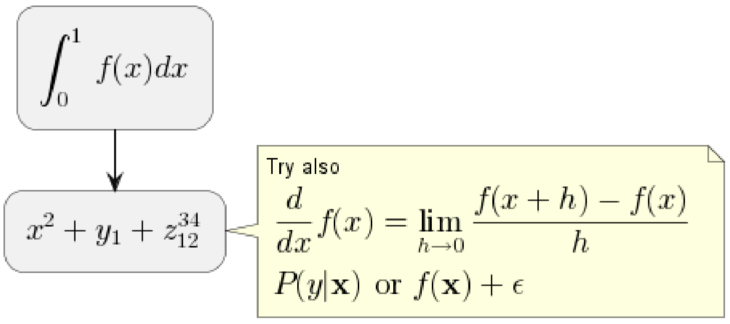
使用 PlantUML 时，若需要输入数学公式，可以使用 AsciiMath：

```
@startuml
:<math>int_0^1f(x)dx</math>;
:<math>x^2+y_1+z_12^34</math>;
note right
Try also
<math>d/dxf(x)=lim_(h->0)(f(x+h)-f(x))/h</math>
<math>P(y|bb"x") or f(bb"x")+epsilon</math>
end note
@enduml
```



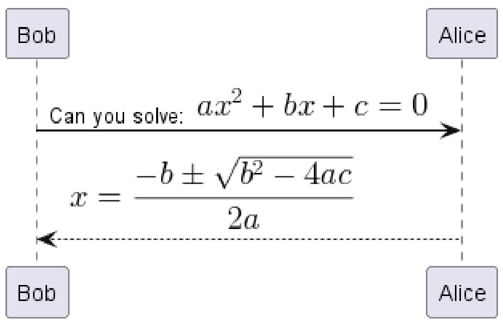
或 JLaTeXMath：

```
@startuml
:<latex>\int_0^1f(x)dx</latex>;
:<latex>x^2+y_1+z_{12}^{34}</latex>;
note right
Try also
<latex>\dfrac{d}{dx}f(x)=\lim\limits_{h \to 0}\dfrac{f(x+h)-f(x)}{h}</latex>
<latex>P(y|\mathbf{x}) \mbox{ or } f(\mathbf{x})+\epsilon</latex>
end note
@enduml
```



其它样例：

```
@startuml
Bob -> Alice : Can you solve: <math>ax^2+bx+c=0</math>
Alice --> Bob: <math>x = (-b+-sqrt(b^2-4ac))/(2a)</math>
@enduml
```


Bob
Alice
Can you solve: $ax^2 + bx + c = 0$
$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
Bob
Alice

19.1 独立图

您也可以用 `@startmath/@endmath` 来创建独立的 AsciiMath 公式。

```
@startmath
f(t)=(a_0)/2 + sum_(n=1)^ooa_ncos((npit)/L)+sum_(n=1)^oo b_n\ sin((npit)/L)
@endmath
```

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} b_n\ \sin\left(\frac{n\pi t}{L}\right)$$

或用 `@startlatex/@endlatex` 来创建独立的 JLaTeXMath 公式。

```
@startlatex
\sum_{i=0}^{n-1} (a_i + b_i^2)
@endlatex
```

$$\sum_{i=0}^{n-1} (a_i + b_i^2)$$

19.2 这是如何工作的?

要绘制这些公式, PlantUML 使用了两个开源项目:

- AsciiMath 转换 AsciiMath 符号为 LaTeX 表达式。
- JLatexMath 来显示 LaTex 数学公式。JLaTeXMath 是最好的显示 LaTeX 代码的 Java 类库。

ASCIIMathTeXImg.js 是一个小到足以集成到 PlantUML 标准发版的。

由于 JLatexMath 太大, 您要单独到下载它, 然后解压 4 jar 文件 (*batik-all-1.7.jar*, *jlatexmath-minimal-1.0.3.jar*, *jlm_cyrillic.jar* 和 *jlm_greek.jar*) 到 PlantUML.jar 同一目录下。

20 实体关系图

基于信息工程符号。

这是对现有类图的一个扩展。这个扩展增加了：

- 信息工程符号的额外关系。
- 一个映射到类图的 entity 别名 class 。
- 一个额外的可见性修改器 * ，用于识别强制属性。

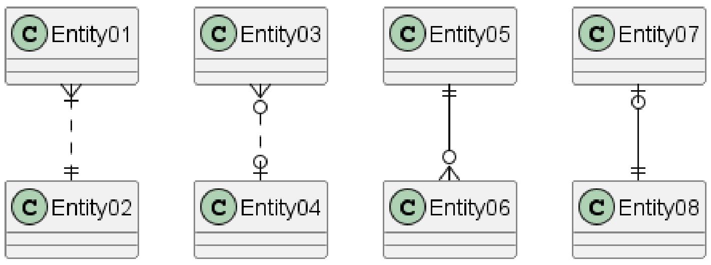
除此之外，绘制图表的语法与类图相同。类图的所有其他特征也被支持。

20.1 Information Engineering Relations

Type	Symbol
Zero or One	|o--
Exactly One	||--
Zero or Many	}o--
One or Many	}|--

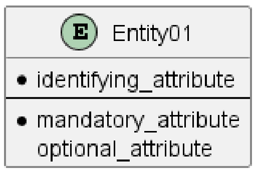
Examples:

```
@startuml
Entity01 }|..|| Entity02
Entity03 }o..o| Entity04
Entity05 ||--o{ Entity06
Entity07 |o--|| Entity08
@enduml
```



20.2 实体

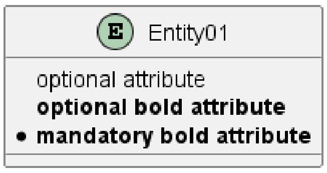
```
@startuml
entity Entity01 {
  * identifying_attribute
  --
  * mandatory_attribute
  optional_attribute
}
@enduml
```



同样，这也是正常的类图语法（除了使用 entity ，而不是 class ）。你可以在类图中做的任何事情都可以在这里完成。

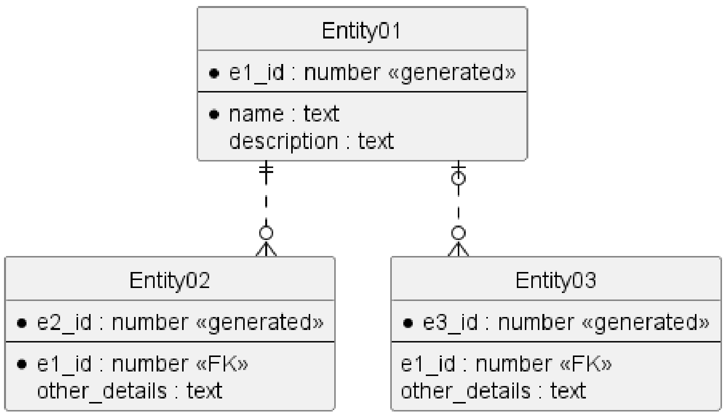
* 可见性修饰符可以用来识别强制属性。在修饰符之后可以使用空格，以避免与克里奥尔语的黑体字冲突。

```
@startuml
entity Entity01 {
   optional attribute
   **optional bold attribute**
   * **mandatory bold attribute**
}
@enduml
```



20.3 完整的例子

```
@startuml

' hide the spot
hide circle

' avoid problems with angled crows feet
skinparam linetype ortho

entity "Entity01" as e01 {
  *e1_id : number <<generated>>
  --
  *name : text
  description : text
}

entity "Entity02" as e02 {
  *e2_id : number <<generated>>
  --
  *e1_id : number <<FK>>
  other_details : text
}

entity "Entity03" as e03 {
  *e3_id : number <<generated>>
  --
  e1_id : number <<FK>>
  other_details : text
}

e01 ||..o{ e02
e01 |o..o{ e03

@enduml
```



目前，当鱼尾纹与实体成一定角度绘制时，鱼尾纹看起来不是很好。这可以通过使用 `linetype ortho` skinparam 来避免。

21 通用命令

Explore these commands to create diagrams that are both functional and aesthetically pleasing, tailoring each element to your exact specifications.

21.1 注释

21.1.1 简单的注释

所有以 `simple quote '` 开始的都是一个的注释。

```
@startuml
'Line comments use a single apostrophe
@enduml
```

21.1.2 </zh3> * 块状注释 <zh3>

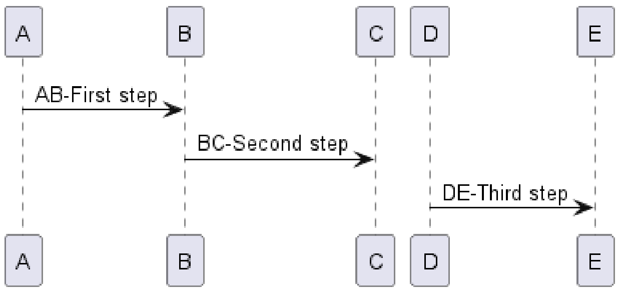
块状注释使用 C 风格的注释，除了用一撇一捺代替', 然后你也可以把注释放在几行上，用/' 开始，'/ 结束。

```
@startuml
/'
many lines comments
here
'/
@enduml
```

[Ref. QA-1353]

然后你也可以在同一行中放置块状注释，如。

```
@startuml
/' case 1 '/   A -> B : AB-First step
               B -> C : BC-Second step
/' case 2 '/   D -> E : DE-Third step
@enduml
```



[参考资料：QA-3906 和 QA-3910] 。

[Ref. GH-214]

21.2 缩放

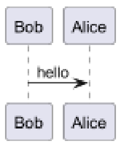
You can use the `scale` command to zoom the generated image.

You can use either a number or a fraction to define the scale factor. You can also specify either width or height (in pixel). And you can also give both width and height : the image is scaled to fit inside the specified dimension.

- `scale 1.5`
- `scale 2/3`
- `scale 200 width`
- `scale 200 height`

- `scale 200*100`
- `scale max 300*200`
- `scale max 1024 width`
- `scale max 800 height`

```
@startuml
scale 180*90
Bob->Alice : hello
@enduml
```

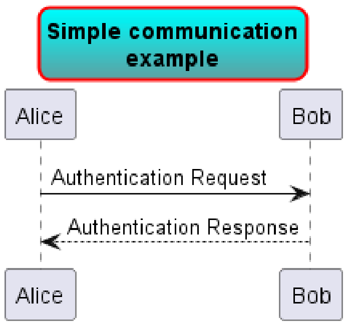


21.3 标题

使用 `title` 关键字添加标题。你可以在标题描述中使用 添加新行。

Some skinparam settings are available to put borders on the title.

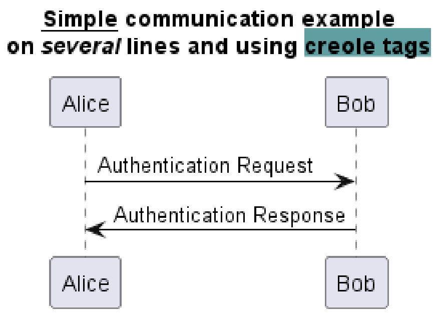
```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue

title Simple communication\nexample

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

@enduml
```



You can use creole formatting in the title.

You can also define title on several lines using `title` and `end title` keywords.

```
@startuml

title
 <u>Simple</u> communication example
 on <i>several</i> lines and using <back:cadetblue>creole tags</back>
end title

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```


Simple communication example
on several lines and using creole tags
Alice
Bob
Authentication Request
Authentication Response

21.4 图片标题

使用 caption 关键字在图像下放置一个标题.

```
@startuml

caption figure 1
Alice -> Bob: Hello

@enduml
```

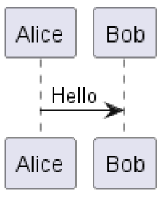


figure 1

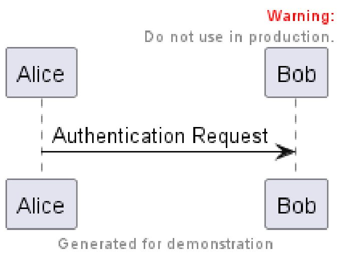
21.5 页眉和页脚

你可以使用 header 和 footer 命令在生成的图中增加页眉和页脚。

你可以选择指定 center, left 或 right 关键字使页眉或页脚实现居中、左对齐和右对齐。

和标题一样，页眉或页脚内容可以在多行中定义，而且同样可以在页眉或页脚中输入一些 HTML 代码。

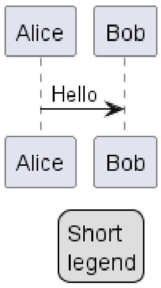
```
@startuml
Alice -> Bob: Authentication Request

header
<font color=red>Warning:</font>
Do not use in production.
endheader

center footer Generated for demonstration

@enduml
```



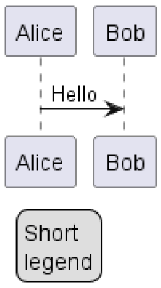
21.6 图例说明

legend 和 end legend 作为关键词，用于配置一个图例 (legend)。支持可选地使用 left,right,center 为这个图例指定对齐方式.

```
@startuml

Alice -> Bob : Hello
legend right
  Short
  legend
endlegend

@enduml
```



```
@startuml

Alice -> Bob : Hello
legend left
  Short
  legend
endlegend

@enduml
```

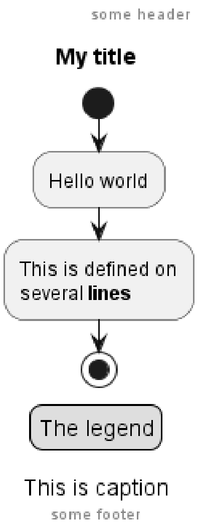


21.7 Appendix: Examples on all diagram

21.7.1 Activity

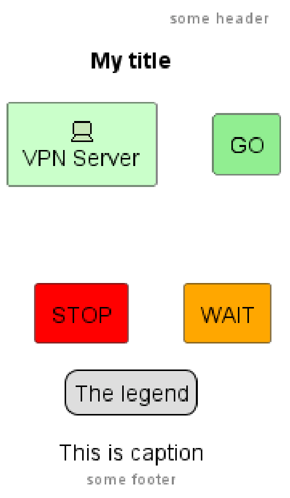
```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
```

```
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



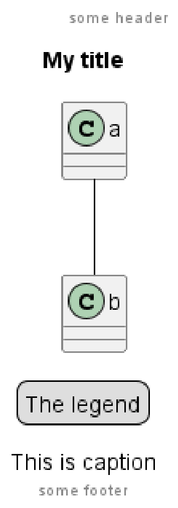
21.7.2 Archimate

```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml
```



21.7.3 Class

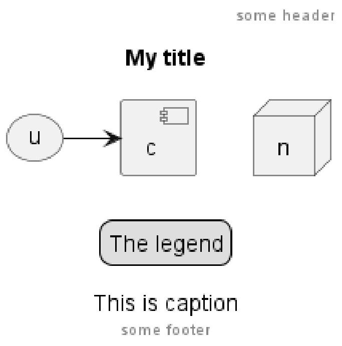
```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

a -- b

@enduml
```



21.7.4 Component, Deployment, Use-Case

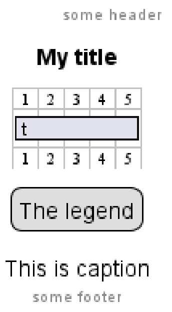
```
@startuml
header some header

footer some footer
```

```
title My title

caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml
```



21.7.5 Gantt project planning

```
@startgantt
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend


[t] lasts 5 days

@endgantt
```

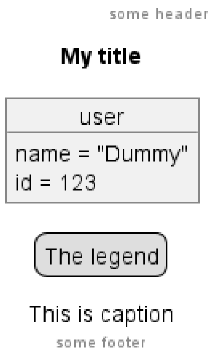


TODO: DONE *[(Header, footer) corrected on V1.2020.18]*

21.7.6 Object

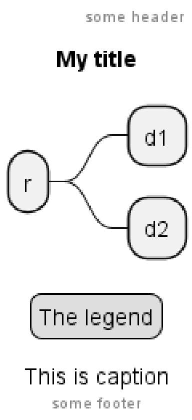
```
@startuml
```

```
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

object user {
  name = "Dummy"
  id = 123
}

@enduml
```



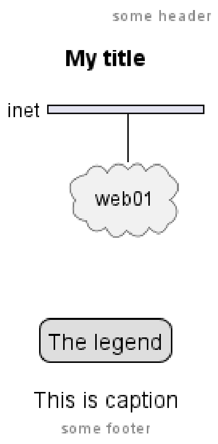
21.7.7 MindMap

```
@startmindmap
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```



21.7.8 Network (nwdiag)

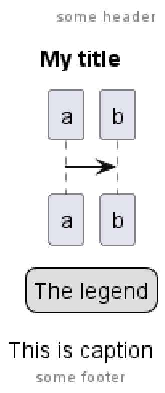
```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

nwdiag {
  network inet {
      web01 [shape = cloud]
  }
}

@enduml
```



21.7.9 Sequence

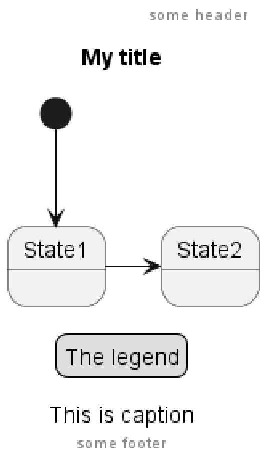
```
@startuml
header some header

footer some footer
```

```
title My title

caption This is caption

legend
The legend
end legend

a->b
@enduml
```



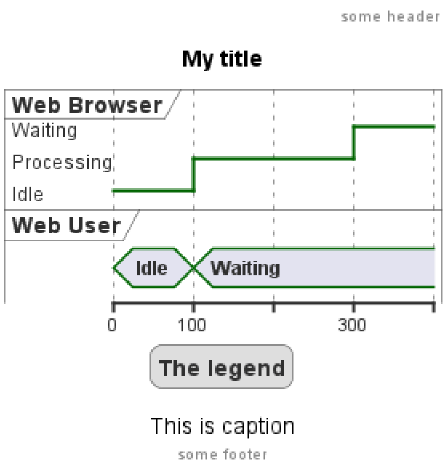
21.7.10 State

```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml
```



21.7.11 Timing

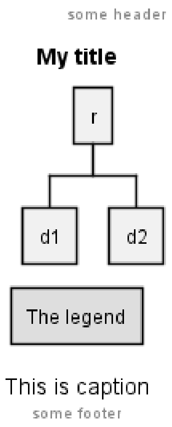
```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml
```



21.7.12 Work Breakdown Structure (WBS)

```
@startwbs
header some header

footer some footer

title My title

caption This is caption

legend
```

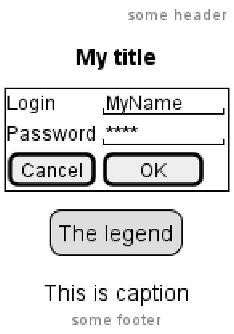
```
The legend
end legend

* r
** d1
** d2

@endwbs
```



TODO: DONE *[Corrected on V1.2020.17]*

21.7.13 Wireframe (SALT)

```
@startsalt
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
  Login     | "MyName   "
  Password | "****      "
  [Cancel] | [  OK   ]
}
@endsalt
```



TODO: DONE *[Corrected on V1.2020.18]*

21.8 Appendix: Examples on all diagram with style

TODO: DONE

FYI:

- all is only good for **`Sequence diagram`**
- `title`, `caption` and `legend` are good for all diagrams except for **`salt diagram`**

TODO: FIXME

- Now *(test on 1.2020.18-19)* `header`, `footer` are not good for **all other diagrams** except only for **`Sequence diagram`**.

To be fix; Thanks

TODO: FIXME

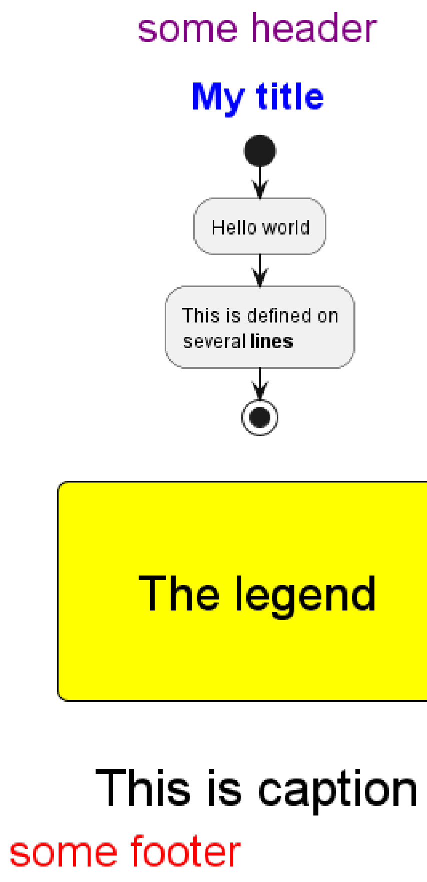
Here are tests of `title`, `header`, `footer`, `caption` or `legend` on all the diagram with the debug style:

```
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
```

21.8.1 Activity

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}
```

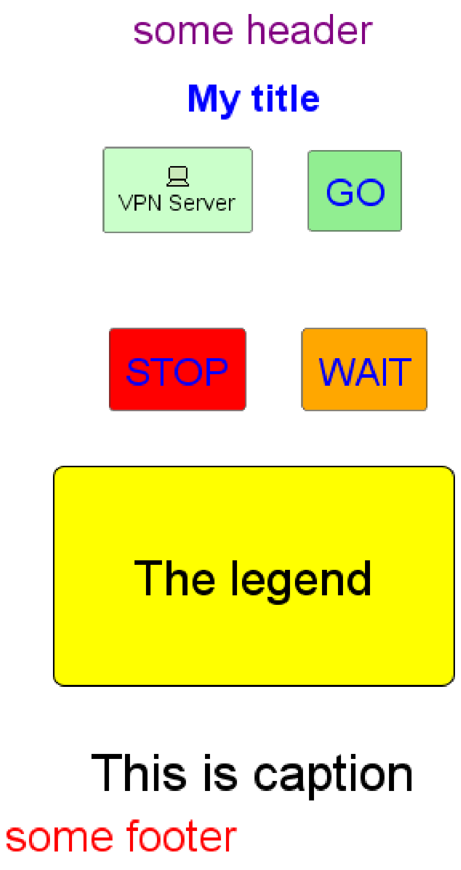
```
header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



21.8.2 Archimate

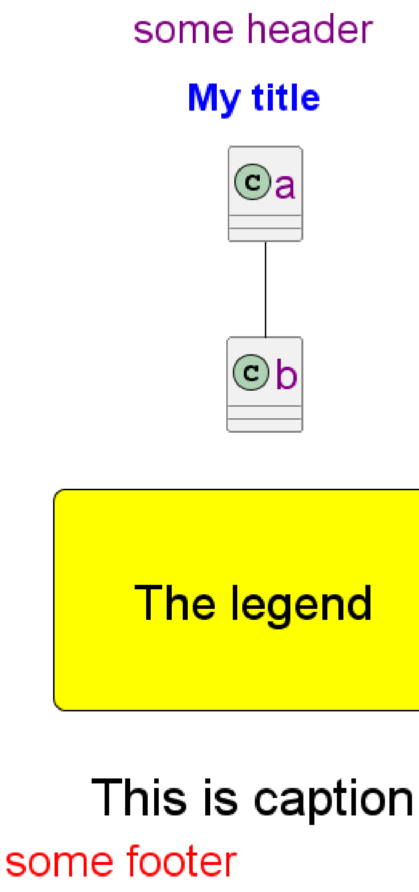
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}
```

```
caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml
```



21.8.3 Class

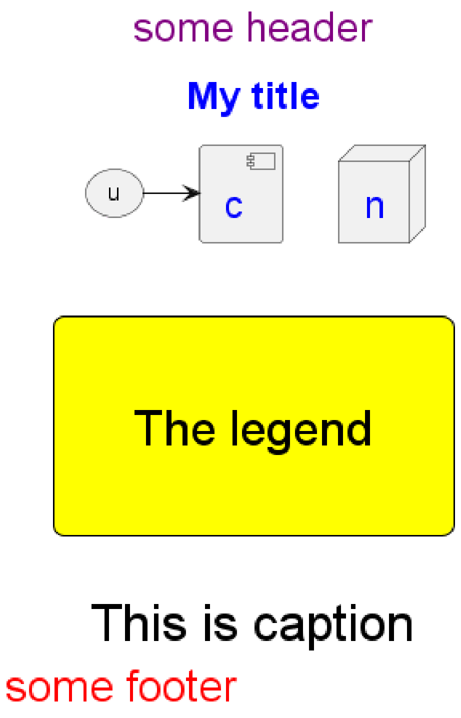
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

a -- b

@enduml
```



21.8.4 Component, Deployment, Use-Case

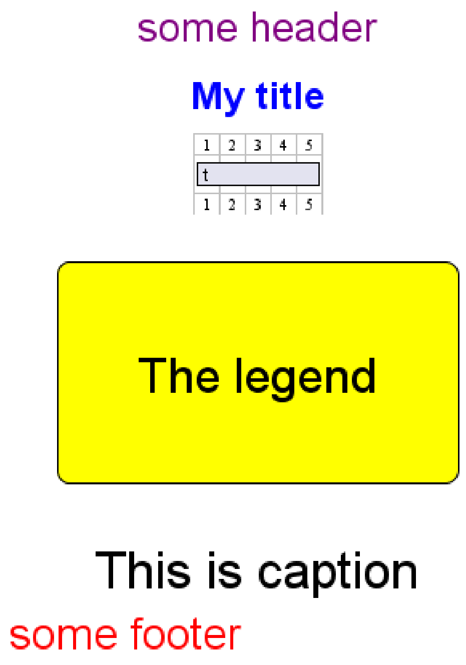
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}
```

```
caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml
```



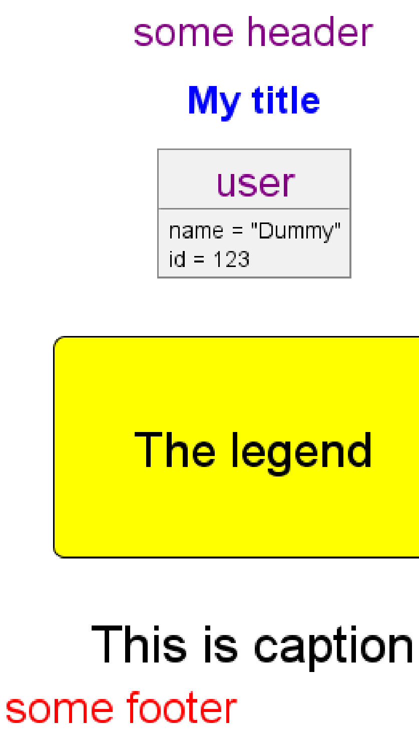
21.8.5 Gantt project planning

```
@startgantt
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend


[t] lasts 5 days

@endgantt
```



21.8.6 Object

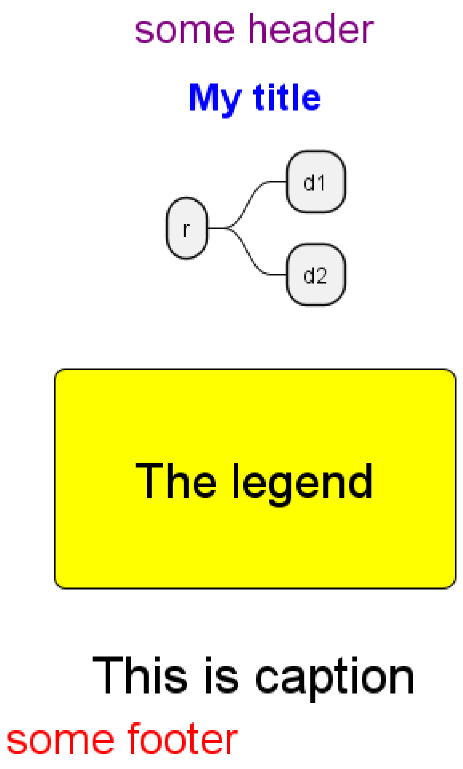
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

object user {
  name = "Dummy"
  id = 123
}

@enduml
```



21.8.7 MindMap

```
@startmindmap
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header
```

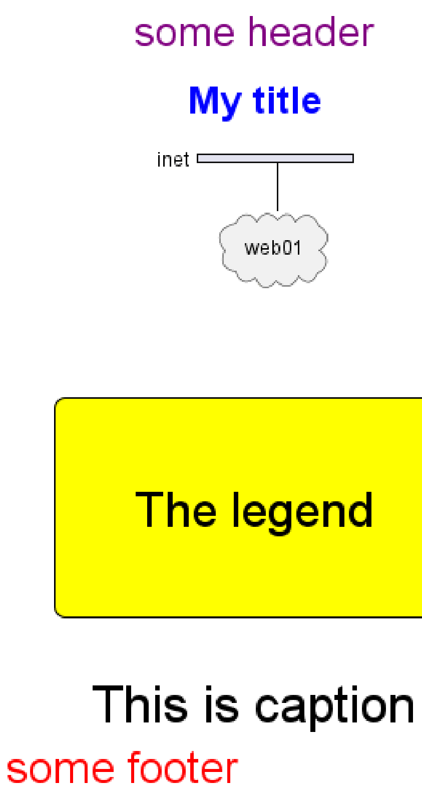
```
footer some footer

title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```



21.8.8 Network (nwdiag)

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}
```

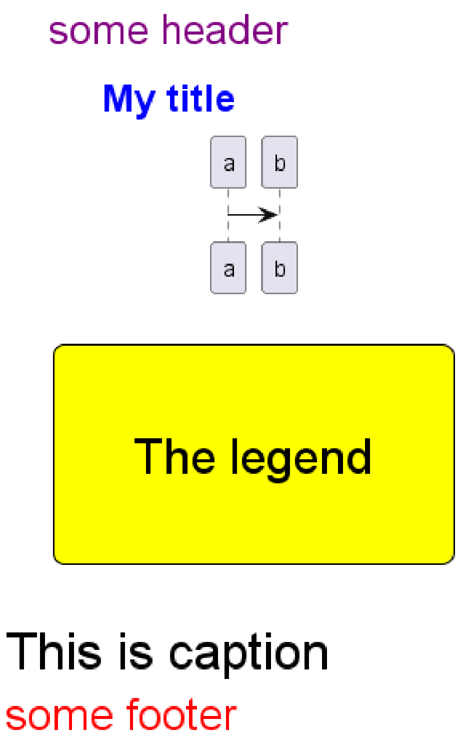
```
footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

nwdiag {
  network inet {
      web01 [shape = cloud]
  }
}

@enduml
```



21.8.9 Sequence

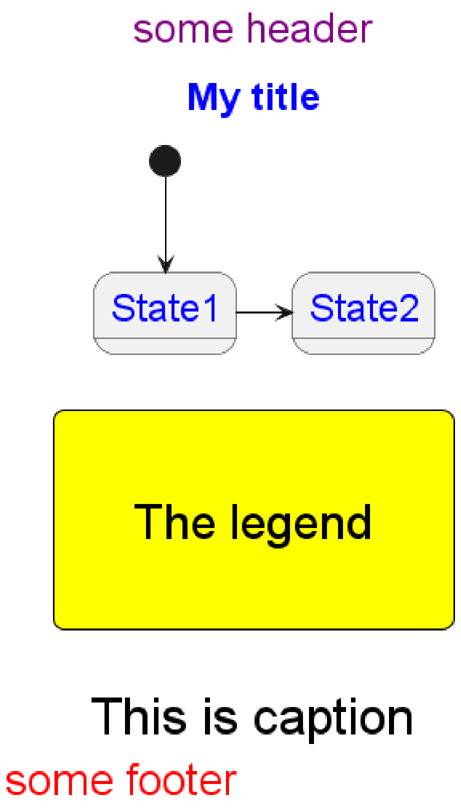
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
```

```
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

a->b
@enduml
```



21.8.10 State

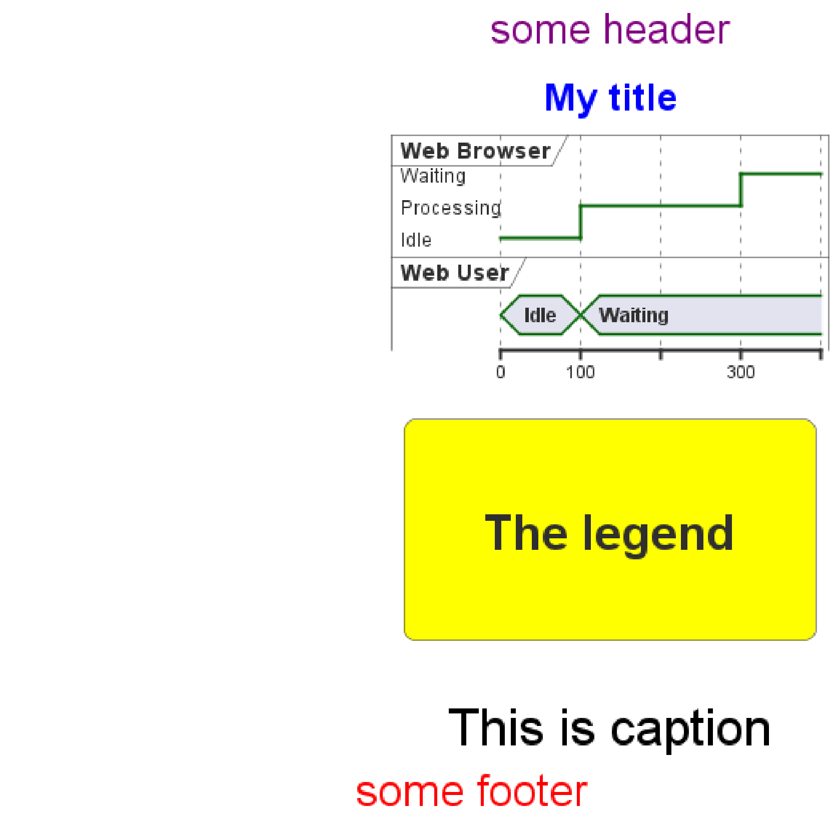
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml
```



21.8.11 Timing

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting
```

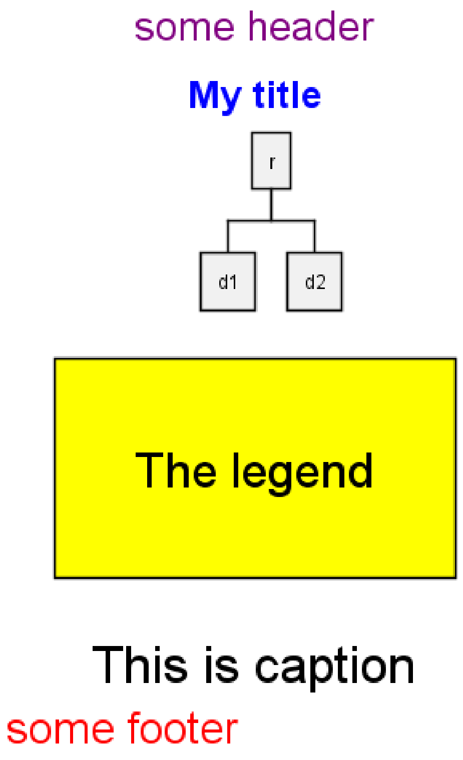
```
@enduml
```



21.8.12 Work Breakdown Structure (WBS)

```
@startwbs
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}
```

```
caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endwbs
```

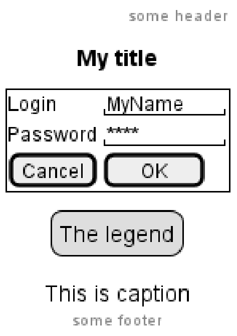


21.8.13 Wireframe (SALT)

TODO: FIXME Fix all (**`title`**, **`caption`**, **`legend`**, **`header`**, **`footer`**) for salt. **TODO:** FIXME

```
@startsalt
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}
```

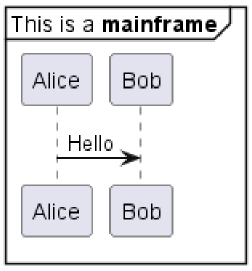
```
header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
@startsalt
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



21.9 Mainframe

```
@startuml
mainframe This is a **mainframe**
```

```
Alice->Bob : Hello
@enduml
```

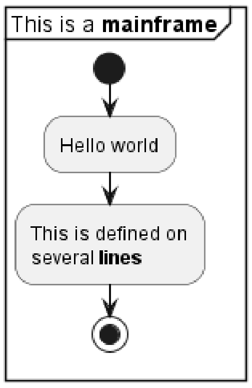


[Ref. QA-4019 and Issue#148]

21.10 附录。所有图表上的主机实例

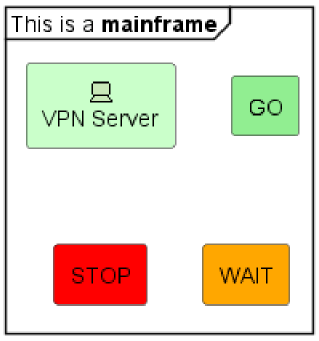
21.10.1 活动

```
@startuml
mainframe This is a **mainframe**

start
:Hello world;
:This is defined on
several **lines**;
stop
@enduml
```



21.10.2 架构

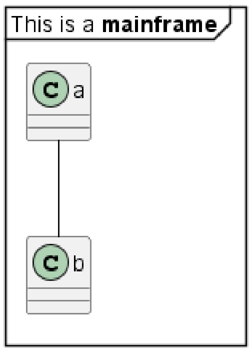
```
@startuml
mainframe This is a **mainframe**

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>
rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```



TODO: FIXME 栏目在顶部和左侧进行了裁剪 FIXME

21.10.3 类别

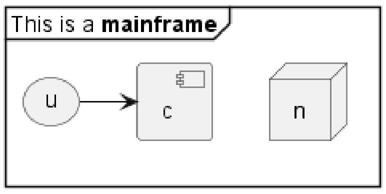
```
@startuml
mainframe This is a **mainframe**

a -- b
@enduml
```



TODO: FIXME 纪事在顶部和左侧进行了裁剪 FIXME

21.10.4 组件、部署、用例

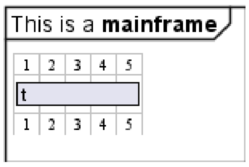
```
@startuml
mainframe This is a **mainframe**

node n
(u) -> [c]
@enduml
```



TODO: FIXME ︱顶部和左侧的裁剪 FIXME### ︱

21.10.5 甘特项目计划

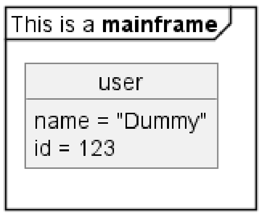
```
@startgantt
mainframe This is a **mainframe**

[t] lasts 5 days
@endgantt
```



TODO: FIXMEÌ 在顶部和左侧进行了裁剪 FIXME

21.10.6 对象

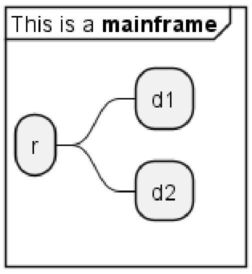
```
@startuml
mainframe This is a **mainframe**

object user {
  name = "Dummy"
  id = 123
}
@enduml
```



TODO: FIXME 丨顶部被裁剪！ FIXME

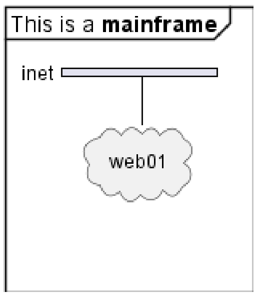
21.10.7 心智图谱

```
@startmindmap
mainframe This is a **mainframe**

* r
** d1
** d2
@endmindmap
```



21.10.8 网络 (nwdiag)

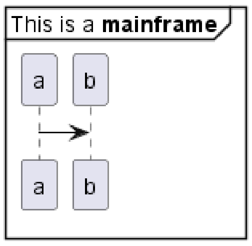
```
@startuml
mainframe This is a **mainframe**

nwdiag {
  network inet {
      web01 [shape = cloud]
  }
}
@enduml
```



TODO: FIXME 锦标赛顶部被裁剪！FIXME

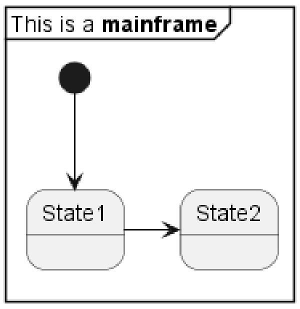
21.10.9 序列

```
@startuml
mainframe This is a **mainframe**

a->b
@enduml
```



21.10.10 状况

```
@startuml
mainframe This is a **mainframe**

[*] --> State1
State1 -> State2
@enduml
```

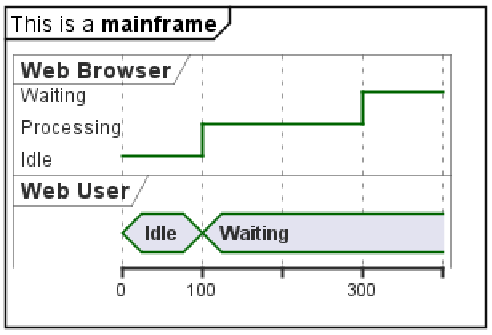


TODO: FIXME 锦标赛顶部和左侧被裁剪 FIXME

21.10.11 时间安排

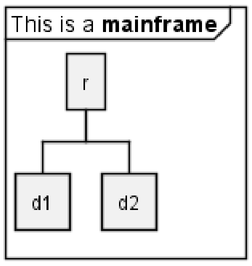
```
@startuml
mainframe This is a **mainframe**

robust "Web Browser" as WB
concise "Web User" as WU
@0
WU is Idle
```

```
WB is Idle
@100
WU is Waiting
WB is Processing
@300
WB is Waiting
@enduml
```



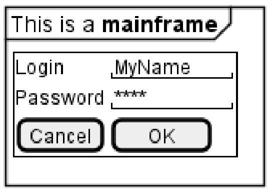
21.10.12 工作分解结构 (WBS)

```
@startwbs
mainframe This is a **mainframe**
* r
** d1
** d2
@endwbs
```



21.10.13 线框 (SALT)

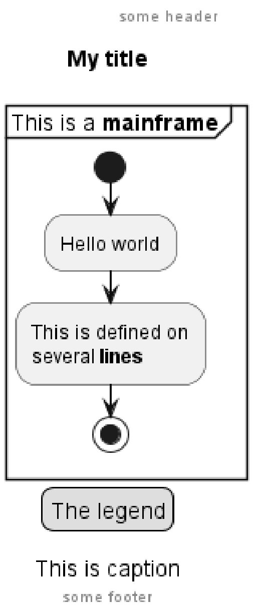
```
@startsalt
mainframe This is a **mainframe**
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



21.11 Appendix: Examples of title, header, footer, caption, legend and mainframe on all diagram

21.11.1 Activity

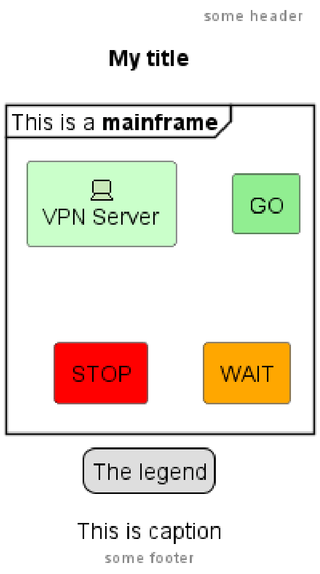
```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



21.11.2 Archimate

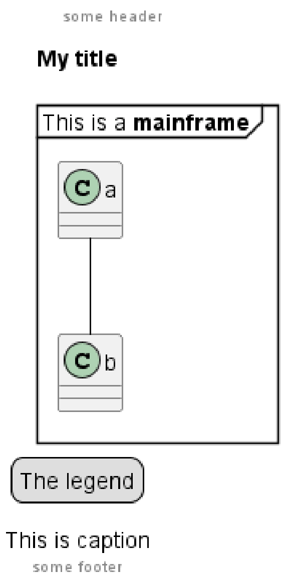
```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption
```

```
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml
```



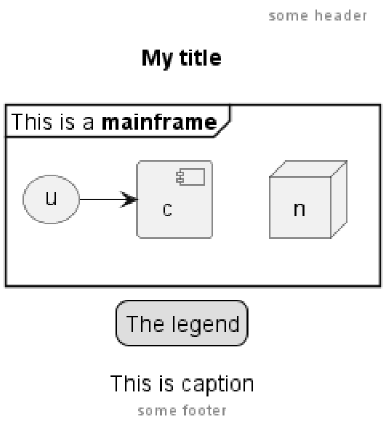
21.11.3 Class

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

a -- b

@enduml
```



21.11.4 Component, Deployment, Use-Case

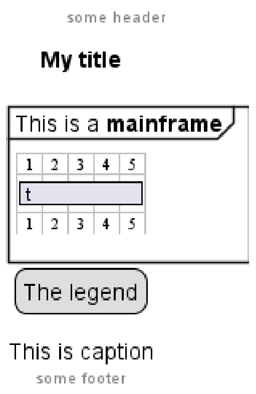
```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml
```



21.11.5 Gantt project planning

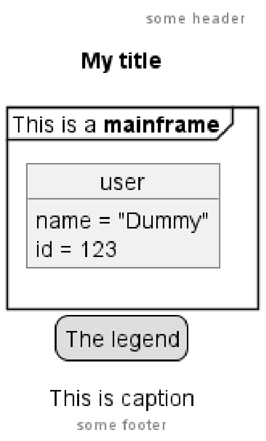
```
@startgantt
mainframe This is a **mainframe**
header some header
```

```
footer some footer

title My title

caption This is caption

legend
The legend
end legend


[t] lasts 5 days

@endgantt
```



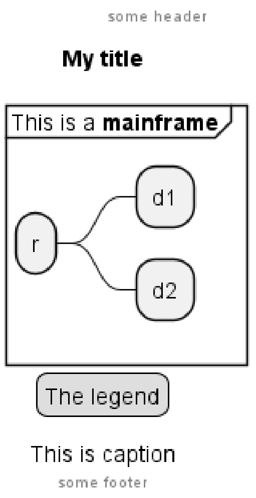
21.11.6 Object

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

object user {
  name = "Dummy"
  id = 123
}

@enduml
```



21.11.7 MindMap

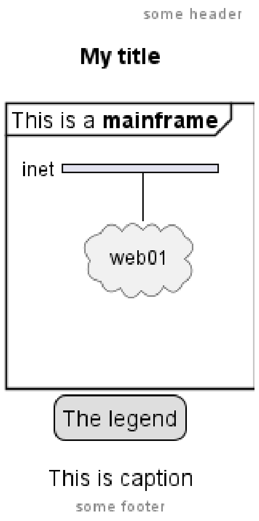
```
@startmindmap
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```



21.11.8 Network (nwdiag)

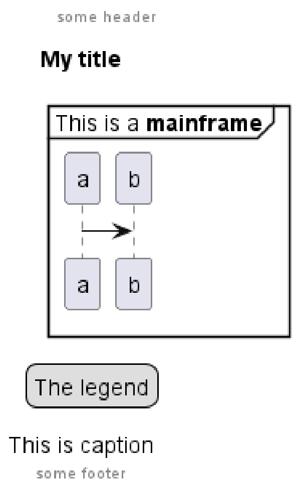
```
@startuml
mainframe This is a **mainframe**
header some header
```

```
footer some footer

title My title

caption This is caption

legend
The legend
end legend

nwdiag {
  network inet {
      web01 [shape = cloud]
  }
}

@enduml
```



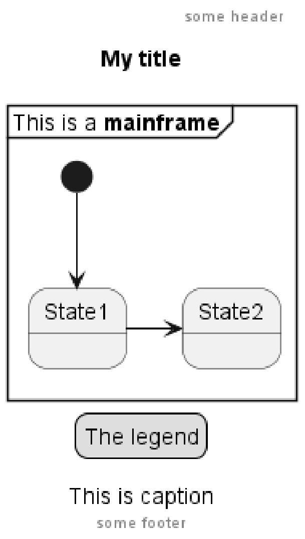
21.11.9 Sequence

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

a->b
@enduml
```



21.11.10 State

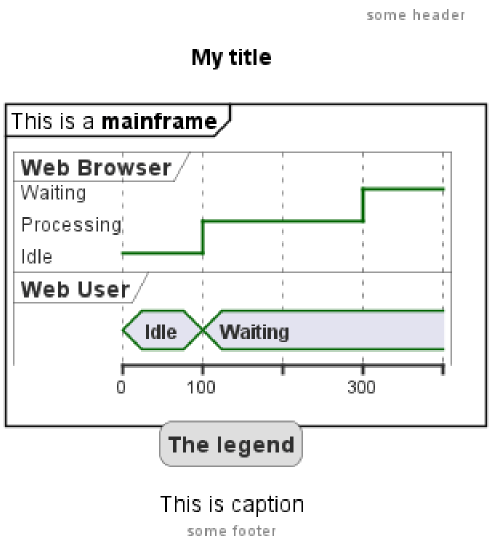
```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml
```



21.11.11 Timing

```
@startuml
mainframe This is a **mainframe**
```

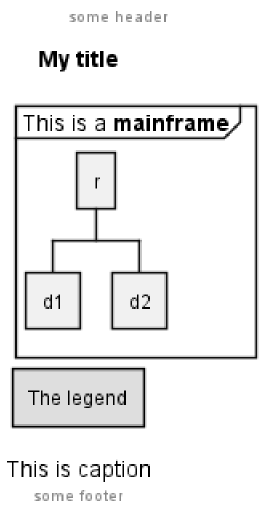
```
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml
```



21.11.12 Work Breakdown Structure (WBS)

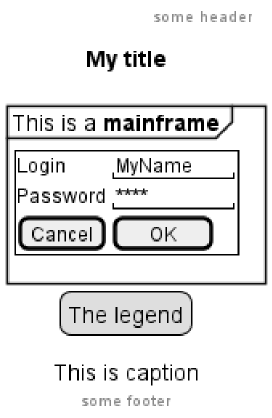
```
@startwbs
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption
```

```
legend
The legend
end legend

* r
** d1
** d2

@endwbs
```



21.11.13 Wireframe (SALT)

```
@startsalt
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```


some header
My title
This is a mainframe
Login
MyName
Password
****
Cancel
OK
The legend
This is caption
some footer

22 Creole

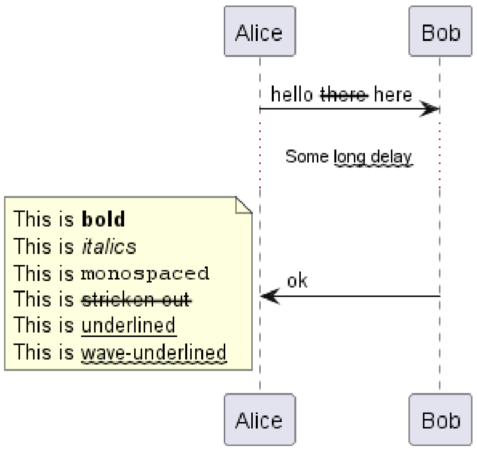
Creole is a lightweight common markup language for various wikis. A light-weight Creole engine is integrated in PlantUML to have a standardized way to emit styled text.

All diagrams support this syntax.

Note that compatibility with HTML syntax is preserved.

22.1 Emphasized text

```
@startuml
Alice -> Bob : hello --there-- here
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
end note
@enduml
```

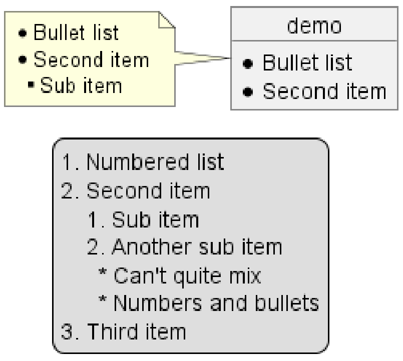


22.2 Lists

You can use numbered and bulleted lists in node text, notes, etc.

TODO: FIXME You cannot quite mix numbers and bullets in a list and its sublist.

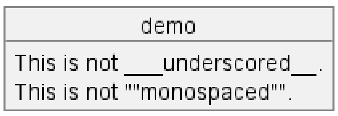
```
@startuml
object demo {
  * Bullet list
  * Second item
}
note left
  * Bullet list
  * Second item
  ** Sub item
end note

legend
  # Numbered list
  # Second item
  ## Sub item
  ## Another sub item
        * Can't quite mix
        * Numbers and bullets
  # Third item
end legend
@enduml
```



22.3 Escape character

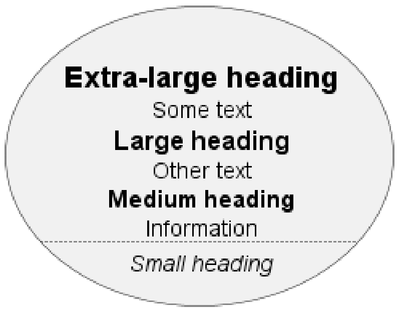
You can use the tilde ~ to escape special creole characters.

```
@startuml
object demo {
  This is not ~___underscored__.
  This is not ~""monospaced"".
}
@enduml
```



22.4 Headings

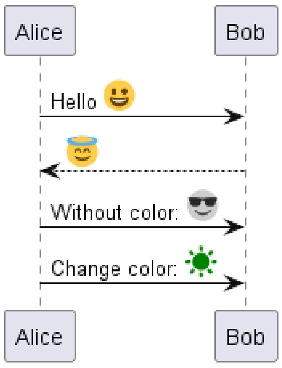
```
@startuml
usecase UC1 as "
= Extra-large heading
Some text
== Large heading
Other text
=== Medium heading
Information
....
==== Small heading"
@enduml
```



22.5 Emoji

All emojis from Twemoji (see EmojiTwo on Github) are available using the following syntax:

```
@startuml
Alice -> Bob : Hello <:1f600:>
return <:innocent:>
Alice -> Bob : Without color: <#0:sunglasses:>
Alice -> Bob : Change color: <#green:sunny:>
@enduml
```



Unlike Unicode Special characters that depend on installed fonts, the emoji are always available. Furthermore, emoji are already colored, but you can recolor them if you like (see examples above).

One can pick emoji from the emoji cheat sheet, the Unicode full-emoji-list, or the flat list emoji.txt in the plantuml source.

You can also use the following PlantUML command to list available emoji:

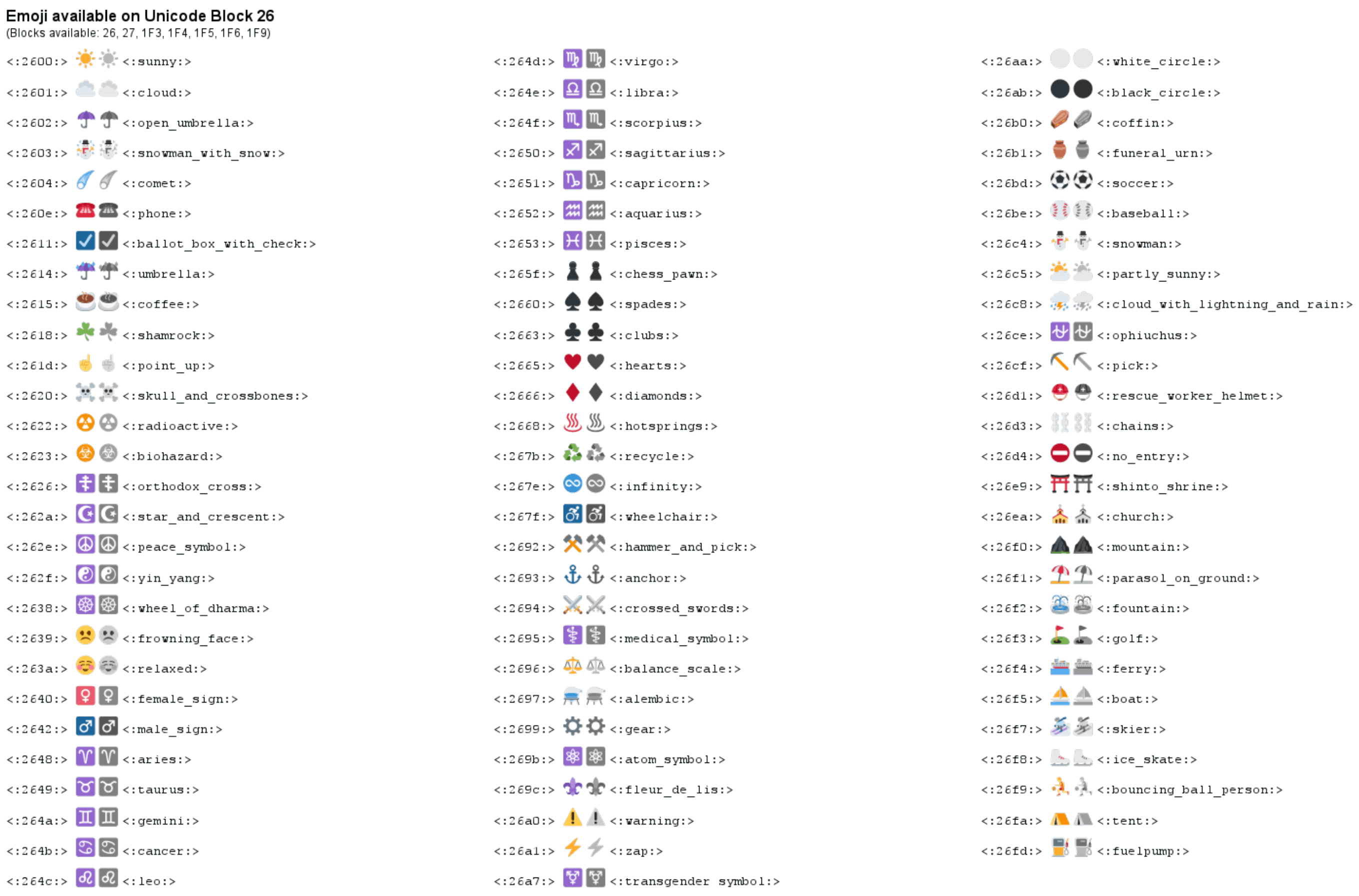
```
@startuml
emoji <block>
@enduml
```

As of 13 April 2023, you can select between 1174 emoji from the following Unicode blocks:

- Unicode block 26: 83 emoji
- Unicode block 27: 33 emoji
- Unicode block 1F3: 246 emoji
- Unicode block 1F4: 255 emoji
- Unicode block 1F5: 136 emoji
- Unicode block 1F6: 181 emoji
- Unicode block 1F9: 240 emoji

22.5.1 Unicode block 26

```
@startuml
emoji 26
@enduml
```

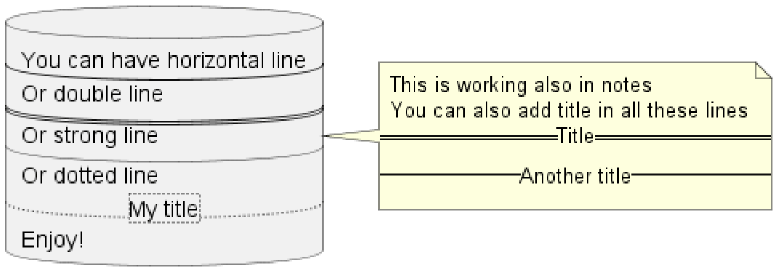


Emoji available on Unicode Block 26
(Blocks available: 26, 27, 1F3, 1F4, 1F5, 1F6, 1F9)

<:2600:>	☀ ☀	<:sunny:>	<:264d:>	♍ ♍	<:virgo:>	<:26aa:>	⚪ ⚪	<:white_circle:>
<:2601:>	☁ ☁	<:cloud:>	<:264e:>	♎ ♎	<:libra:>	<:26ab:>	⚫ ⚫	<:black_circle:>
<:2602:>	☂ ☂	<:open_umbrella:>	<:264f:>	♏ ♏	<:scorpius:>	<:26b0:>	⚰ ⚰	<:coffin:>
<:2603:>	☃ ☃	<:snowman_with_snow:>	<:2650:>	♐ ♐	<:sagittarius:>	<:26b1:>	⚱ ⚱	<:funeral_urn:>
<:2604:>	☄ ☄	<:comet:>	<:2651:>	♑ ♑	<:capricorn:>	<:26bd:>	⚽ ⚽	<:soccer:>
<:260e:>	☎ ☎	<:phone:>	<:2652:>	♒ ♒	<:aquarius:>	<:26be:>	⚾ ⚾	<:baseball:>
<:2611:>	☑ ☑	<:ballot_box_with_check:>	<:2653:>	♓ ♓	<:pisces:>	<:26c4:>	⛄ ⛄	<:snowman:>
<:2614:>	☔ ☔	<:umbrella:>	<:265f:>	♟ ♟	<:chess_pawn:>	<:26c5:>	⛅ ⛅	<:partly_sunny:>
<:2615:>	☕ ☕	<:coffee:>	<:2660:>	♠ ♠	<:spades:>	<:26c8:>	⛈ ⛈	<:cloud_with_lightning_and_rain:>
<:2618:>	☘ ☘	<:shamrock:>	<:2663:>	♣ ♣	<:clubs:>	<:26ce:>	⛎ ⛎	<:ophiuchus:>
<:261d:>	☝ ☝	<:point_up:>	<:2665:>	♥ ♥	<:hearts:>	<:26cf:>	⛏ ⛏	<:pick:>
<:2620:>	☠ ☠	<:skull_and_crossbones:>	<:2666:>	♦ ♦	<:diamonds:>	<:26d1:>	⛑ ⛑	<:rescue_worker_helmet:>
<:2622:>	☢ ☢	<:radioactive:>	<:2668:>	♨ ♨	<:hotsprings:>	<:26d3:>	⛓ ⛓	<:chains:>
<:2623:>	☣ ☣	<:biohazard:>	<:267b:>	♻ ♻	<:recycle:>	<:26d4:>	⛔ ⛔	<:no_entry:>
<:2626:>	☦ ☦	<:orthodox_cross:>	<:267e:>	♾ ♾	<:infinity:>	<:26e9:>	⛩ ⛩	<:shinto_shrine:>
<:262a:>	☪ ☪	<:star_and_crescent:>	<:267f:>	♿ ♿	<:wheelchair:>	<:26ea:>	⛪ ⛪	<:church:>
<:262e:>	☮ ☮	<:peace_symbol:>	<:2692:>	⚒ ⚒	<:hammer_and_pick:>	<:26f0:>	⛰ ⛰	<:mountain:>
<:262f:>	☯ ☯	<:yin_yang:>	<:2693:>	⚓ ⚓	<:anchor:>	<:26f1:>	⛱ ⛱	<:parasol_on_ground:>
<:2638:>	☸ ☸	<:wheel_of_dharma:>	<:2694:>	⚔ ⚔	<:crossed_swords:>	<:26f2:>	⛲ ⛲	<:fountain:>
<:2639:>	☹ ☹	<:frowning_face:>	<:2695:>	⚕ ⚕	<:medical_symbol:>	<:26f3:>	⛳ ⛳	<:golf:>
<:263a:>	☺ ☺	<:relaxed:>	<:2696:>	⚖ ⚖	<:balance_scale:>	<:26f4:>	⛴ ⛴	<:ferry:>
<:2640:>	♀ ♀	<:female_sign:>	<:2697:>	⚗ ⚗	<:alembic:>	<:26f5:>	⛵ ⛵	<:boat:>
<:2642:>	♂ ♂	<:male_sign:>	<:2699:>	⚙ ⚙	<:gear:>	<:26f7:>	⛷ ⛷	<:skier:>
<:2648:>	♈ ♈	<:aries:>	<:269b:>	⚛ ⚛	<:atom_symbol:>	<:26f8:>	⛸ ⛸	<:ice_skate:>
<:2649:>	♉ ♉	<:taurus:>	<:269c:>	⚜ ⚜	<:fleur_de_lis:>	<:26f9:>	⛹ ⛹	<:bouncing_ball_person:>
<:264a:>	♊ ♊	<:gemini:>	<:26a0:>	⚠ ⚠	<:warning:>	<:26fa:>	⛺ ⛺	<:tent:>
<:264b:>	♋ ♋	<:cancer:>	<:26a1:>	⚡ ⚡	<:zap:>	<:26fd:>	⛽ ⛽	<:fuelpump:>
<:264c:>	♌ ♌	<:leo:>	<:26a7:>	⚧ ⚧	<:transgender_symbol:>			

22.6 Horizontal lines

```
@startuml
database DB1 as "
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
Enjoy!
"
note right
  This is working also in notes
  You can also add title in all these lines
  ==Title==
  --Another title--
end note

@enduml
```


You can have horizontal line
Or double line
Or strong line
Or dotted line
My title
Enjoy!
This is working also in notes
You can also add title in all these lines
Title
Another title

22.7 Links

You can also use URL and links.

Simple links are define using two square brackets (or three square brackets for field or method on class diagram).

Example:

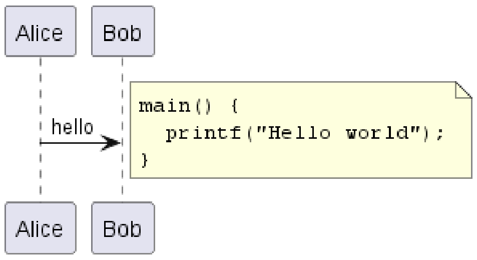
- `[[http://plantuml.com]]`
- `[[http://plantuml.com This label is printed]]`
- `[[http://plantuml.com{Optional tooltip} This label is printed]]`

URL can also be authenticated.

22.8 Code

You can use `<code>` to display some programming code in your diagram (sorry, syntax highlighting is not yet supported).

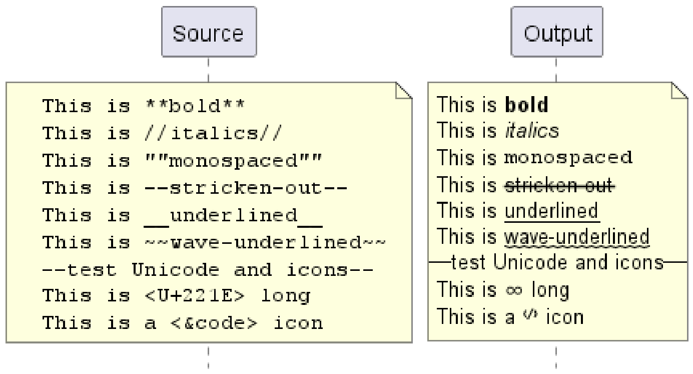
```
@startuml
Alice -> Bob : hello
note right
<code>
main() {
  printf("Hello world");
}
</code>
end note
@enduml
```



```
main() {
  printf("Hello world");
}
```

This is especially useful to illustrate some PlantUML code and the resulting rendering:

```
@startuml
hide footbox
note over Source
<code>
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
</code>
end note
/note over Output
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
end note
@enduml
```



22.9 Table

22.9.1 Create a table

It is possible to build table, with | separator.

```
@startuml
skinparam titleFontSize 14
title
  Example of simple table
  |= |= table |= header |
  | a | table | row |
  | b | table | row |
end title
[*] --> State1
@enduml
```

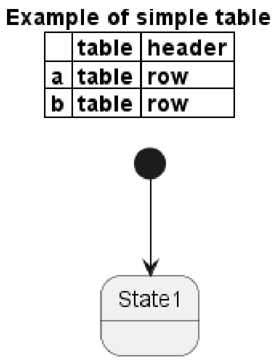


	table	header
a	table	row
b	table	row

22.9.2 Add color on rows or cells

You can specify background colors of rows and cells:

```
@startuml
start
:Here is the result
|= |= table |= header |
| a | table | row |
|<#FF8080> red |<#80FF80> green |<#8080FF> blue |
<#yellow>| b | table | row |;
@enduml
```

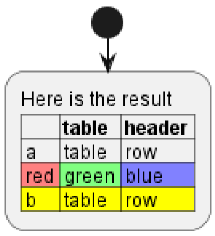
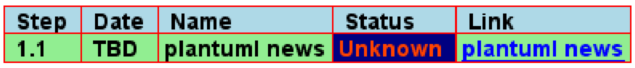


	table	header
a	table	row
red	green	blue
b	table	row

22.9.3 Add color on border and text

You can also specify colors of text and borders.

```
@startuml
title
<#lightblue,#red>|=  Step  |=  Date  |=  Name  |=  Status  |=  Link  |
<#lightgreen>|  1.1  |  TBD  | plantuml news |<#Navy><color:OrangeRed><b> Unknown  | [[https://plantu
end title
@enduml
```



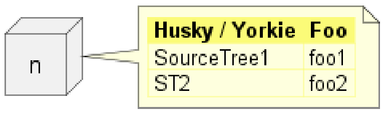
Step	Date	Name	Status	Link
1.1	TBD	plantuml news	Unknown	plantuml news

[Ref. QA-7184]

22.9.4 No border or same color as the background

You can also set the border color to the same color as the background.

```
@startuml
node n
note right of n
  <#FBFB77,#FBFB77>|= Husky / Yorkie |= Foo |
  | SourceTree1 | foo1 |
  | ST2 | foo2 |
end note
@enduml
```



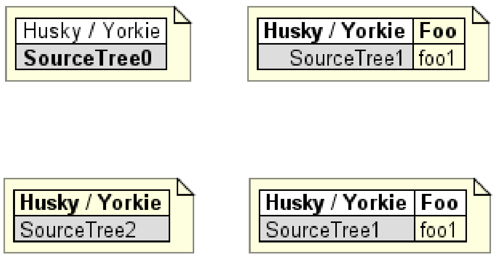
Husky / Yorkie	Foo
SourceTree1	foo1
ST2	foo2

[Ref. QA-12448]

22.9.5 Bold header or not

= as the first char of a cell indicates whether to make it bold (usually used for headers), or not.

```
@startuml
note as deepCSS0
  |<#white> Husky / Yorkie |
  |=<#gainsboro> SourceTree0 |
endnote

note as deepCSS1
  |= <#white> Husky / Yorkie |= Foo |
  |<#gainsboro><r> SourceTree1 | foo1 |
endnote

note as deepCSS2
  |= Husky / Yorkie |
  |<#gainsboro> SourceTree2 |
endnote

note as deepCSS3
  <#white>|= Husky / Yorkie |= Foo |
  |<#gainsboro> SourceTree1 | foo1 |
endnote
@enduml
```



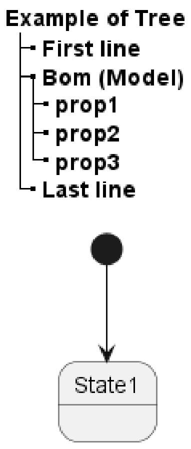
[Ref. QA-10923]

22.10 Tree

You can use |_ characters to build a tree.

On common commands, like title:

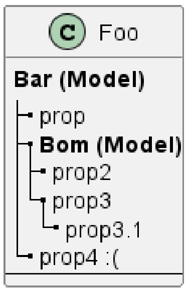
```
@startuml
skinparam titleFontSize 14
title
  Example of Tree
  |_ First line
  |_ **Bom (Model)**
    |_ prop1
    |_ prop2
    |_ prop3
  |_ Last line
end title
[*] --> State1
@enduml
```



On Class diagram.

(Please note how we have to use an empty second compartment, else the parentheses in **(Model)** cause that text to be moved to a separate first compartment):

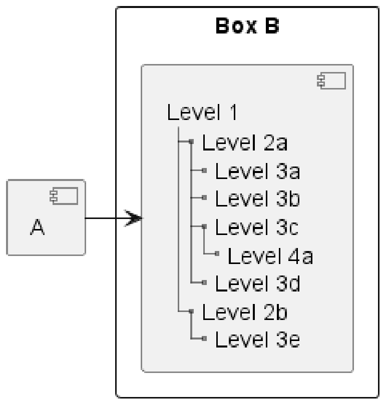
```
@startuml
class Foo {
**Bar (Model)**
|_ prop
|_ **Bom (Model)**
  |_ prop2
  |_ prop3
    |_ prop3.1
|_ prop4 :(
--
}
@enduml
```



[Ref. QA-3448]

On Component or Deployment diagrams:

```
@startuml
[A] as A
rectangle "Box B" {
    component B [
        Level 1
        |_ Level 2a
          |_ Level 3a
          |_ Level 3b
          |_ Level 3c
            |_ Level 4a
          |_ Level 3d
        |_ Level 2b
          |_ Level 3e
    ]
}
A -> B
@enduml
```

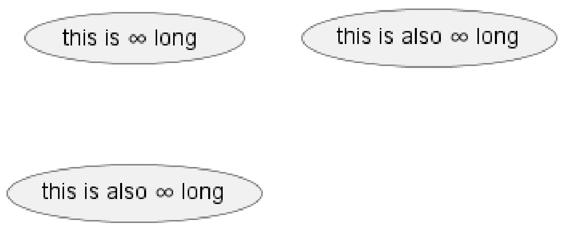


[Ref. QA-11365]

22.11 Special characters

It's possible to use any unicode character, either directly or with syntax `&#nnnnnn;` (decimal) or `<U+XXXXX>` (hex):

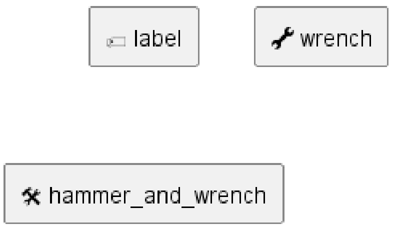
```
@startuml
usecase direct as "this is ∞ long"
usecase ampHash as "this is also ∞ long"
usecase angleBrackets as "this is also <U+221E> long"
@enduml
```



Please note that not all Unicode chars appear correctly, depending on installed fonts.

- You can use the listfonts command with a test string of your desired characters, to see which fonts may include them.
- For characters that are emoji, it's better to use the Emoji notation that doesn't depend on installed fonts, and the emoji are colored.
- The PlantUML server has the "Noto Emoji" font that has most emoji. If you want to render diagrams on your local system, you should check which fonts you have.
- Unfortunately "Noto Emoji" lacks normal chars, so you need to switch fonts, eg

```
@startuml
rectangle "<font:Noto Emoji><U+1F3F7></font> label"
rectangle "<font:Noto Emoji><U+1F527></font> wrench"
rectangle "<font:Noto Emoji><U+1F6E0></font> hammer_and_wrench"
@enduml
```



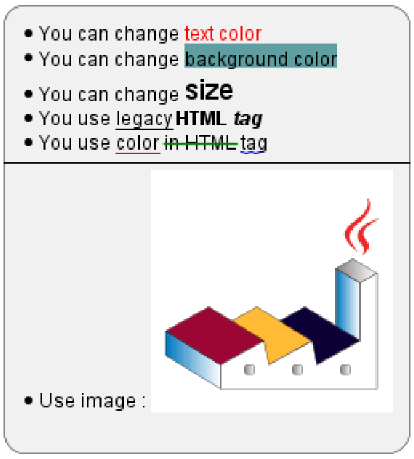
See Issue 72 for more details.

22.12 Legacy HTML

You can mix Creole with the following HTML tags:

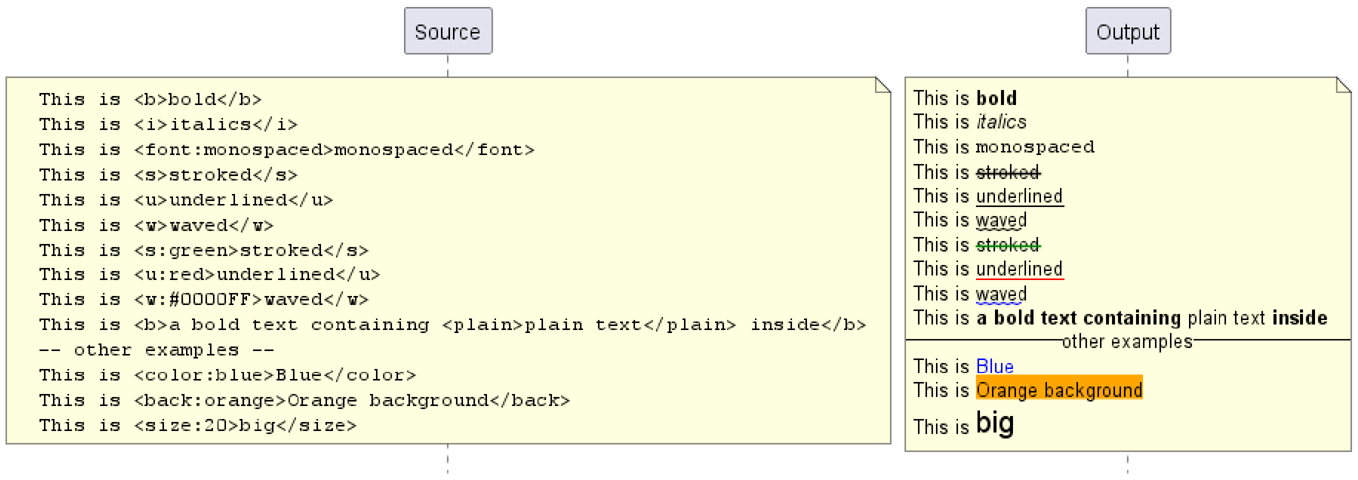
- `<b>` for bold text
- `<u>` or `<u:#AAAAAA>` or `<u:[[color|colorName]]>` for underline
- `<i>` for italic
- `<s>` or `<s:#AAAAAA>` or `<s:[[color|colorName]]>` for strike text
- `<w>` or `<w:#AAAAAA>` or `<w:[[color|colorName]]>` for wave underline text
- `<plain>` for plain text
- `<color:#AAAAAA>` or `<color:[[color|colorName]]>`
- `<back:#AAAAAA>` or `<back:[[color|colorName]]>` for background color
- `<size:nn>` to change font size
- `<img:file>` : the file must be accessible by the filesystem
- `<img:http://plantuml.com/logo3.png>` : the URL must be available from the Internet

```
@startuml
:* You can change <color:red>text color</color>
* You can change <back:cadetblue>background color</back>
* You can change <size:18>size</size>
* You use <u>legacy</u> <b>HTML <i>tag</i></b>
* You use <u:red>color</u> <s:green>in HTML</s> <w:#0000FF>tag</w>
----
* Use image : <img:http://plantuml.com/logo3.png>
;
@enduml
```


• You can change text color
• You can change background color
• You can change size
• You use legacy HTML tag
• You use color ~~in HTML~~ tag
• Use image :

22.12.1 Common HTML element

```
@startuml
hide footbox
note over Source
<code>
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
  This is <b>a bold text containing <plain>plain text</plain> inside</b>
  -- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
</code>
end note
/note over Output
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
  This is <b>a bold text containing <plain>plain text</plain> inside</b>
  -- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
end note
@enduml
```



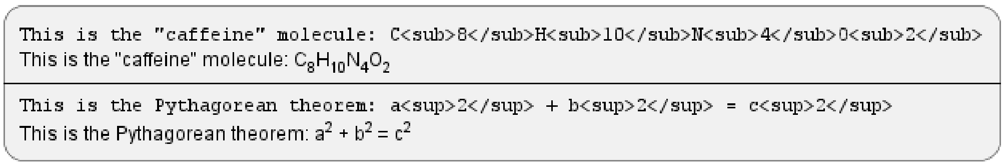
```
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
This is <b>a bold text containing <plain>plain text</plain> inside</b>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
```

This is **bold**
This is *italics*
This is `monospaced`
This is ~~stroked~~
This is underlined
This is waved
This is ~~stroked~~
This is underlined
This is waved
This is **a bold text containing** plain text **inside**
other examples
This is Blue
This is Orange background
This is big

[Ref. QA-5254 for `plain`*]*

22.12.2 Subscript and Superscript element [sub, sup]

```
@startuml
:<code>
This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
</code>
This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
----
<code>
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>
</code>
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>;
@enduml
```

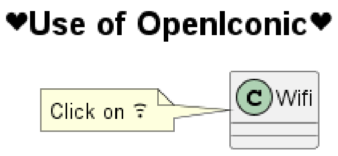

This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
This is the "caffeine" molecule: $C_8H_{10}N_4O_2$
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>
This is the Pythagorean theorem: $a^2 + b^2 = c^2$

22.13 OpenIconic

OpenIconic is a very nice open-source icon set. Those icons are integrated in the creole parser, so you can use them out-of-the-box.

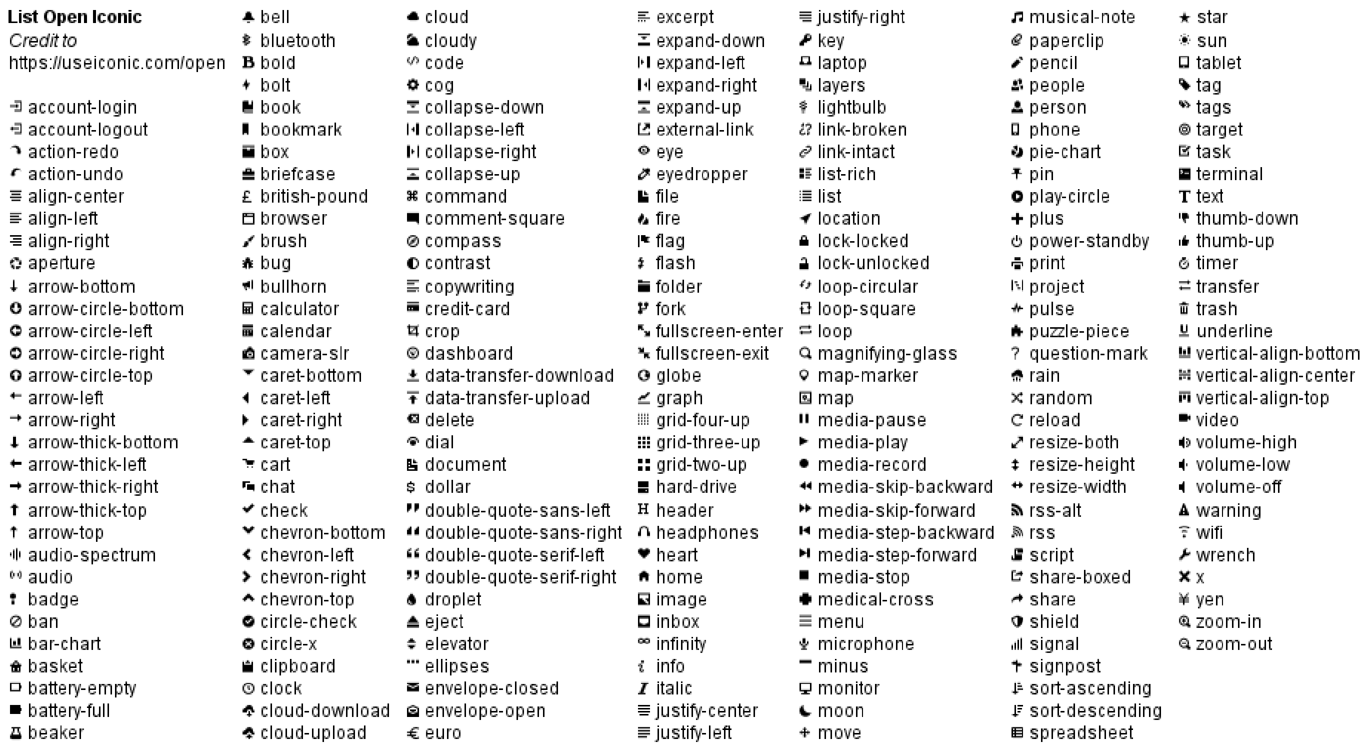
Use the following syntax: `<&ICON_NAME>`.

```
@startuml
title: <size:20><&heart>Use of OpenIconic<&heart></size>
class Wifi
note left
  Click on <&wifi>
end note
@enduml
```



The complete list is available with the following special command:

```
@startuml
listopeniconic
@enduml
```

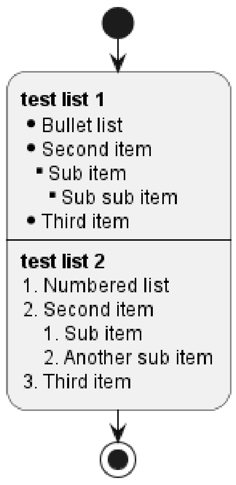


List Open Iconic	bell	cloud	excerpt	justify-right	musical-note	star
Credit to	bluetooth	cloudy	expand-down	key	paperclip	sun
https://useiconic.com/open	bold	code	expand-left	laptop	pencil	tablet
	bolt	cog	expand-right	layers	people	tag
account-login	book	collapse-down	expand-up	lightbulb	person	tags
account-logout	bookmark	collapse-left	external-link	link-broken	phone	target
action-redo	box	collapse-right	eye	link-intact	pie-chart	task
action-undo	briefcase	collapse-up	eyedropper	list-rich	pin	terminal
align-center	british-pound	command	file	list	play-circle	text
align-left	browser	comment-square	fire	location	plus	thumb-down
align-right	brush	compass	flag	lock-locked	power-standby	thumb-up
aperture	bug	contrast	flash	lock-unlocked	print	timer
arrow-bottom	bullhorn	copywriting	folder	loop-circular	project	transfer
arrow-circle-bottom	calculator	credit-card	fork	loop-square	pulse	trash
arrow-circle-left	calendar	crop	fullscreen-enter	loop	puzzle-piece	underline
arrow-circle-right	camera-slr	dashboard	fullscreen-exit	magnifying-glass	question-mark	vertical-align-bottom
arrow-circle-top	caret-bottom	data-transfer-download	globe	map-marker	rain	vertical-align-center
arrow-left	caret-left	data-transfer-upload	graph	map	random	vertical-align-top
arrow-right	caret-right	delete	grid-four-up	media-pause	reload	video
arrow-thick-bottom	caret-top	dial	grid-three-up	media-play	resize-both	volume-high
arrow-thick-left	cart	document	grid-two-up	media-record	resize-height	volume-low
arrow-thick-right	chat	dollar	hard-drive	media-skip-backward	resize-width	volume-off
arrow-thick-top	check	double-quote-sans-left	header	media-skip-forward	rss-alt	warning
arrow-top	chevron-bottom	double-quote-sans-right	headphones	media-step-backward	rss	wifi
audio-spectrum	chevron-left	double-quote-serif-left	heart	media-step-forward	script	wrench
audio	chevron-right	double-quote-serif-right	home	media-stop	share-boxed	x
badge	chevron-top	droplet	image	medical-cross	share	yen
ban	circle-check	eject	inbox	menu	shield	zoom-in
bar-chart	circle-x	elevator	infinity	microphone	signal	zoom-out
basket	clipboard	ellipses	info	minus	signpost	
battery-empty	clock	envelope-closed	italic	monitor	sort-ascending	
battery-full	cloud-download	envelope-open	justify-center	moon	sort-descending	
beaker	cloud-upload	euro	justify-left	move	spreadsheet	

22.14 Appendix: Examples of "Creole List" on all diagrams

22.14.1 Activity

```
@startuml
start
:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
stop
@enduml
```



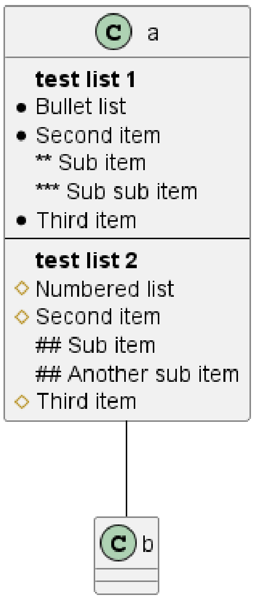
22.14.2 Class

TODO: FIXME

- *Sub item*
- *Sub sub item*

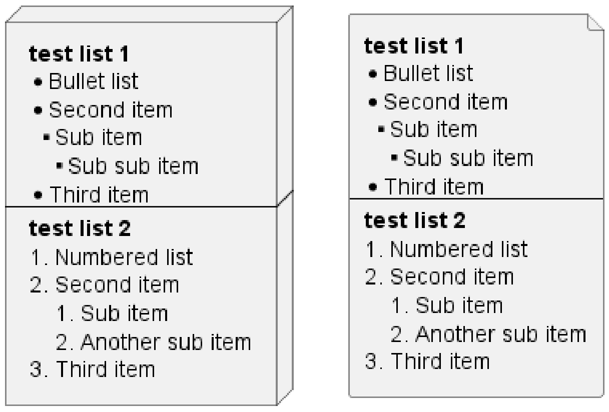
TODO: FIXME

```
@startuml

class a {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

a -- b

@enduml
```



22.14.3 Component, Deployment, Use-Case

```
@startuml
node n [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

file f as "
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
"
@enduml
```



TODO: DONE *[Corrected in V1.2020.18]*

22.14.4 Gantt project planning

N/A

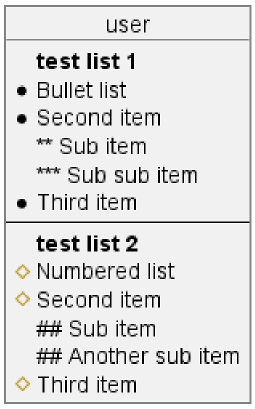
22.14.5 Object

TODO: FIXME

- *Sub item*
- *Sub sub item*

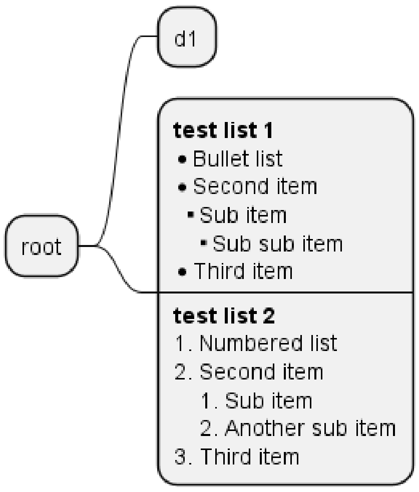
TODO: FIXME

```
@startuml
object user {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

@enduml
```



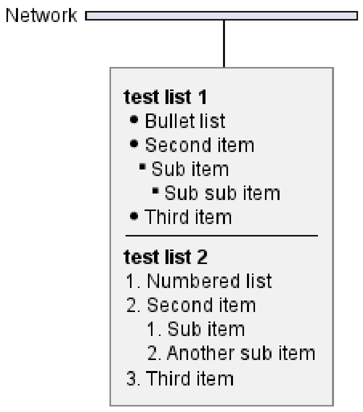
22.14.6 MindMap

```
@startmindmap

* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;


@endmindmap
```



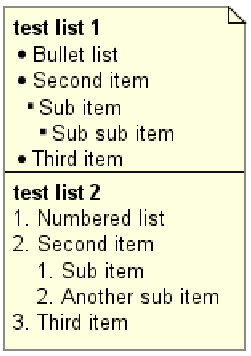
22.14.7 Network (nwdiag)

```
@startuml
nwdiag {
  network Network {
      Server [description="**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub i
}
@enduml
```



22.14.8 Note

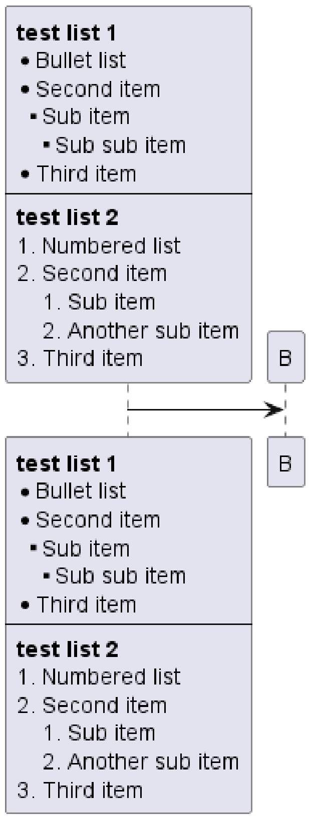
```
@startuml
note as n
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
end note
@enduml
```



22.14.9 Sequence

```
@startuml
<style>
participant {HorizontalAlignment left}
</style>
```

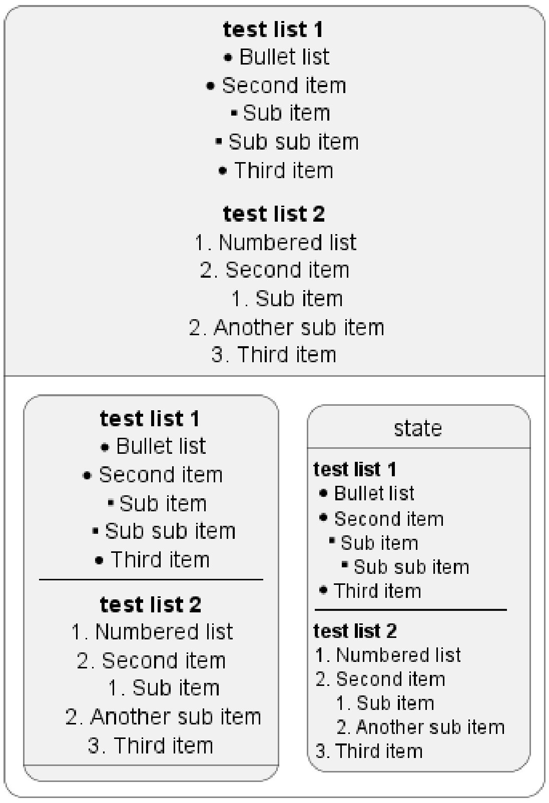
```
participant Participant [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

participant B

Participant -> B
@enduml
```



[Ref. QA-15232]

22.14.10 State

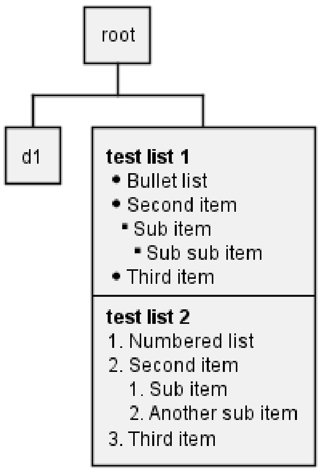
```
@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n-
state "**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n-
state : **test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n-
}
@enduml
```



[Ref. QA-16978]

22.14.11 WBS

```
@startwbs

* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;

@endwbs
```

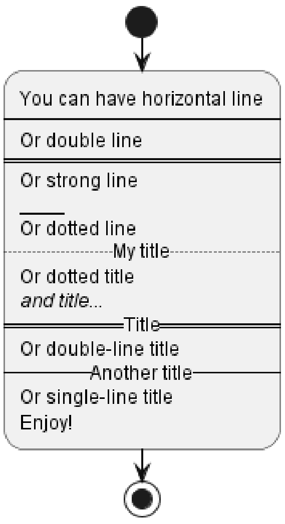


22.15 Appendix: Examples of "Creole horizontal lines" on all diagrams

22.15.1 Activity

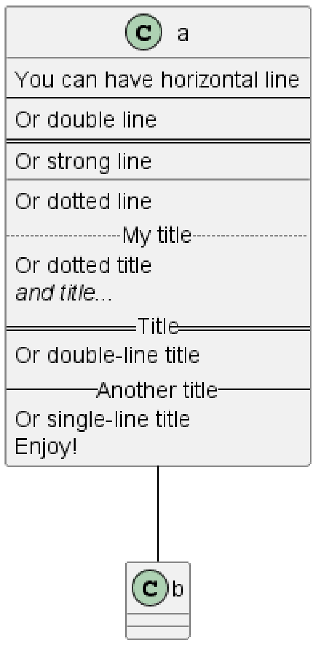
TODO: FIXME strong line ____ **TODO:** FIXME

```
@startuml
start
:You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
stop
@enduml
```



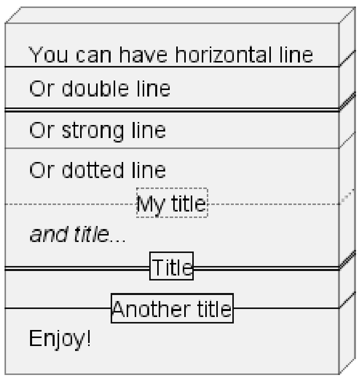
22.15.2 Class

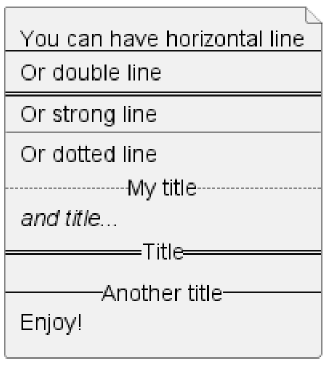
```
@startuml

class a {
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}

a -- b

@enduml
```

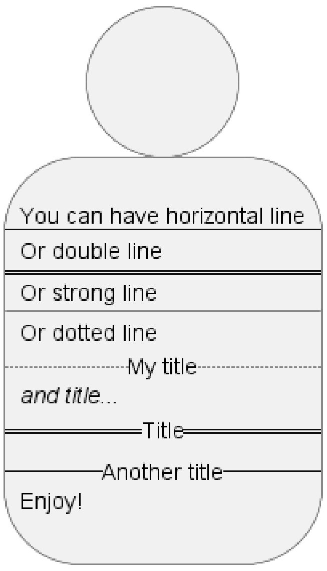


22.15.3 Component, Deployment, Use-Case

```
@startuml
node n [
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
]

file f as "
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
"

person p [

You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!

]
@enduml
```





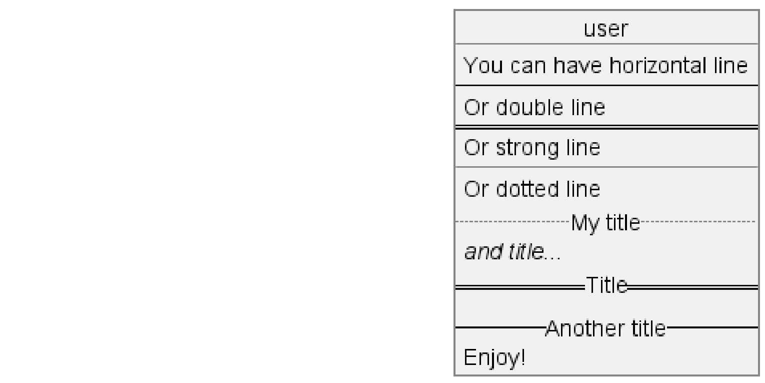


22.15.4 Gantt project planning

N/A

22.15.5 Object

```
@startuml
object user {
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
}

@enduml
```

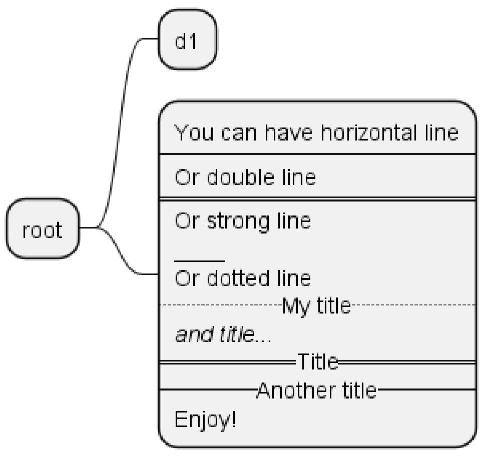


TODO: DONE *[Corrected on V1.2020.18]*

22.15.6 MindMap

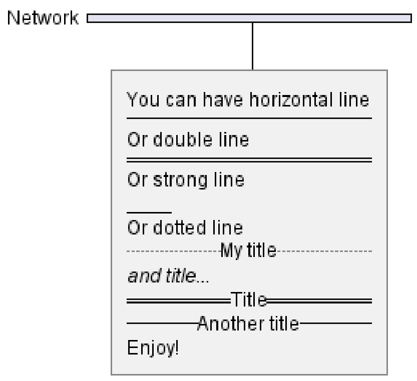
TODO: FIXME strong line ____ **TODO:** FIXME

```
@startmindmap

* root
** d1
**:You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!;

@endmindmap
```



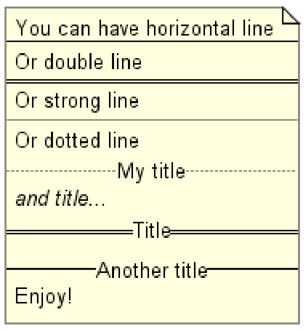
22.15.7 Network (nwdiag)

```
@startuml
nwdiag {
  network Network {
      Server [description="You can have horizontal line\n----\nOr double line\n====\nOr strong line\
}
@enduml
```



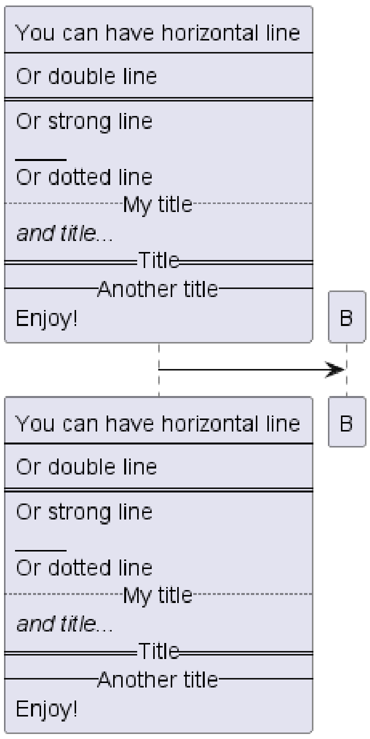
22.15.8 Note

```
@startuml
note as n
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
end note
@enduml
```



22.15.9 Sequence

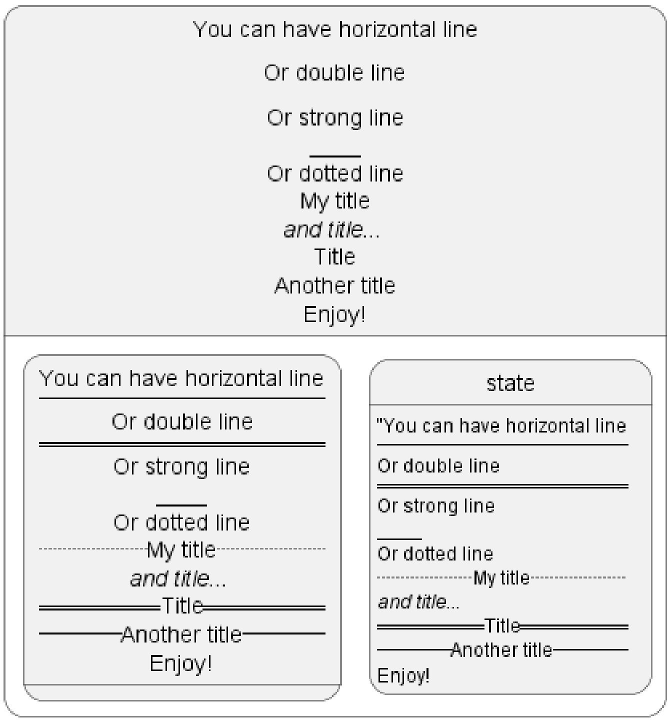
```
@startuml
<style>
participant {HorizontalAlignment left}
</style>
participant Participant [
You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
]

participant B

Participant -> B
@enduml
```



[Ref. QA-15232]

22.15.10 State

```
@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n____\nOr dotted line
state "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n____\nOr dotted line
state : "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n____\nOr dotted li
}
@enduml
```

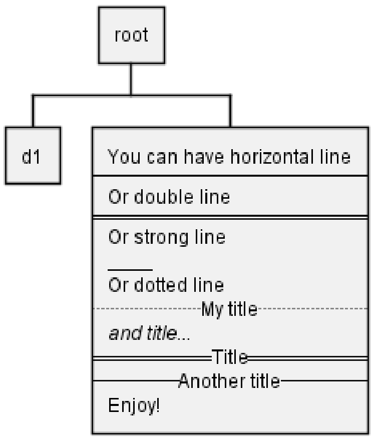


[Ref. QA-16978]

22.15.11 WBS

TODO: FIXME strong line ____ **TODO:** FIXME

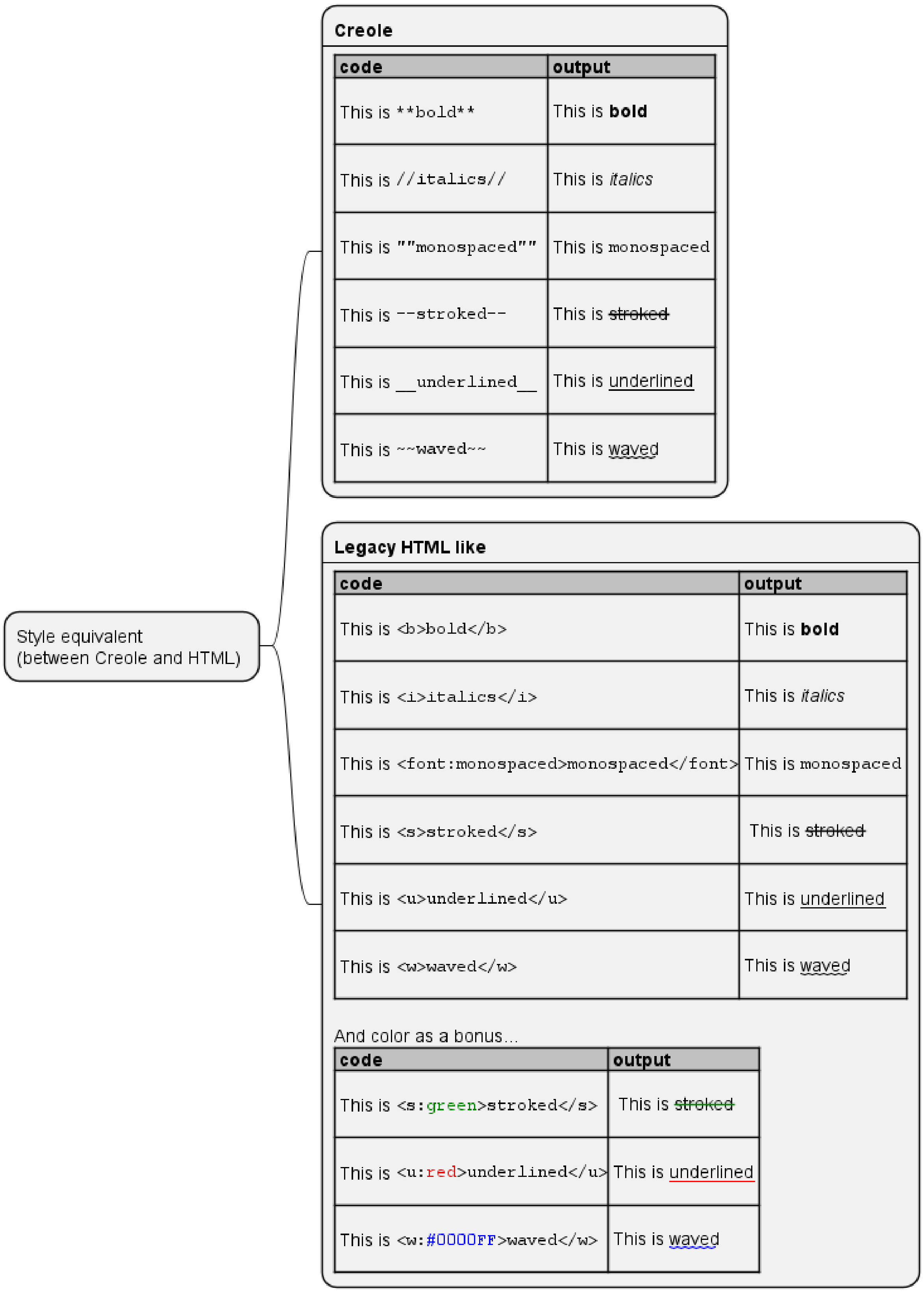
```
@startwbs

* root
** d1
**:You can have horizontal line
----
Or double line
====
Or strong line
____
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!;

@endwbs
```



22.16 Style equivalent (between Creole and HTML)

Style	Creole	Legacy HTML like
bold	This is `**bold**`	This is `<b>bold</b>`
italics	This is `//italics//`	This is `<i>italics</i>`
`monospaced`	This is `""monospaced""`	This is `<font:monospaced>monospaced</font>`
stroked	This is `--stroked--`	This is `<s>stroked</s>`
underlined	This is `__underlined__`	This is `<u>underlined</u>`
waved	This is `~~~`	This is `<w>waved</w>`

```
@startmindmap
* Style equivalent\n(between Creole and HTML)
**:**Creole**
----
<#silver>|= code|= output|
| \n This is ""~**bold**""\n | \n This is **bold** |
| \n This is ""~//italics//""\n | \n This is //italics// |
| \n This is ""~""monospaced~"" ""\n | \n This is ""monospaced"" |
| \n This is ""~--stroked--""\n | \n This is --stroked-- |
| \n This is ""~__underlined__""\n |  \n This is __underlined__ |
| \n This is ""<U+007E><U+007E>waved<U+007E><U+007E>""\n | \n This is ~~waved~~ |;
**:<b>Legacy HTML like
----
<#silver>|= code|= output|
| \n This is ""~<b>bold</b>""\n | \n This is <b>bold</b> |
| \n This is ""~<i>italics</i>""\n | \n This is <i>italics</i> |
| \n This is ""~<font:monospaced>monospaced</font>""\n | \n This is <font:monospaced>monospaced</fon
| \n This is ""~<s>stroked</s>""\n | \n  This is <s>stroked</s> |
| \n This is ""~<u>underlined</u>""\n | \n This is <u>underlined</u> |
| \n This is ""~<w>waved</w>""\n | \n This is <w>waved</w> |

And color as a bonus...
<#silver>|= code|= output|
| \n This is ""~<s:""<color:green>""green""</color>"">stroked</s>""\n | \n  This is <s:green>stroked
| \n This is ""~<u:""<color:red>""red""</color>"">underlined</u>""\n | \n This is <u:red>underlined<
| \n This is ""~<w:""<color:#0000FF>""#0000FF""</color>"">waved</w>""\n | \n This is <w:#0000FF>wave
@endmindmap
```



Creole

code	output
This is `**bold**`	This is **bold**
This is `//italics//`	This is *italics*
This is `""monospaced""`	This is `monospaced`
This is `--stroked--`	This is ~~stroked~~
This is `__underlined__`	This is underlined
This is `~~waved~~`	This is waved

Legacy HTML like

code	output
This is `<b>bold</b>`	This is **bold**
This is `<i>italics</i>`	This is *italics*
This is `<font:monospaced>monospaced</font>`	This is `monospaced`
This is `<s>stroked</s>`	This is ~~stroked~~
This is `<u>underlined</u>`	This is underlined
This is `<w>waved</w>`	This is waved

And color as a bonus...

code	output
This is `<s:green>stroked</s>`	This is ~~stroked~~
This is `<u:red>underlined</u>`	This is underlined
This is `<w:#0000FF>waved</w>`	This is waved

23 Defining and using sprites

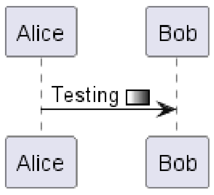
A *Sprite* is a small graphic element that can be used in diagrams.

In PlantUML, sprites are monochrome and can have either 4, 8 or 16 gray level.

To define a sprite, you have to use a hexadecimal digit between 0 and F per pixel.

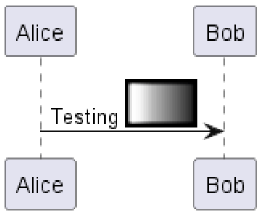
Then you can use the sprite using `<$XXX>` where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1>
@enduml
```



You can scale the sprite.

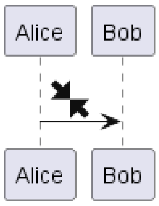
```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```



23.1 Inline SVG sprite

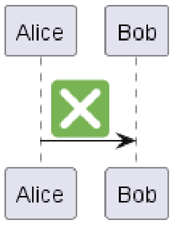
You can also use inlined SVG for sprites. Only a tiny subset of SVG directives is possible, so you probably have to compress existing SVG files using https://vecta.io/nano.

```
@startuml
sprite foo1 <svg width="8" height="8" viewBox="0 0 8 8"><path d="M1 0l-1 1 1.5 1.5-1.5 1.5h4v-4l-1.5

Alice->Bob : <$foo1*3>
@enduml
```



Another example:

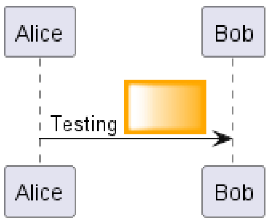
```
@startuml
sprite foo1 <svg viewBox="0 0 36 36">
<path fill="#77B255" d="M36 32c0 2.209-1.791 4-4 4H4c-2.209 0-4-1.791-4-4V4c0-2.209 1.791-4 4-4h28c2
<path fill="#FFF" d="M21.529 18.00618.238-8.238c.977-.976.977-2.559 0-3.535-.977-.977-2.559-.977-3.53
</svg>

Alice->Bob : <$foo1>

@enduml
```



23.2 Changing colors

Although sprites are monochrome, it's possible to change their color.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1,scale=3.4,color=orange>
@enduml
```



23.3 Encoding Sprite

To encode sprite, you can use the command line like:

```
java -jar plantuml.jar -encodesprite 16z foo.png
```

where `foo.png` is the image file you want to use (it will be converted to gray automatically).

After `-encodesprite`, you have to specify a format: `4, 8, 16, 4z, 8z` or `16z`.

The number indicates the gray level and the optional `z` is used to enable compression in sprite definition.

23.4 Importing Sprite

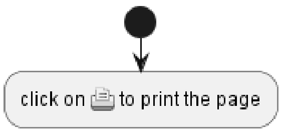
You can also launch the GUI to generate a sprite from an existing image.

Click in the menubar then on `File/Open Sprite Window`.

After copying an image into you clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

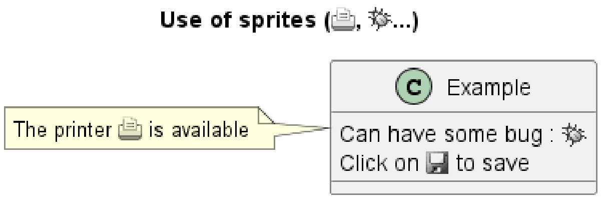
23.5 Examples

```
@startuml
sprite $printer [15x15/8z] NOtH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwvl
start
:click on <$printer> to print the page;
@enduml
```



```
@startuml
 sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEqB1z27aeqCqixa8S4OT7C53cKpsHpaYPDJY_12MHM-BLRyywPhrrl
 sprite $printer [15x15/8z] NOtH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
 sprite $disk {
   444445566677881
   436000000009991
   43600000000ACA1
   53700000001A7A1
   53700000012B8A1
   53800000123B8A1
   63800001233C9A1
   634999AABBC99B1
   744566778899AB1
   7456AAAAA99AAB1
   8566AFC228AABB1
   8567AC8118BBBB1
   867BD4433BBBBB1
   39AAAAABBBBBBC1
}
```

```
 title Use of sprites (<$printer>, <$bug>...)

 class Example {
 Can have some bug : <$bug>
 Click on <$disk> to save
 }

 note left : The printer <$printer> is available

@enduml
```



23.6 StdLib

The PlantUML StdLib includes a number of ready icons in various IT areas such as architecture, cloud services, logos etc. It including AWS, Azure, Kubernetes, C4, product Logos and many others. To explore these libraries:

- Browse the Github folders of PlantUML StdLib
- Browse the source repos of StdLib collections that interest you. Eg if you are interested in logos you can find that it came from gilbarbara-plantuml-sprites, and quickly find its

sprites-list. (The next section shows how to list selected sprites but unfortunately that's in grayscale whereas this custom listing is in color.)

- Study the in-depth Hitchhiker' s Guide to PlantUML, eg sections Standard Library Sprites and PlantUML Stdlib Overview

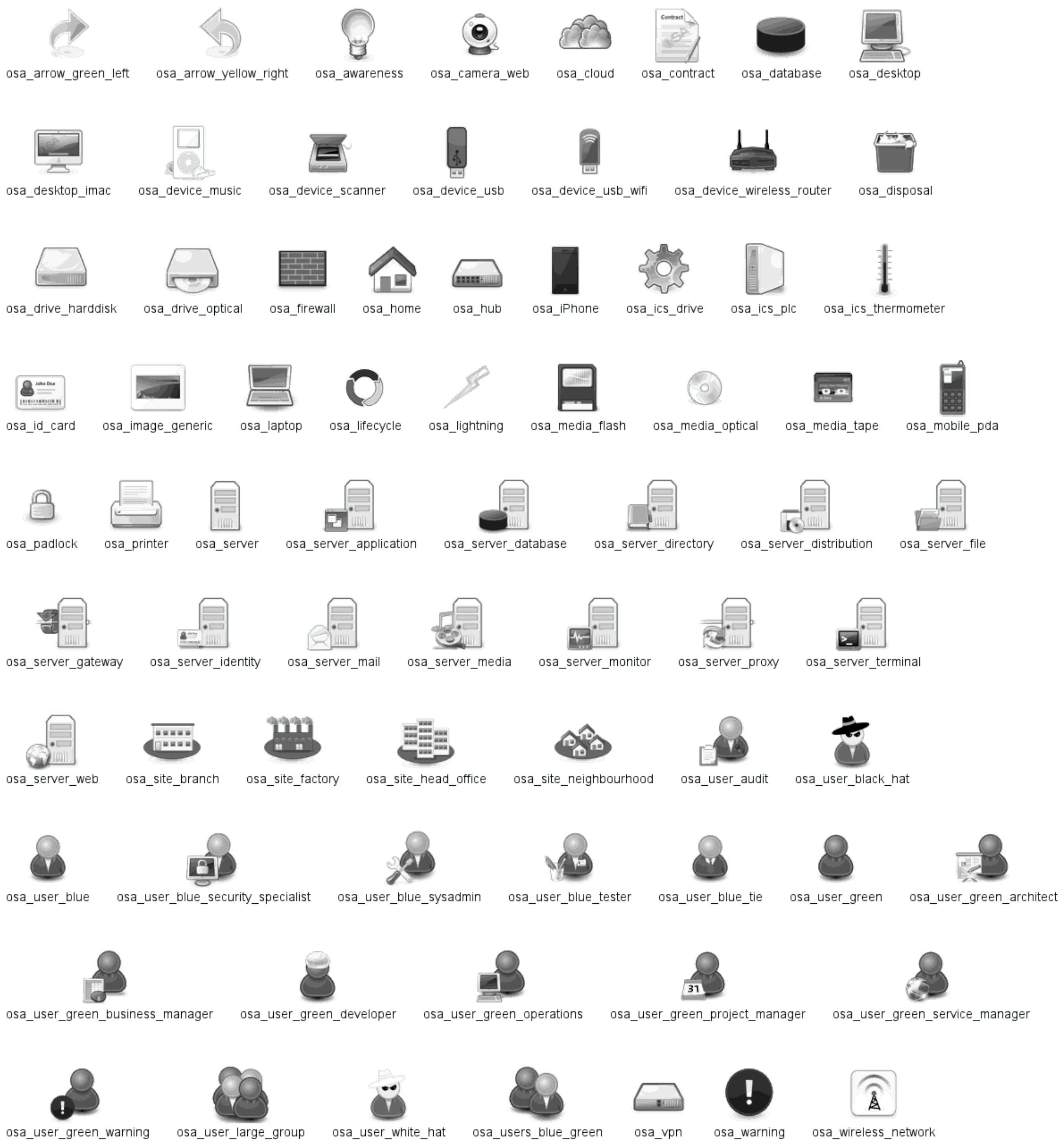
23.7 Listing Sprites

You can use the `listsprites` command to show available sprites:

- Used on its own, it just shows ArchiMate sprites
- If you include some sprite libraries in your diagram, the command shows all these sprites, as explained in View all the icons with listsprites.

(Example from Hitchhikers Guide to PlantUML)

```
@startuml
!define osaPuml https://raw.githubusercontent.com/Crashedmind/PlantUML-opensecurityarchitecture2-icon
!include osaPuml/Common.puml
!include osaPuml/User/all.puml
!include osaPuml/Hardware/all.puml
!include osaPuml/Misc/all.puml
!include osaPuml/Server/all.puml
!include osaPuml/Site/all.puml

listsprites

' From The Hitchhiker' s Guide to PlantUML
@enduml
```



Most collections have files called `all` that allow you to see a whole sub-collection at once. Else you need to find the sprites that interest you and include them one by one. Unfortunately, the version of a collection included in StdLib often does not have such `all` files, so as you see above we include the collection from github, not from StdLib.

All sprites are in grayscale, but most collections define specific macros that include appropriate (vendor-specific) colors.

24 Skinparam 命令

你可以使用 `skinparam` 命令来改变绘图的颜色和字体。

blockquote 原文: You can change colors and font of the drawing using the `skinparam` command. blockquote

示例:

```
skinparam backgroundColor transparent
```

24.1 使用

你可以（以以下方式）使用本命令：

- 在图 (diagram) 的定义中，和其他命令类似
- 在一个包含文件中
- 在一个配置文件中，提供给命令行或者 ANT 任务使用。

blockquote You can use this command : * In the diagram definition, like any other commands, * In an included file, * In a configuration file, provided in the command line or the ANT task. blockquote

24.2 内嵌

为了避免重复 (xxxx 的部分)，允许内嵌（相关的）定义。

因此，如下的定义：

blockquote To avoid repetition, it is possible to nest definition. So the following definition : blockquote

```
skinparam xxxxParam1 value1
skinparam xxxxParam2 value2
skinparam xxxxParam3 value3
skinparam xxxxParam4 value4
```

严格等价于: blockquote is strictly equivalent to: blockquote

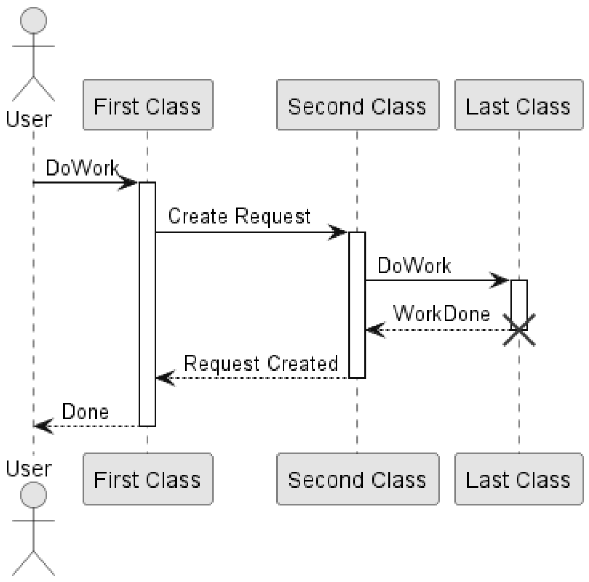
```
skinparam xxxx {
    Param1 value1
    Param2 value2
    Param3 value3
    Param4 value4
}
```

24.3 黑白 (Black and White)

你可以强制使用黑白输出格式，通过 `skinparam monochrome true` 命令。blockquote You can force the use of a black&white output using `skinparam monochrome true` command. blockquote

```
@startuml

skinparam monochrome true

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

A -> B: Create Request
activate B
```

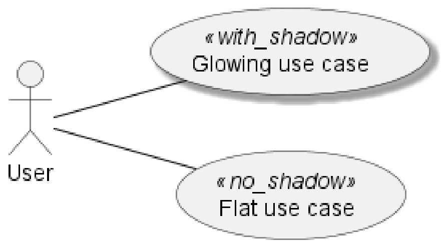
```
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```



24.4 Shadowing

You can disable the shadowing using the `skinparam shadowing false` command.

```
@startuml

left to right direction

skinparam shadowing<<no_shadow>> false
skinparam shadowing<<with_shadow>> true

actor User
(Glowing use case) <<with_shadow>> as guc
(Flat use case) <<no_shadow>> as fuc
User -- guc
User -- fuc

@enduml
```

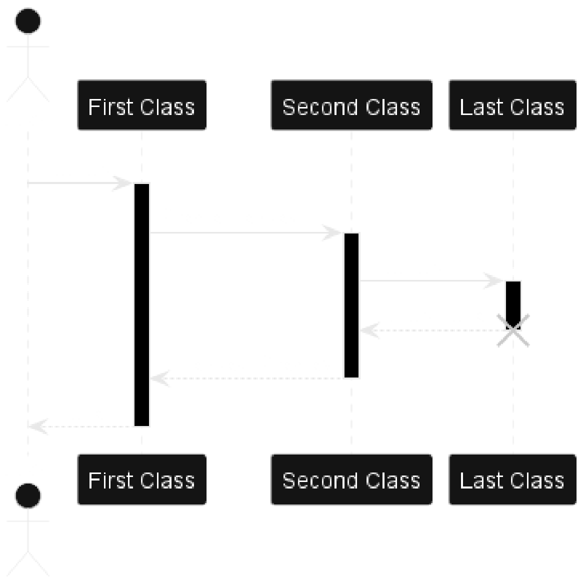


24.5 颜色翻转 (Reverse colors)

可以通过 `skinparam monochrome reverse` 命令，强制使用黑和白的输出，在黑色背景的环境下，尤其适用。

blockquote You can force the use of a black&white output using `skinparam monochrome reverse` command. This can be useful for black background environment. blockquote

```
@startuml

skinparam monochrome reverse

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```

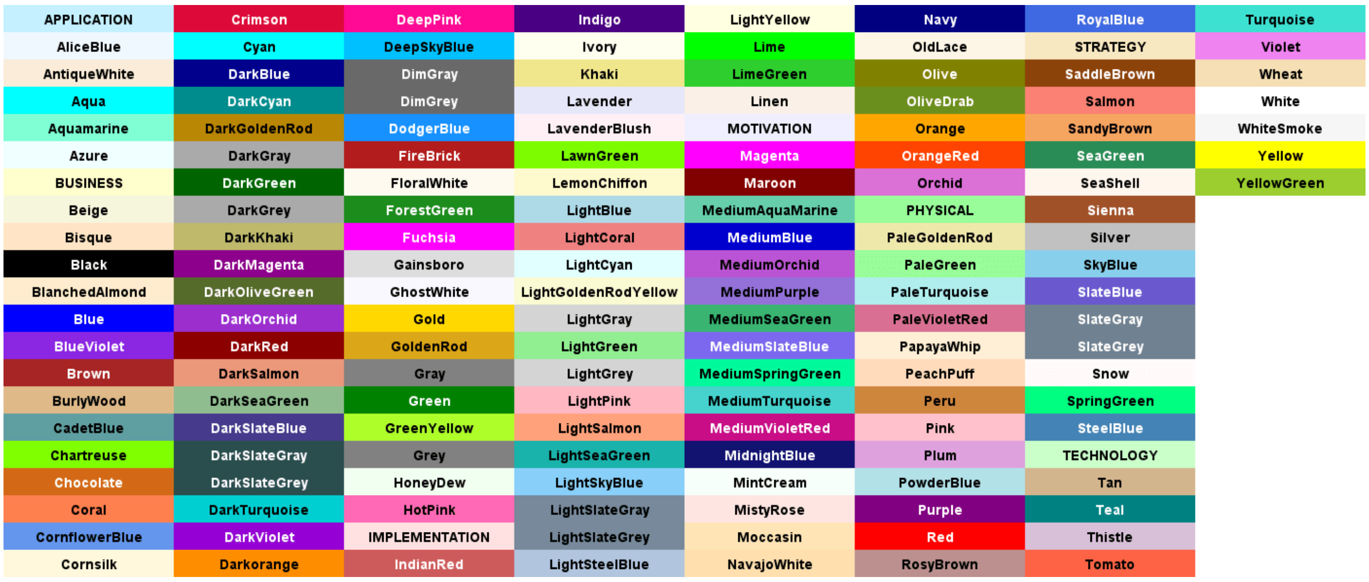


24.6 颜色 (Colors)

你可以使用标准颜色名称或者 RGB 码

blockquote You can use either standard color name or RGB code. blockquote

```
@startuml
colors
@enduml
```



APPLICATION	Crimson	DeepPink	Indigo	LightYellow	Navy	RoyalBlue	Turquoise
AliceBlue	Cyan	DeepSkyBlue	Ivory	Lime	OldLace	STRATEGY	Violet
AntiqueWhite	DarkBlue	DimGray	Khaki	LimeGreen	Olive	SaddleBrown	Wheat
Aqua	DarkCyan	DimGrey	Lavender	Linen	OliveDrab	Salmon	White
Aquamarine	DarkGoldenRod	DodgerBlue	LavenderBlush	MOTIVATION	Orange	SandyBrown	WhiteSmoke
Azure	DarkGray	FireBrick	LawnGreen	Magenta	OrangeRed	SeaGreen	Yellow
BUSINESS	DarkGreen	FloralWhite	LemonChiffon	Maroon	Orchid	SeaShell	YellowGreen
Beige	DarkGrey	ForestGreen	LightBlue	MediumAquaMarine	PHYSICAL	Sienna	
Bisque	DarkKhaki	Fuchsia	LightCoral	MediumBlue	PaleGoldenRod	Silver	
Black	DarkMagenta	Gainsboro	LightCyan	MediumOrchid	PaleGreen	SkyBlue	
BlanchedAlmond	DarkOliveGreen	GhostWhite	LightGoldenRodYellow	MediumPurple	PaleTurquoise	SlateBlue	
Blue	DarkOrchid	Gold	LightGray	MediumSeaGreen	PaleVioletRed	SlateGray	
BlueViolet	DarkRed	GoldenRod	LightGreen	MediumSlateBlue	PapayaWhip	SlateGrey	
Brown	DarkSalmon	Gray	LightGrey	MediumSpringGreen	PeachPuff	Snow	
BurlyWood	DarkSeaGreen	Green	LightPink	MediumTurquoise	Peru	SpringGreen	
CadetBlue	DarkSlateBlue	GreenYellow	LightSalmon	MediumVioletRed	Pink	SteelBlue	
Chartreuse	DarkSlateGray	Grey	LightSeaGreen	MidnightBlue	Plum	TECHNOLOGY	
Chocolate	DarkSlateGrey	HoneyDew	LightSkyBlue	MintCream	PowderBlue	Tan	
Coral	DarkTurquoise	HotPink	LightSlateGray	MistyRose	Purple	Teal	
CornflowerBlue	DarkViolet	IMPLEMENTATION	LightSlateGrey	Moccasin	Red	Thistle	
Cornsilk	Darkorange	IndianRed	LightSteelBlue	NavajoWhite	RosyBrown	Tomato	

`transparent` 只能用于图片背景

blockquote `transparent` can only be used for background of the image. blockquote

24.7 字体颜色、名称、大小 (Font color, name and size)

可以通过使用 `xxxFontColor`, `xxxFontSize` , `xxxFontName` 三个参数，来修改绘图中的字体 (颜色、大小、名称)。

blockquote You can change the font for the drawing using `xxxFontColor`, `xxxFontSize` and `xxxFontName` parameters. blockquote

示例:

```
skinparam classFontColor red
skinparam classFontSize 10
skinparam classFontName Aapex
```

也可以使用 `skinparam defaultFontName` 命令, 来修改默认的字体。

blockquote You can also change the default font for all fonts using `skinparam defaultFontName`. blockquote

Example:

```
skinparam defaultFontName Aapex
```

请注意：字体名称高度依赖于操作系统，因此不要过度使用它，当你考虑到可移植性时。`Helvetica` and `Courier` 应该是全平台可用。

blockquote Please note the fontname is highly system dependent, so do not over use it, if you look for portability. `Helvetica` and `Courier` should be available on all system. blockquote

还有更多的参数可用，你可以通过下面的命令打印它们:

```
java -jar plantuml.jar -language
```

blockquote A lot of parameters are available. You can list them using the following command: java -jar plantuml.jar -language blockquote

24.8 文本对齐 (Text Alignment)

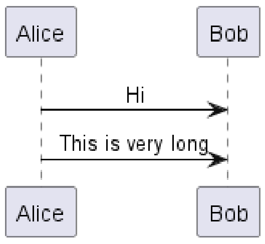
通过 `left`, `right` or `center`, 可以设置文本对齐.

也可以 `sequenceMessageAlign` 指令赋值为 `direction` 或 `reverseDirection` 以便让文本对齐与箭头方向一致。

blockquote Text alignment can be set up to `left`, `right` or `center`. You can also use `direction` or `reverseDirection` values for `sequenceMessageAlign` which align text depending on arrow direction. blockquote

Param name	**Default value**	**Comment**
sequenceMessageAlign	left	用于时序图中的消息 (message)
sequenceReferenceAlign	center	在时序图中用于 `ref over`

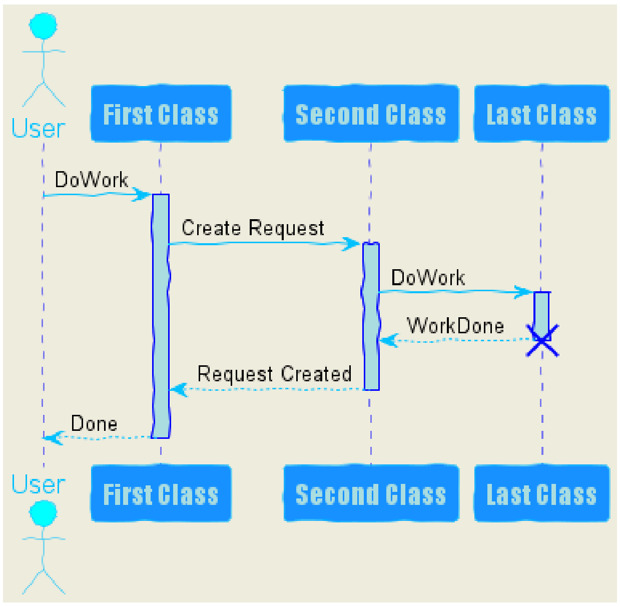
```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Alice -> Bob : This is very long
@enduml
```



24.9 Examples

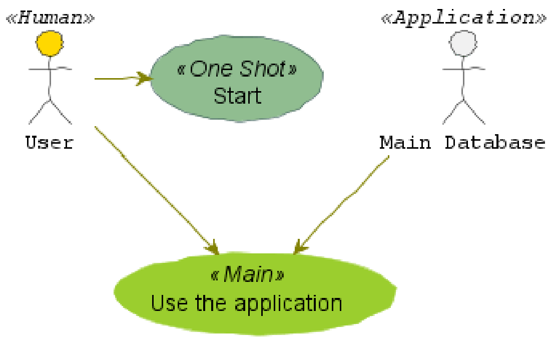
```
@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

skinparam sequence {
ArrowColor DeepSkyBlue
ActorBorderColor DeepSkyBlue
LifeLineBorderColor blue
LifeLineBackgroundColor #A9DCDF

ParticipantBorderColor DeepSkyBlue
ParticipantBackgroundColor DodgerBlue
ParticipantFontName Impact
ParticipantFontSize 17
ParticipantFontColor #A9DCDF

ActorBackgroundColor aqua
ActorFontColor DeepSkyBlue
ActorFontSize 17
ActorFontName Aapex
}

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A
```

```
A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A
@enduml
```

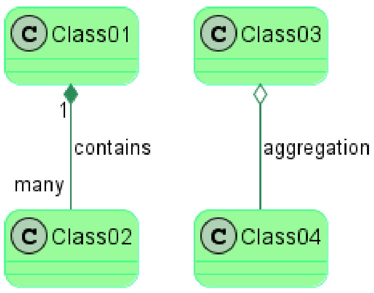


```
@startuml
skinparam handwritten true

skinparam actor {
BorderColor black
FontName Courier
        BackgroundColor<< Human >> Gold
}

skinparam usecase {
BackgroundColor DarkSeaGreen
BorderColor DarkSlateGray

BackgroundColor<< Main >> YellowGreen
BorderColor<< Main >> YellowGreen

ArrowColor Olive
}

User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>
```

```
User -> (Start)
User --> (Use)

MySql --> (Use)
@enduml
```

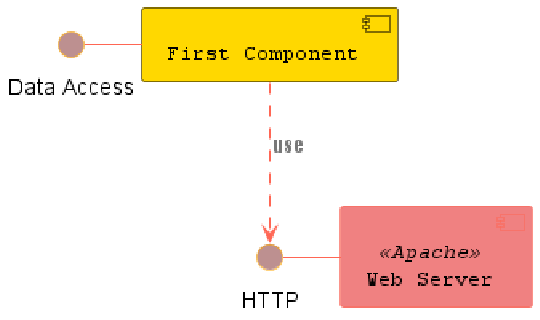


```
@startuml
skinparam roundcorner 20
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation
@enduml
```

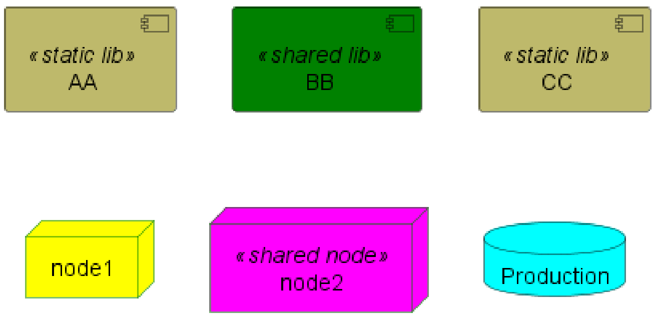


```
@startuml
skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

skinparam component {
  FontSize 13
  BackgroundColor<<Apache>> LightCoral
  BorderColor<<Apache>> #FF6655
  FontName Courier
  BorderColor black
  BackgroundColor gold
  ArrowFontName Impact
  ArrowColor #FF6655
  ArrowFontColor #777777
}
```

```
() "Data Access" as DA
[Web Server] << Apache >>

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server]
@enduml
```



```
@startuml
[AA] <<static lib>>
[BB] <<shared lib>>
[CC] <<static lib>>

node node1
node node2 <<shared node>>
database Production

skinparam component {
    backgroundColor<<static lib>> DarkKhaki
    backgroundColor<<shared lib>> Green
}

skinparam node {
borderColor Green
backgroundColor Yellow
backgroundColor<<shared node>> Magenta
}
skinparam databaseBackgroundColor Aqua
@enduml
```



24.10 所有 skinparam 的参数列表 (List of all skinparam parameters)

blockquote 本文档并不总能保持最新，你可以使用下面命令查看完成的参数列表 blockquote

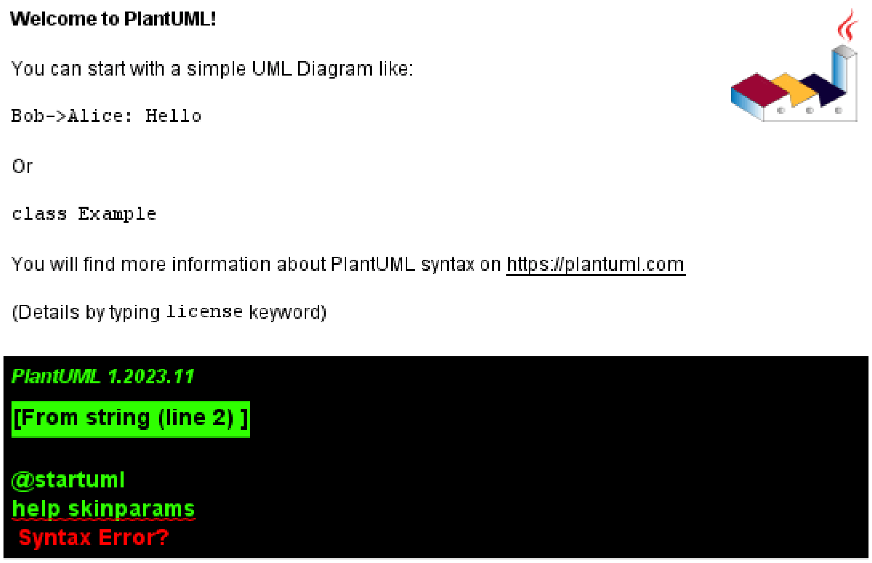
blockquote Since the documentation is not always up to date, you can have the complete list of parameters using this command: blockquote

```
java -jar plantuml.jar -language
```

或者可以使用命令,产生一幅有所有 skinparam 参数的图: blockquote Or you can generate a "diagram" with a list of all the skinparam parameters using: blockquote

结果如下: blockquote That will give you the following result: blockquote

```
@startuml
help skinparams
@enduml
```


Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(Details by typing `license` keyword)

```
PlantUML 1.2023.11
[From string (line 2) ]

@startuml
help skinparams
 Syntax Error?
```

你也可以在 https://plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html 查看''skinparam''的参数. You can also view each skinparam parameters with its results displayed at the page ['All Skin Parameters'](https:*plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html) of ['Ashley's PlantUML Doc'](https:*plantuml-documentation.readthedocs.io/en/latest/index.html#): * https:*plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html.

25 预处理

一些预处理功能包含在 **PlantUML** 中,, 并可用于所有的图。

这些功能与 C 语言的预处理器非常相似，只是特殊字符 `#` 改为感叹号 `!` 。

25.1 定义变量

虽然这不是必须的, 我们强烈建议变量名以 `$` 开头。

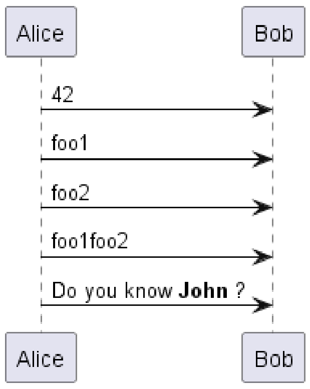
变量有三种数据类型:

- 整型 *(int)*;
- 字符串 *(str)* - 必须被单引号或双引号包围;
- **JSON** *(JSON)* - 必须被大括号包围。

(JSON 变量的定义与使用, 详见 *Preprocessing-JSON* 页面*)*

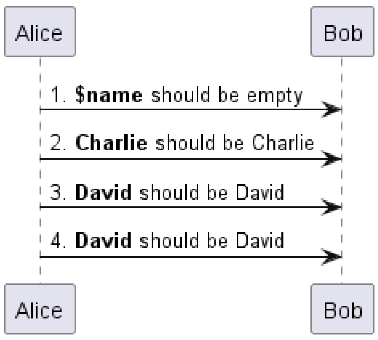
在函数外创建的变量作用域是 **global**, 你可以在任何地方访问他们 (包括函数). 当定义变量的时候你可以使用 `global` 强调这一点。

```
@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
```



你还可以使用以下语法，仅在变量没有被定义时才对变量进行赋值：`!$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
```



25.2 布尔表达式

25.2.1 布尔表示法 [0 是假的]

没有真正的布尔类型，但是 PlantUML 使用这个整数约定：

- 整数 `0` 表示 `false`
- 任何非空数字（如 `1`）或任何字符串（如`"1"`，甚至`"0"`）* 表示 `true` 。

[Ref. QA-9702]

25.2.2 布尔运算和运算符 [&&, ||, ()]

你可以使用布尔表达式，在测试中，使用：

- 小括号 `()`;
- 和运算符 `&&`;
- 或运算符 `||` 。

(见下一个例子，在 if 测试中。)

25.2.3 布尔内置函数 [%false(), %true(), %not(<exp>)===]

为了方便，你可以使用这些布尔内置函数：

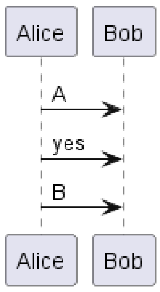
- `%false()`
- `%true()`
- `%not(<exp>)`

[参见 preprocessing#0umqmmdy1n9yk362kjka[内置函数]]

25.3 条件

- 可以在条件里使用表达式.
- 支持 *else* 语法

```
@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
```

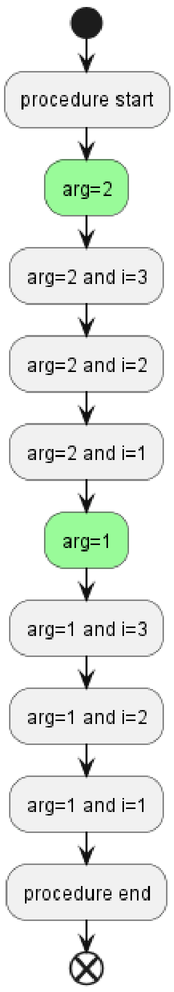


25.4 While 循环 [!while, !endwhile]

你可以使用`!while` 和`!endwhile` 关键字来实现重复循环。

25.4.1 While 循环（在活动图上）。

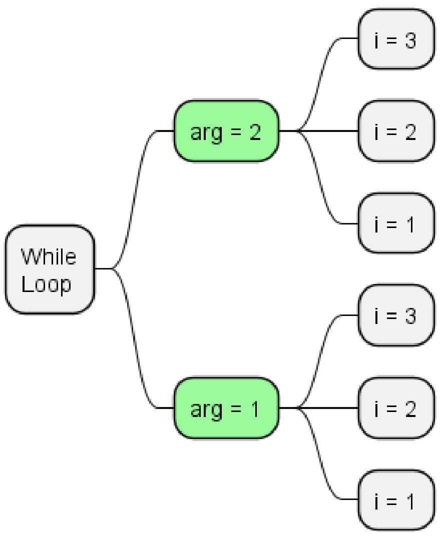
```
@startuml
!procedure $foo($arg)
  :procedure start;
  !while $arg!=0
    !$i=3
    #palegreen:arg=$arg;
    !while $i!=0
      :arg=$arg and i=$i;
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
  :procedure end;
!endprocedure

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



[改编自 QA-10838]

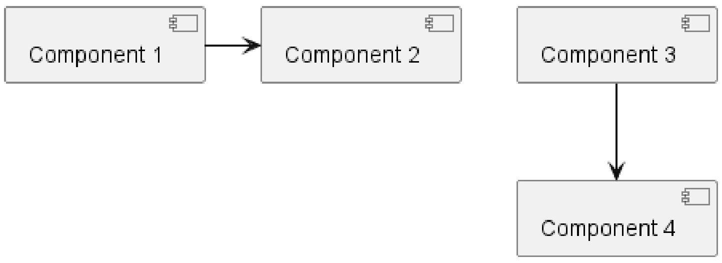
25.4.2 While 循环（在 Mindmap 图上）。

```
@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
```



25.4.3 While 循环 (在组件/部署图上)

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

$foo(4)

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



[参考 QA-14088]

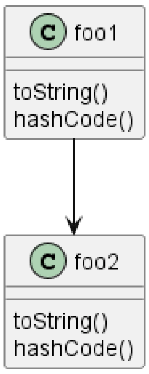
25.5 空函数

- 函数名 必须以 $ 开头
- 参数名 必须以 $ 开头
- 空函数可以调用其他空函数

例:

```
@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
```



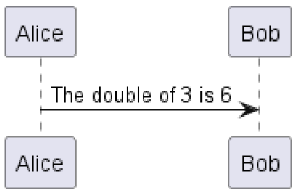
函数里定义的变量作用域为 **local**. 意味着随函数一同销毁.

25.6 返回函数

返回函数不输出任何东西. 它只是定义了一个你可以调用的函数:

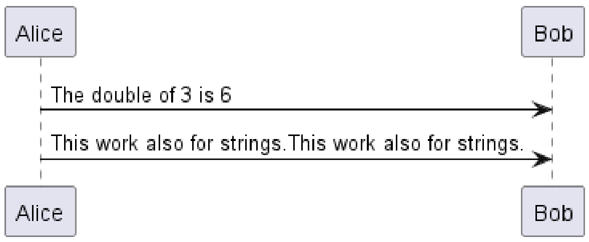
- 直接在变量和图中文本中使用
- 被其他返回函数调用
- 被其他空函数调用
- 函数名 应该以一个 $ 开头
- 参数名 应该以一个 $ 开关

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

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



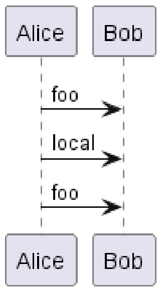
可以简化函数定义为一行:

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

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



像空函数一样, 变量默认为'local' 本地变量 (随函数退出销毁). 并且, 你可以在函数中访问'global' 全局变量. 并且, 如何一个全局变量已存在, 你仍可以使用 `local` 关键字创建一个同名的本地变量.

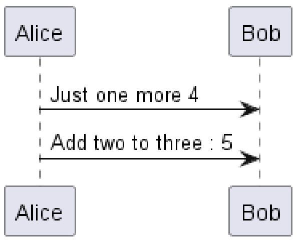
```
@startuml
!procedure $dummy()
!local $ijk = "local"
Alice -> Bob : $ijk
!endprocedure

!global $ijk = "foo"

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



25.7 参数默认值

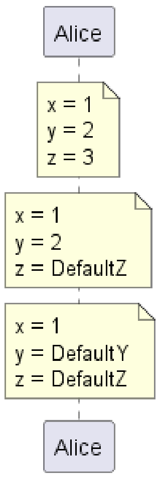
在返回和空函数中, 你可以定义参数默认值.

```
@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
```



只有在尾端的参数列表才可以定义默认值.

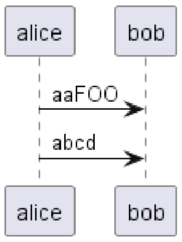
```
@startuml
!procedure defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
  x = $x
  y = $y
  z = $z
end note
!endprocedure

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



25.8 非引号函数

默认情况下, 调用一个函数需要引号. 可以使用 `unquoted` 关键字指明一个函数的参数不需要使用引号.

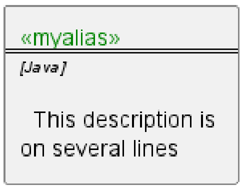
```
@startuml
!unquoted function id($text1, $text2="FOO") return $text1 + $text2

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



25.9 关键字参数

和 Python 一样，你可以使用关键字参数：

```
@startuml

!unquoted procedure $element($alias, $description="", $label="", $technology="", $size=12, $colour="g
rectangle $alias as "
<color:$colour><<$alias>></color>
==$label==
//<size:$size>[$technology]</size>//

  $description"
!endprocedure

$element(myalias, "This description is %newline()on several lines", $size=10, $technology="Java")
@enduml
```

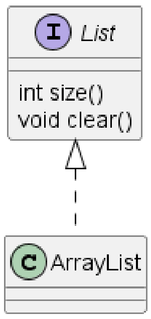


25.10 Including files or URL [!include, !include_many, !include_once]

Use the `!include` directive to include file in your diagram. Using URL, you can also include file from Internet/Intranet. Protected Internet resources can also be accessed, this is described in URL authentication.

Imagine you have the very same class that appears in many diagrams. Instead of duplicating the description of this class, you can define a file that contains the description.

```
@startuml

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



File List.iuml

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

The file `List.iuml` can be included in many diagrams, and any modification in this file will change all diagrams that include it.

You can also put several `@startuml/@enduml` text block in an included file and then specify which block you want to include adding `!0` where `0` is the block number. The `!0` notation denotes the first diagram.

For example, if you use `!include foo.txt!1`, the second `@startuml/@enduml` block within `foo.txt` will be included.

You can also put an id to some `@startuml/@enduml` text block in an included file using `@startuml(id=MY_OWN_ID)` syntax and then include the block adding `!MY_OWN_ID` when including the file, so using something like `!include foo.txt!MY_OWN_ID`.

By default, a file can only be included once. You can use `!include_many` instead of `!include` if you want to include some file several times. Note that there is also a `!include_once` directive that raises an error if a file is included several times.

25.11 Including Subpart [!startsub, !endsub, !includesub]

You can also use `!startsub NAME` and `!endsub` to indicate sections of text to include from other files using `!includesub`. For example:

file1.puml:

```
@startuml

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

file1.puml would be rendered exactly as if it were:

```
@startuml

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

However, this would also allow you to have another file2.puml like this:

file2.puml

```
@startuml

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

This file would be rendered exactly as if:

```
@startuml

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

25.12 Builtin functions [%]

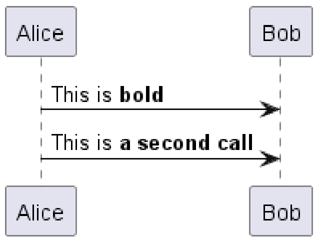
Some functions are defined by default. Their name starts by `%`

Name	Description
`%chr`	Return a character from a give Unicode value
`%darken`	Return a darken color of a given color with some ratio
`%date`	Retrieve current date. You can provide an optional format for the date
	You can provide another optional time (on epoch format)
`%dec2hex`	Return the hexadecimal string (String) of a decimal value (Int)
`%dirpath`	Retrieve current dirpath
`%feature`	Check if some feature is available in the current PlantUML running version
`%false`	Return always `false`
`%file_exists`	Check if a file exists on the local filesystem
`%filename`	Retrieve current filename
`%function_exists`	Check if a function exists
`%get_variable_value`	Retrieve some variable value
`%getenv`	Retrieve environment variable value
`%hex2dec`	Return the decimal value (Int) of a hexadecimal string (String)
`%hsl_color`	Return the RGBa color from a HSL color `%hsl_color(h, s, l)` or `%hsl_color(h, s, l,`
`%intval`	Convert a String to Int
`%is_dark`	Check if a color is a dark one
`%is_light`	Check if a color is a light one
`%lighten`	Return a lighten color of a given color with some ratio
`%load_json`	Load JSON data from local file or external URL
`%lower`	Return a lowercase string
`%newline`	Return a newline
`%not`	Return the logical negation of an expression
`%now`	Return the current epoch time
`%ord`	Return a Unicode value from a given character
`%lighten`	Return a lighten color of a given color with some ratio
`%reverse_color`	Reverse a color using RGB
`%reverse_hsluv_color`	Reverse a color using HSLuv
`%set_variable_value`	Set a global variable
`%size`	Return the size of any string or JSON structure
`%string`	Convert an expression to String
`%strlen`	Calculate the length of a String
`%strpos`	Search a substring in a string
`%substr`	Extract a substring. Takes 2 or 3 arguments
`%true`	Return always `true`
`%upper`	Return an uppercase string
`%variable_exists`	Check if a variable exists
`%version`	Return PlantUML current version

25.13 Logging [!log]

You can use `!log` to add some log output when generating the diagram. This has no impact at all on the diagram itself. However, those logs are printed in the command line's output stream. This could be useful for debug purpose.

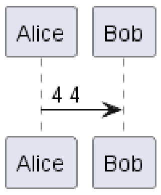
```
@startuml
!function bold($text)
!$result = "<b>"+ $text +"</b>"
!log Calling bold function with $text. The result is $result
!return $result
!endfunction

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



25.14 Memory dump [!dump_memory]

You can use `!dump_memory` to dump the full content of the memory when generating the diagram. An optional string can be put after `!dump_memory`. This has no impact at all on the diagram itself. This could be useful for debug purpose.

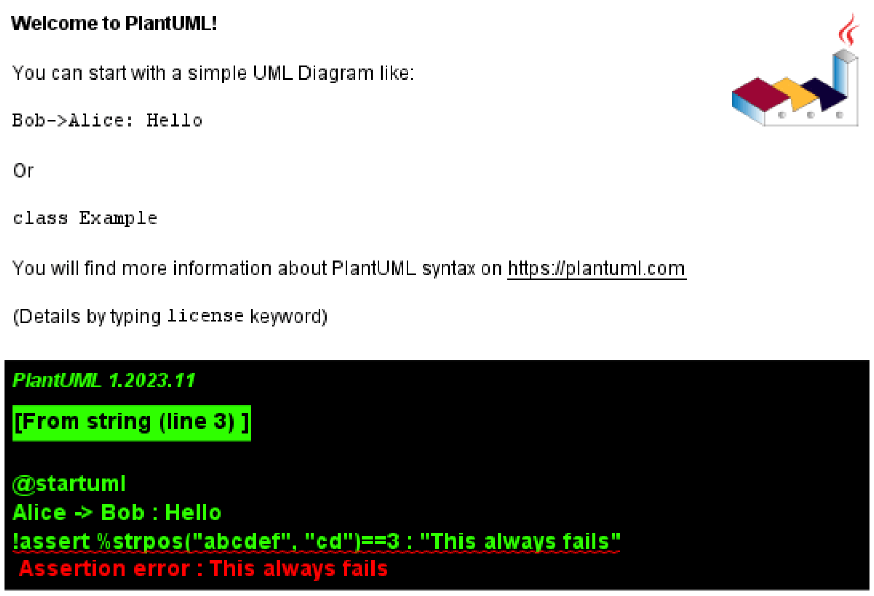
```
@startuml
!function $inc($string)
!$val = %intval($string)
!log value is $val
!dump_memory
!return $val+1
!endfunction

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



25.15 Assertion [!assert]

You can put assertions in your diagram.

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


Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(Details by typing `license` keyword)

```
PlantUML 1.2023.11
[From string (line 3) ]

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

25.16 Building custom library [!import, !include]

It's possible to package a set of included files into a single .zip or .jar archive. This single zip/jar can then be imported into your diagram using `!import` directive.

Once the library has been imported, you can `!include` file from this single zip/jar.

Example:

```
@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

...
```

25.17 Search path

You can specify the java property `plantuml.include.path` in the command line.

For example:

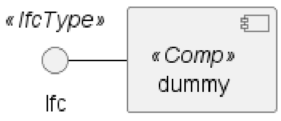
```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

Note the this -D option has to put before the -jar option. -D options after the -jar option will be used to define constants within plantuml preprocessor.

25.18 Argument concatenation [##]

It is possible to append text to a macro argument using the `##` syntax.

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

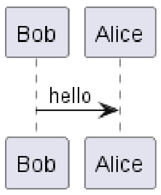


25.19 Dynamic invocation [`%invoke_procedure()`, `%call_user_func()`]

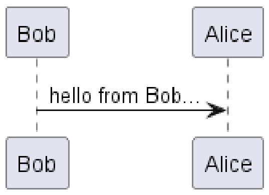
You can dynamically invoke a procedure using the special `%invoke_procedure()` procedure. This procedure takes as first argument the name of the actual procedure to be called. The optional following arguments are copied to the called procedure.

For example, you can have:

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

!$wrapper = "$go"

%invoke_procedure($wrapper)
@enduml
```

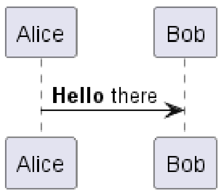


```
@startuml
!procedure $go($txt)
  Bob -> Alice : $txt
!endprocedure

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



For return functions, you can use the corresponding special function `%call_user_func()` :

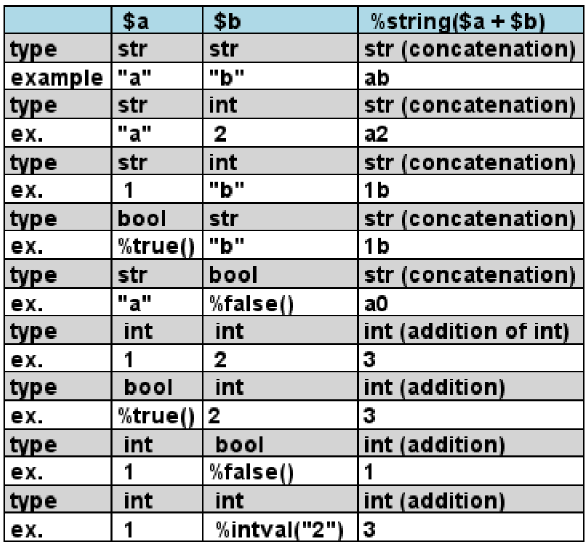
```
@startuml
!function bold($text)
!return "<b>"+ $text +"</b>"
!endfunction

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



25.20 Evaluation of addition depending of data types [+]

Evaluation of `$a + $b` depending of type of `$a` or `$b`

```
@startuml
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	$b	%string($a + $b)
type	str	str	str (concatenation)
example	"a"	"b"	ab
type	str	int	str (concatenation)
ex.	"a"	2	a2
type	str	int	str (concatenation)
ex.	1	"b"	1b
type	bool	str	str (concatenation)
ex.	%true()	"b"	1b
type	str	bool	str (concatenation)
ex.	"a"	%false()	a0
type	int	int	int (addition of int)
ex.	1	2	3
type	bool	int	int (addition)
ex.	%true()	2	3
type	int	bool	int (addition)
ex.	1	%false()	1
type	int	int	int (addition)
ex.	1	%intval("2")	3

25.21 Preprocessing JSON

You can extend the functionality of the current Preprocessing with JSON Preprocessing features:

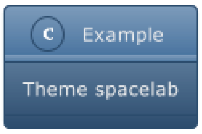
- JSON Variable definition
- Access to JSON data
- Loop over JSON array

(See more details on Preprocessing-JSON page)

25.22 Including theme [!theme]

Use the `!theme` directive to change the default theme of your diagram.

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



You will find more information on the dedicated page.

25.23 迁移说明

目前的预处理是由 legacy preprocessor. 升级而来.

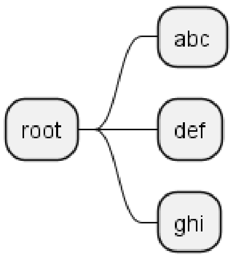
虽然一些历史遗留功能仍被目前的预处理支持, 但是你不应该继续使用 (他们不久将会被移除).

- You should not use `!define` and `!definelong` anymore. Use `!function`, `!procedure` or variable definition instead. `!define` should be replaced by return `!function` and `!definelong` should be replaced by `!procedure`.
- 你不应该再使用 `!define` 和 `!definelong`. 使用 `!function` 和定义变量替换他们. `!define` 替换为返回函数而 `!definelong` 应该替换为 void function.
- `!include` 现在允许多包含: 不应该再使用 `!include_many`
- `!include` 现在可以授受 URL, 所以不再需要 `!includeurl`
- 一些特性 (比如 `%date%`) 替换为内建函数 (例如 `%date()`)
- 当不带参数调用历史遗留`!definelong` 宏的时候, 你必须使用括号. 必须使用 `my_own_definelong()` 因为 `my_own_definelong` 不带括号的形式不被新的预处理语法解析.

如果你有什么疑问请联系我们.

25.24 %Splitstr builtin function

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

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



[Ref. QA-15374]

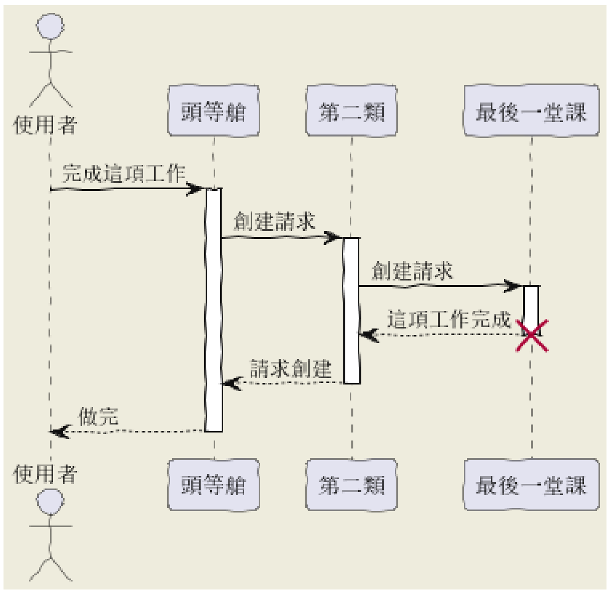
26 Unicode

The PlantUML language use *letters* to define actor, usecase and so on.

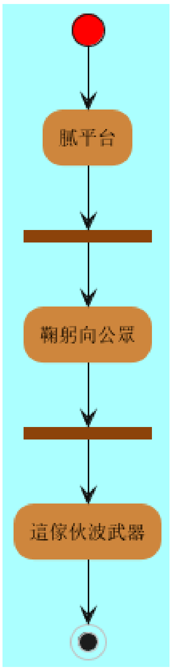
But *letters* are not only A-Z latin characters, it could be *any kind of letter from any language.*

26.1 Examples

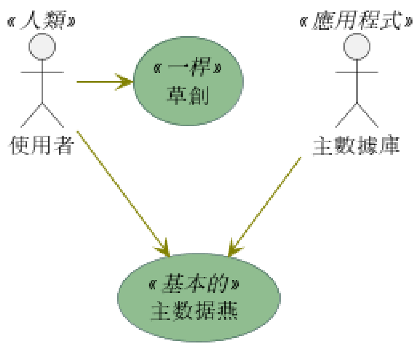
```
@startuml
skinparam handwritten true
skinparam backgroundColor #EEEBDC

actor 使用者
participant "頭等艙" as A
participant "第二類" as B
participant "最後一堂課" as 別的東西

使用者 -> A: 完成這項工作
activate A

A -> B: 創建請求
activate B

B -> 別的東西: 創建請求
activate 別的東西
別的東西 --> B: 這項工作完成
destroy 別的東西

B --> A: 請求創建
deactivate B

A --> 使用者: 做完
deactivate A
@enduml
```



```
@startuml

(*) --> "膩平台"
--> === S1 ===
--> 鞠躬向公眾
--> === S2 ===
--> 這傢伙波武器
--> (*)

skinparam backgroundColor #AAFFFF
skinparam activityStartColor red
skinparam activityBarColor SaddleBrown
skinparam activityEndColor Silver
skinparam activityBackgroundColor Peru
skinparam activityBorderColor Peru
@enduml
```



```
@startuml

skinparam usecaseBackgroundColor DarkSeaGreen
skinparam usecaseArrowColor Olive
skinparam actorBorderColor black
skinparam usecaseBorderColor DarkSlateGray

使用者 << 人類 >>
"主數據庫" as 數據庫 << 應用程式 >>
(草創) << 一桿 >>
"主数据燕" as (贏余) << 基本的 >>

使用者 -> (草創)
使用者 --> (贏余)

數據庫  --> (贏余)
@enduml
```



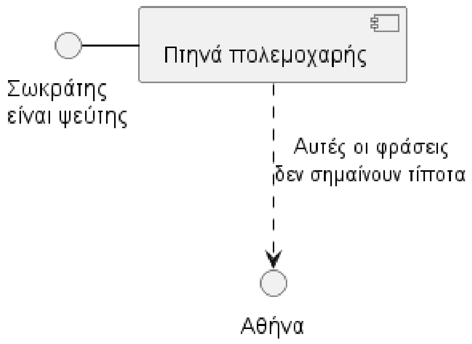
@startuml

() ”Σ   ” as Σ

Σ   - [Π   ]

[Π   ] ..> () Α   : Α

@enduml



26.2 Charset

The default charset used when *reading* the text files containing the UML text description is system dependent.

Normally, it should just be fine, but in some case, you may want to the use another charset. For example, with the command line:

```
java -jar plantuml.jar -charset UTF-8 files.txt
```

Or, with the ant task:

```
<!-- Put images in c:/images directory -->
<target name="main">
<plantuml dir="./src" charset="UTF-8" />
```

Depending of your Java installation, the following charset should be available: `ISO-8859-1`, `UTF-8`, `UTF-16BE`, `UTF-16LE`, `UTF-16`.

26.3 Using Unicode Character on PlantUML

On PlantUML diagram, you can integrate:

- Special characters using `&#XXXX;` or `<U+XXXX>` form;
- Emoji using `<:XXXXX:>` or `<:NameOfEmoji:>`form.

27 PlantUML Standard Library

Welcome to the guide on PlantUML's official **Standard Library** (`stdlib`). Here, we delve into this integral resource that is now included in all official releases of PlantUML, facilitating a richer diagram creation experience. The library borrows its file inclusion convention from the "C standard library", a well-established protocol in the programming world.

27.0.1 Standard Library Overview

The Standard Library is a repository of files and resources, constantly updated to enhance your PlantUML experience. It forms the backbone of PlantUML, offering a range of functionalities and features to explore.

27.0.2 Contribution from the Community

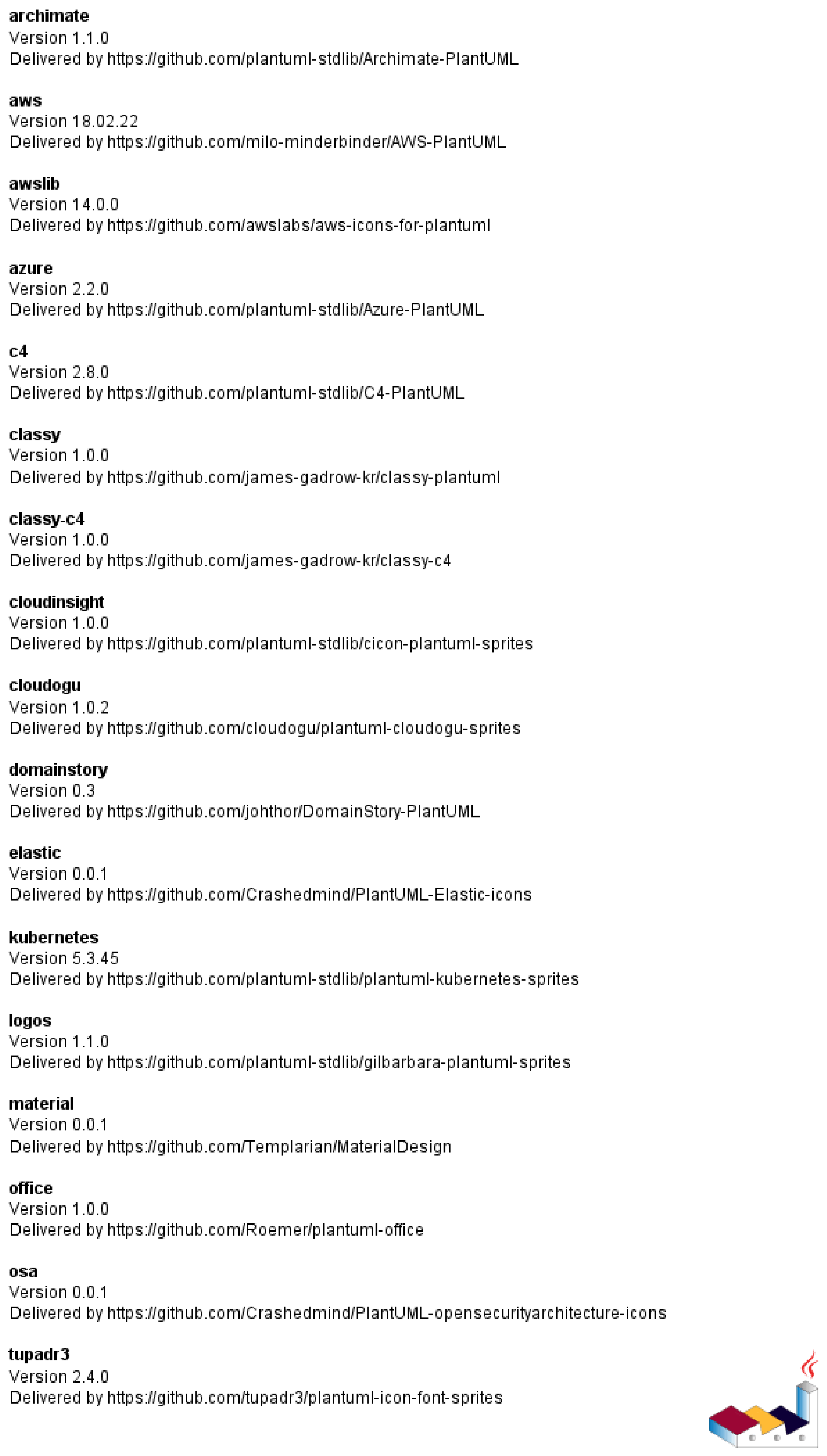
A significant portion of the library's contents are generously provided by third-party contributors. We extend our heartfelt gratitude to them for their invaluable contributions that have played a pivotal role in enriching the library.

We encourage users to delve into the abundant resources the Standard Library offers, to not only enhance their diagram crafting experience but also possibly contribute and be a part of this collaborative endeavor.

27.1 List of Standard Library

You can list standard library folders using the special diagram:

```
@startuml
stdlib
@enduml
```



archimate
Version 1.1.0
Delivered by https://github.com/plantuml-stdlib/Archimate-PlantUML

aws
Version 18.02.22
Delivered by https://github.com/milo-minderbinder/AWS-PlantUML

awslib
Version 14.0.0
Delivered by https://github.com/awslabs/aws-icons-for-plantuml

azure
Version 2.2.0
Delivered by https://github.com/plantuml-stdlib/Azure-PlantUML

c4
Version 2.8.0
Delivered by https://github.com/plantuml-stdlib/C4-PlantUML

classy
Version 1.0.0
Delivered by https://github.com/james-gadrow-kr/classy-plantuml

classy-c4
Version 1.0.0
Delivered by https://github.com/james-gadrow-kr/classy-c4

cloudinsight
Version 1.0.0
Delivered by https://github.com/plantuml-stdlib/cicon-plantuml-sprites

cloudogu
Version 1.0.2
Delivered by https://github.com/cloudogu/plantuml-cloudogu-sprites

domainstory
Version 0.3
Delivered by https://github.com/johthor/DomainStory-PlantUML

elastic
Version 0.0.1
Delivered by https://github.com/Crashedmind/PlantUML-Elastic-icons

kubernetes
Version 5.3.45
Delivered by https://github.com/plantuml-stdlib/plantuml-kubernetes-sprites

logos
Version 1.1.0
Delivered by https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites

material
Version 0.0.1
Delivered by https://github.com/Templarian/MaterialDesign

office
Version 1.0.0
Delivered by https://github.com/Roemer/plantuml-office

osa
Version 0.0.1
Delivered by https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons

tupadr3
Version 2.4.0
Delivered by https://github.com/tupadr3/plantuml-icon-font-sprites

It is also possible to use the command line `java -jar plantuml.jar -stdlib` to display the same list.

Finally, you can extract the full standard library sources using `java -jar plantuml.jar -extractstdlib`. All files will be extracted in the folder `stdlib`.

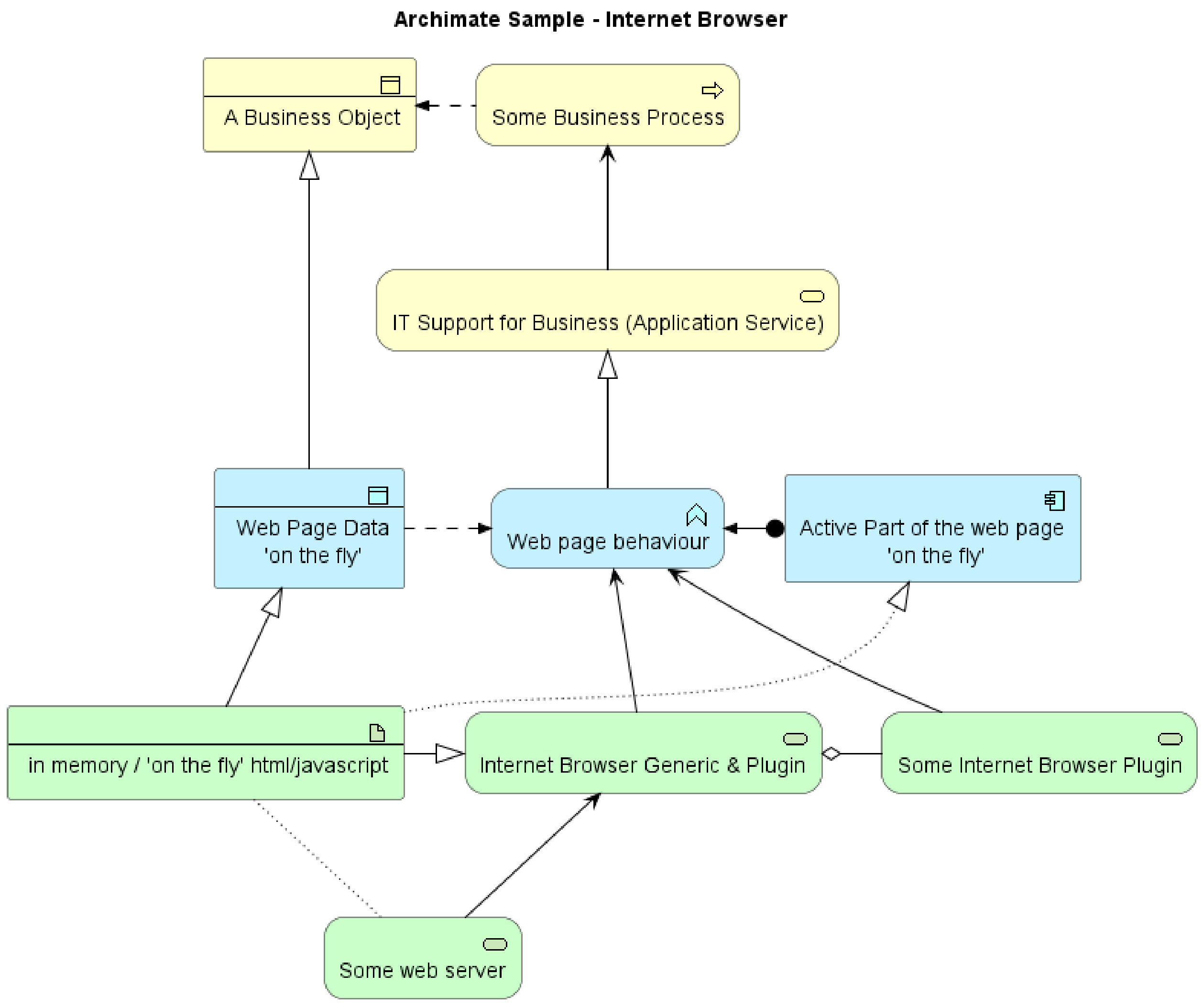
Sources used to build official PlantUML releases are hosted here https://github.com/plantuml/plantuml-stdlib. You can create Pull Request to update or add some library if you find it relevant.

27.2 ArchiMate [archimate]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/archimate
`src`	https://github.com/ebbypeter/Archimate-PlantUML
`orig`	https://en.wikipedia.org/wiki/ArchiMate

This repository contains ArchiMate PlantUML macros and other includes for creating Archimate Diagrams easily and consistantly.

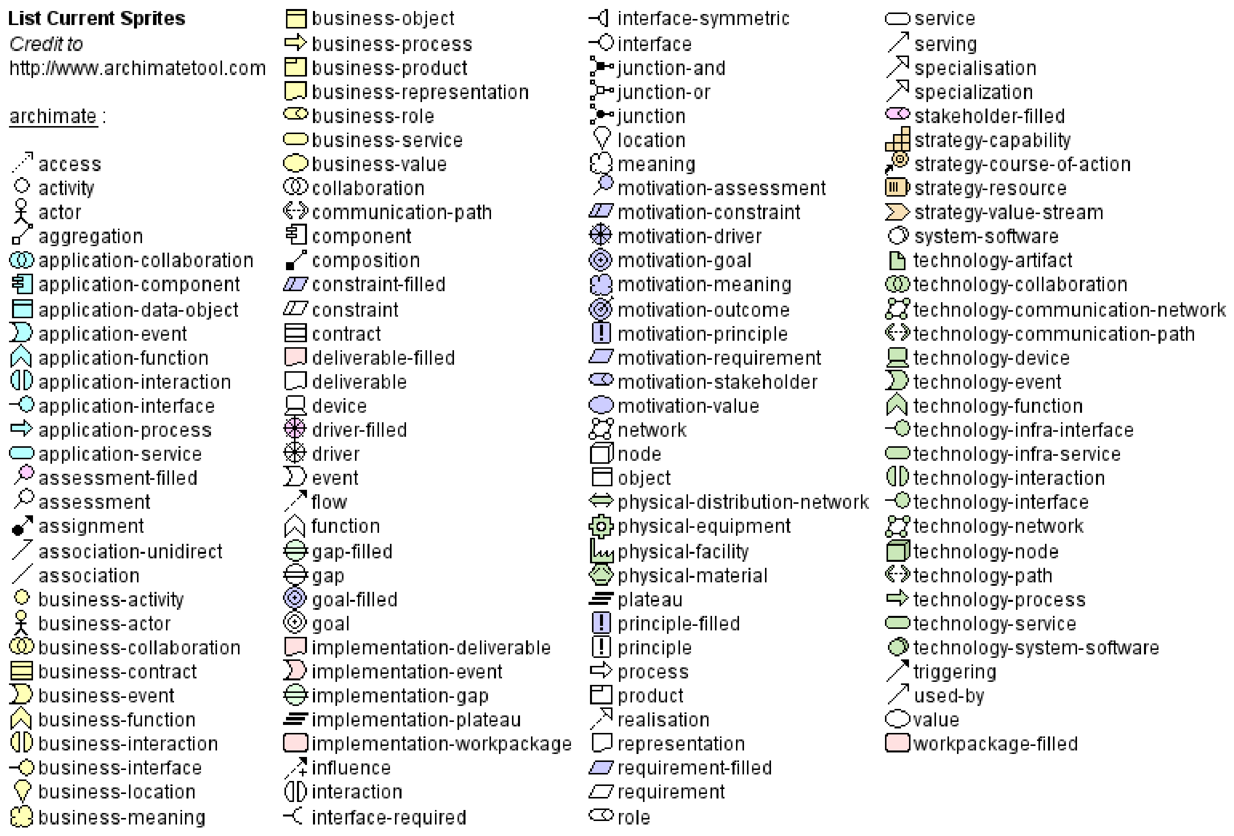
```
@startuml
!include <archimate/Archimate>

title Archimate Sample - Internet Browser

' Elements
Business_Object(businessObject, "A Business Object")
Business_Process(someBusinessProcess,"Some Business Process")
Business_Service(itSupportService, "IT Support for Business (Application Service)")

Application_DataObject(dataObject, "Web Page Data \n 'on the fly'")
Application_Function(webpageBehaviour, "Web page behaviour")
Application_Component(ActivePartWebPage, "Active Part of the web page \n 'on the fly'")

Technology_Artifact(inMemoryItem,"in memory / 'on the fly' html/javascript")
Technology_Service(internetBrowser, "Internet Browser Generic & Plugin")
Technology_Service(internetBrowserPlugin, "Some Internet Browser Plugin")
Technology_Service(webServer, "Some web server")

'Relationships
Rel_Flow_Left(someBusinessProcess, businessObject, "")
Rel_Serving_Up(itSupportService, someBusinessProcess, "")
Rel_Specialization_Up(webpageBehaviour, itSupportService, "")
Rel_Flow_Right(dataObject, webpageBehaviour, "")
Rel_Specialization_Up(dataObject, businessObject, "")
Rel_Assignment_Left(ActivePartWebPage, webpageBehaviour, "")
Rel_Specialization_Up(inMemoryItem, dataObject, "")
Rel_Realization_Up(inMemoryItem, ActivePartWebPage, "")
Rel_Specialization_Right(inMemoryItem,internetBrowser, "")
Rel_Serving_Up(internetBrowser, webpageBehaviour, "")
Rel_Serving_Up(internetBrowserPlugin, webpageBehaviour, "")
Rel_Aggregation_Right(internetBrowser, internetBrowserPlugin, "")
Rel_Access_Up(webServer, inMemoryItem, "")
Rel_Serving_Up(webServer, internetBrowser, "")
@enduml
```


Archimate Sample - Internet Browser
A Business Object
Some Business Process
IT Support for Business (Application Service)
Web Page Data 'on the fly'
Web page behaviour
Active Part of the web page 'on the fly'
in memory / 'on the fly' html/javascript
Internet Browser Generic & Plugin
Some Internet Browser Plugin
Some web server

27.2.1 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```



List Current Sprites
Credit to
http://www.archimatetool.com

archimate :

access
activity
actor
aggregation
application-collaboration
application-component
application-data-object
application-event
application-function
application-interaction
application-interface
application-process
application-service
assessment-filled
assessment
assignment
association-unidirect
association
business-activity
business-actor
business-collaboration
business-contract
business-event
business-function
business-interaction
business-interface
business-location
business-meaning
business-object
business-process
business-product
business-representation
business-role
business-service
business-value
collaboration
communication-path
component
composition
constraint-filled
constraint
contract
deliverable-filled
deliverable
device
driver-filled
driver
event
flow
function
gap-filled
gap
goal-filled
goal
implementation-deliverable
implementation-event
implementation-gap
implementation-plateau
implementation-workpackage
influence
interaction
interface-required
interface-symmetric
interface
junction-and
junction-or
junction
location
meaning
motivation-assessment
motivation-constraint
motivation-driver
motivation-goal
motivation-meaning
motivation-outcome
motivation-principle
motivation-requirement
motivation-stakeholder
motivation-value
network
node
object
physical-distribution-network
physical-equipment
physical-facility
physical-material
plateau
principle-filled
principle
process
product
realisation
representation
requirement-filled
requirement
role
service
serving
specialisation
specialization
stakeholder-filled
strategy-capability
strategy-course-of-action
strategy-resource
strategy-value-stream
system-software
technology-artifact
technology-collaboration
technology-communication-network
technology-communication-path
technology-device
technology-event
technology-function
technology-infra-interface
technology-infra-service
technology-interaction
technology-interface
technology-network
technology-node
technology-path
technology-process
technology-service
technology-system-software
triggering
used-by
value
workpackage-filled

27.3 Amazon Labs AWS Library [awslib]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/awslib
`src`	https://github.com/awslabs/aws-icons-for-plantuml
`orig`	https://aws.amazon.com/en/architecture/icons/

The Amazon Labs AWS library provides PlantUML sprites, macros, and other includes for Amazon Web Services (AWS) services and resources.

Used to create PlantUML diagrams with AWS components. All elements are generated from the official AWS Architecture Icons and when combined with PlantUML and the C4 model, are a great way to communicate your design, deployment, and topology as code.

```
@startuml
!include <awslib/AWSCommon>
!include <awslib/InternetOfThings/IoTRule>
!include <awslib/Analytics/KinesisDataStreams>
!include <awslib/ApplicationIntegration/SimpleQueueService>

left to right direction

agent "Published Event" as event #fff

IoTRule(iotRule, "Action Error Rule", "error if Kinesis fails")
KinesisDataStreams(eventStream, "IoT Events", "2 shards")
SimpleQueueService(errorQueue, "Rule Error Queue", "failed Rule actions")

event --> iotRule : JSON message
iotRule --> eventStream : messages
iotRule --> errorQueue : Failed action message
@enduml
```

27.4 Azure library [azure]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/azure
`src`	https://github.com/RicardoNiepel/Azure-PlantUML/
`orig`	Microsoft Azure

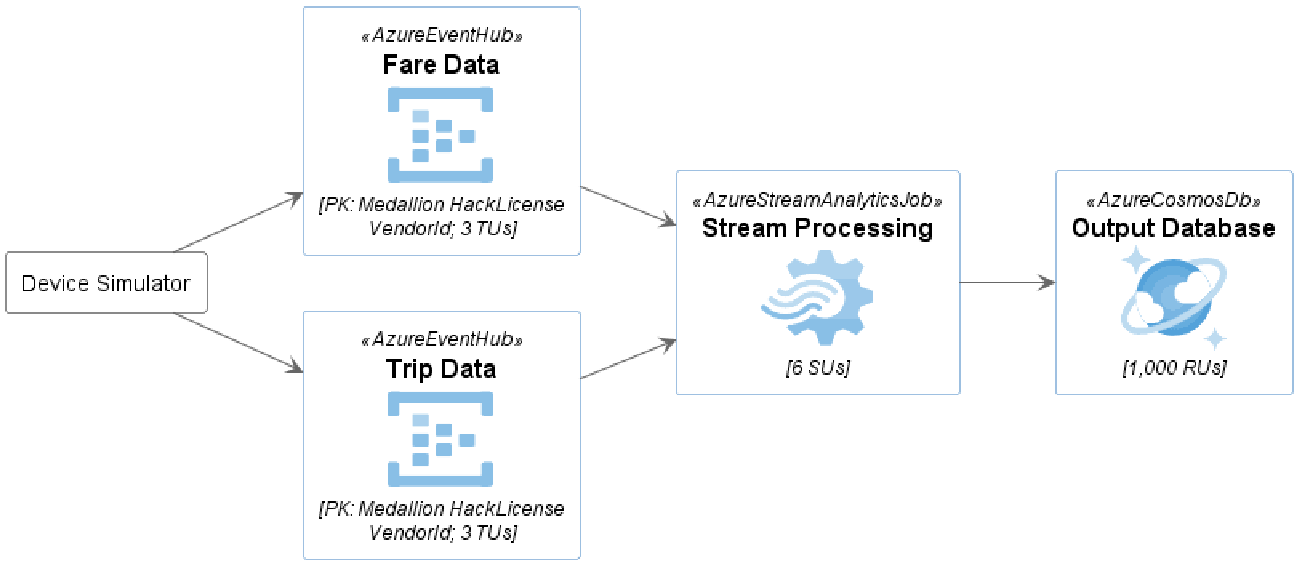
The Azure library consists of Microsoft Azure icons.

Use it by including the file that contains the sprite, eg: `!include <azure/Analytics/AzureEventHub>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `AzureCommon.puml` file, eg: `!include <azure/AzureCommon>`, which contains helper macros defined. With the `AzureCommon.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

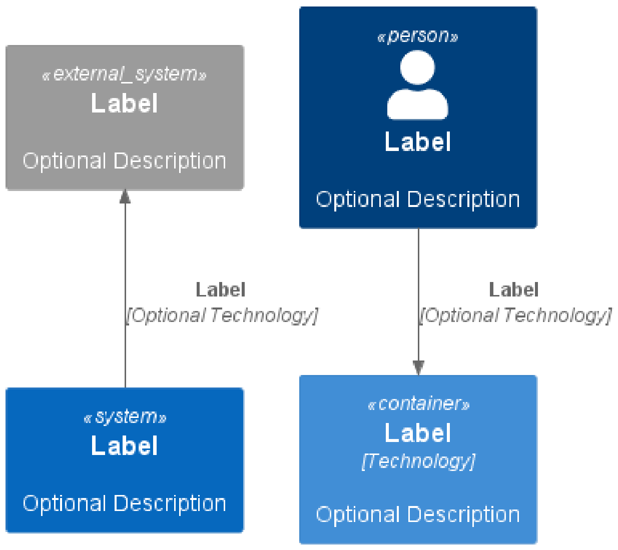
```
@startuml
!include <azure/AzureCommon>
!include <azure/Analytics/AzureEventHub>
!include <azure/Analytics/AzureStreamAnalyticsJob>
!include <azure/Databases/AzureCosmosDb>

left to right direction

agent "Device Simulator" as devices #fff

AzureEventHub(fareDataEventHub, "Fare Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureEventHub(tripDataEventHub, "Trip Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureStreamAnalyticsJob(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")

devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```



27.5 C4 Library [C4]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/C4
src	https://github.com/plantuml-stdlib/C4-PlantUML
orig	https://en.wikipedia.org/wiki/C4_modelhttps://c4model.com

```
@startuml
!include <C4/C4_Container>

Person(personAlias, "Label", "Optional Description")
Container(containerAlias, "Label", "Technology", "Optional Description")
System(systemAlias, "Label", "Optional Description")

System_Ext(extSystemAlias, "Label", "Optional Description")

Rel(personAlias, containerAlias, "Label", "Optional Technology")

Rel_U(systemAlias, extSystemAlias, "Label", "Optional Technology")
@enduml
```



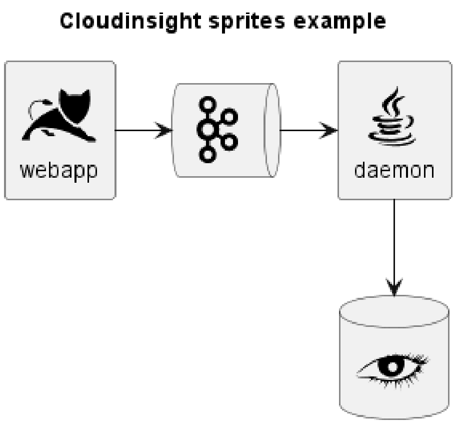
27.6 Cloud Insight [cloudinsight]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudinsight
src	https://github.com/rabelenda/cicon-plantuml-sprites
orig	Cloudinsight icons

This repository contains PlantUML sprites generated from Cloudinsight icons, which can easily be used in PlantUML diagrams for nice visual representation of popular technologies.

```
@startuml
!include <cloudinsight/tomcat>
!include <cloudinsight/kafka>
!include <cloudinsight/java>
!include <cloudinsight/cassandra>

title Cloudinsight sprites example

skinparam monochrome true

rectangle "<$tomcat>\nwebapp" as webapp
```

```
queue "<$kafka>" as kafka
rectangle "<$java>\ndaemon" as daemon
database "<$cassandra>" as cassandra

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml
```

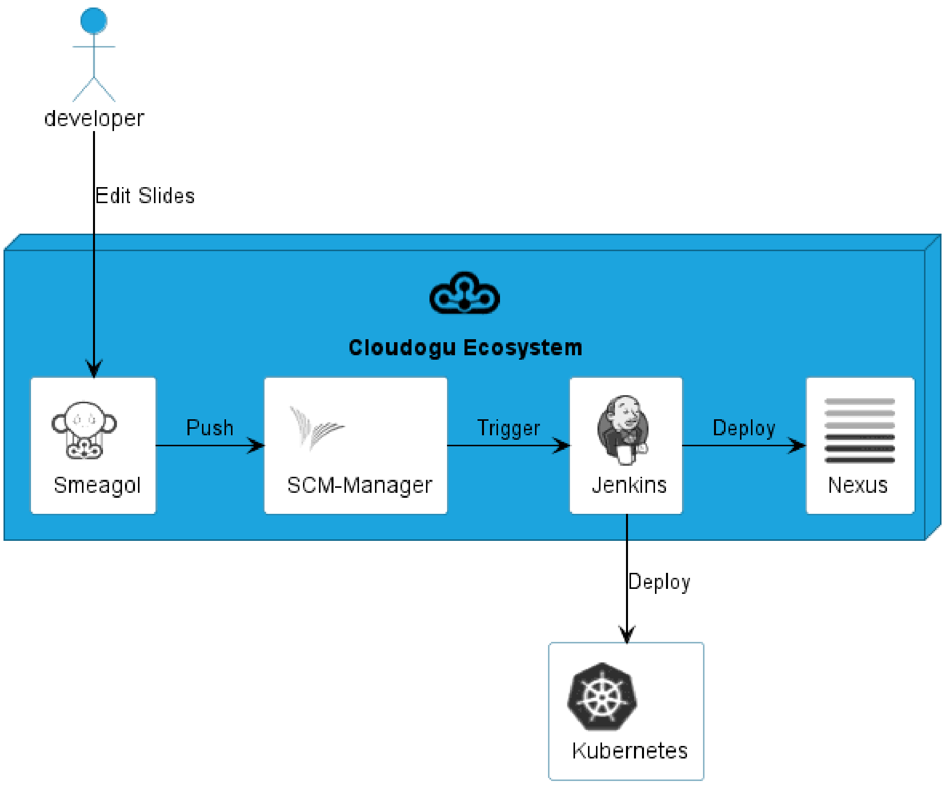


27.7 Cloudogu [cloudogu]

类型	链接
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudogu
src	https://github.com/cloudogu/plantuml-cloudogu-sprites
orig	https://cloudogu.com

Cloudogu 库为 Cloudogu 服务和资源提供了 PlantUML 精灵、宏和其他包括。

```
@startuml
!include <cloudogu/common>
!include <cloudogu/dogus/jenkins>
!include <cloudogu/dogus/cloudogu>
!include <cloudogu/dogus/scm>
!include <cloudogu/dogus/smeagol>
!include <cloudogu/dogus/nexus>
!include <cloudogu/tools/k8s>

node "Cloudogu Ecosystem" <<$cloudogu>> {
DOGU_JENKINS(jenkins, Jenkins) #ffffff
DOGU_SCM(scm, SCM-Manager) #ffffff
DOGU_SMEAGOL(smeagol, Smeagol) #ffffff
DOGU_NEXUS(nexus,Nexus) #ffffff
}

TOOL_K8S(k8s, Kubernetes) #ffffff

actor developer

developer --> smeagol : "Edit Slides"
smeagol -> scm : Push
scm -> jenkins : Trigger
jenkins -> nexus : Deploy
jenkins --> k8s : Deploy
@enduml
```



所有 **cloudogu** 精灵

在 plantuml-cloudogu-sprites 上查看所有可能的 cloudogu 精灵。

27.8 Elastic library [elastic]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/elastic
src	https://github.com/Crashedmind/PlantUML-Elastic-icons
orig	Elastic

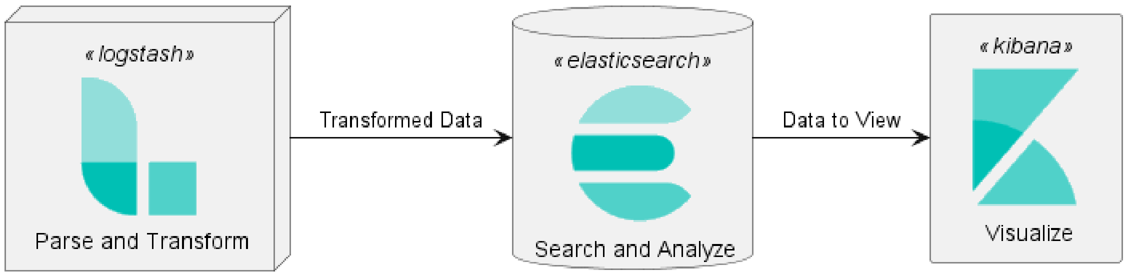
The Elastic library consists of Elastic icons. It is similar in use to the AWS and Azure libraries (it used the same tool to create them).

Use it by including the file that contains the sprite, eg: `!include elastic/elastic_search/elastic_search>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <elastic/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME//OF//SPRITE(parameters...)` macro.

Example of usage:

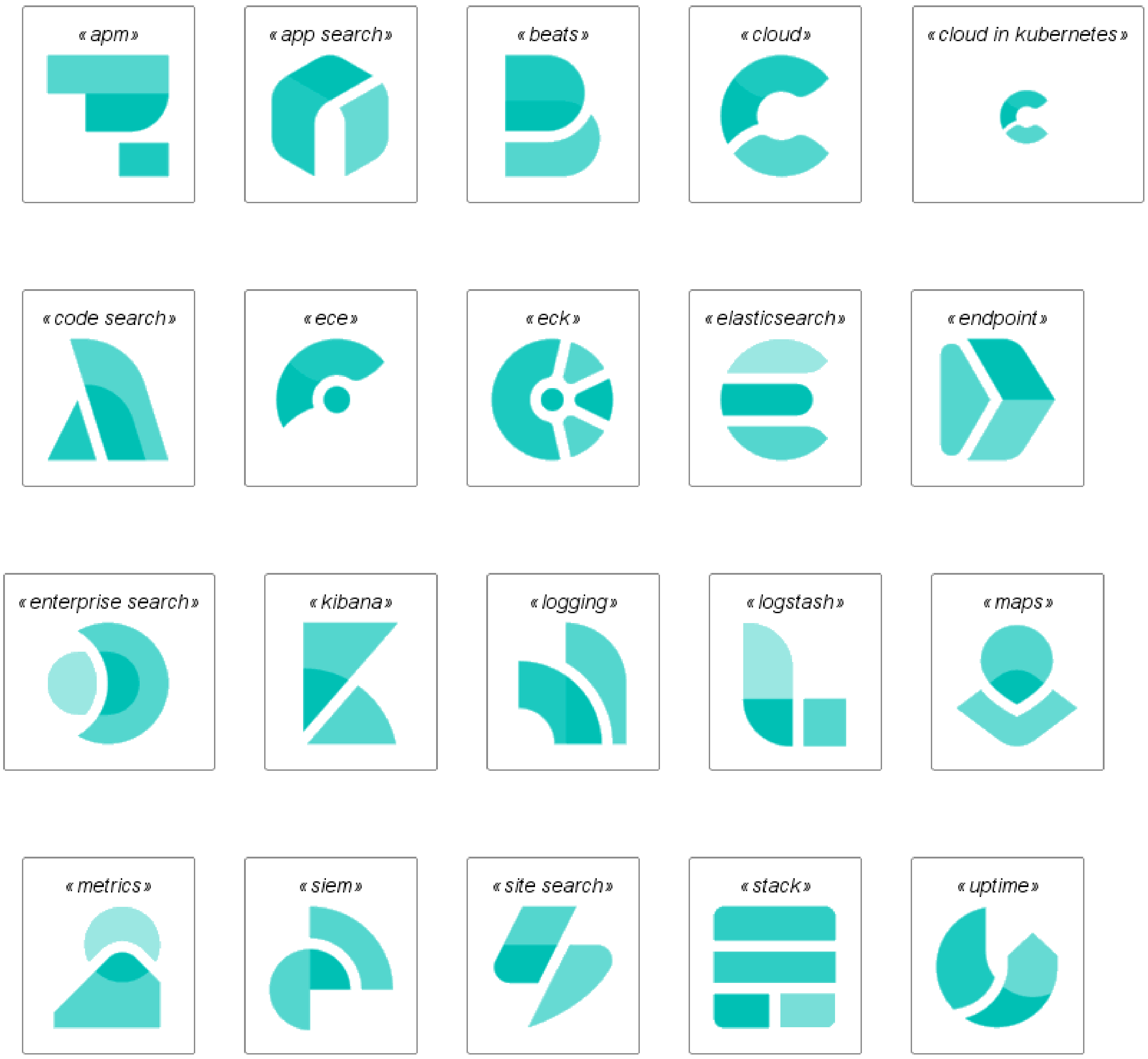
```
@startuml
!include <elastic/common>
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/logstash/logstash>
!include <elastic/kibana/kibana>

ELASTICSEARCH(ElasticSearch, "Search and Analyze",database)
LOGSTASH(Logstash, "Parse and Transform",node)
KIBANA(Kibana, "Visualize",agent)

Logstash -right-> ElasticSearch: Transformed Data
ElasticSearch -right-> Kibana: Data to View
@enduml
```



All Elastic Sprite Set

```
@startuml
'Adapted from https://github.com/Crashedmind/PlantUML-Elastic-icons/blob/master/All.puml

'Elastic stuff here
'================================

!include <elastic/common>
!include <elastic/apm/apm>
!include <elastic/app_search/app_search>
!include <elastic/beats/beats>
!include <elastic/cloud/cloud>
!include <elastic/cloud_in_kubernetes/cloud_in_kubernetes>
!include <elastic/code_search/code_search>
!include <elastic/ece/ece>
!include <elastic/eck/eck>
' Beware of the difference between Crashedmind and plantuml-stdlib version: with '_' usage!
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/endpoint/endpoint>
!include <elastic/enterprise_search/enterprise_search>
!include <elastic/kibana/kibana>
!include <elastic/logging/logging>
!include <elastic/logstash/logstash>
!include <elastic/maps/maps>
!include <elastic/metrics/metrics>
!include <elastic/siem/siem>
!include <elastic/site_search/site_search>
!include <elastic/stack/stack>
!include <elastic/uptime/uptime>

skinparam agentBackgroundColor White

APM(apm)
APP_SEARCH(app_search)
BEATS(beats)
CLOUD(cloud)
CLOUD_IN_KUBERNETES(cloud_in_kubernetes)
CODE_SEARCH(code_search)
ECE(ece)
ECK(eck)
ELASTICSEARCH(elastic_search)
ENDPOINT(endpoint)
ENTERPRISE_SEARCH(enterprise_search)
KIBANA(kibana)
LOGGING(logging)
LOGSTASH(logstash)
MAPS(maps)
METRICS(metrics)
```

```
SIEM(siem)
SITE_SEARCH(site_search)
STACK(stack)
UPTIME(uptime)
@enduml
```



27.9 Google Material Icons [material]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/material
`src`	https://github.com/Templarian/MaterialDesign
`orig`	Material Design Icons

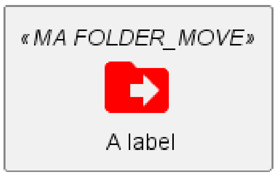
This library consists of a free Material style icons from Google and other artists.

Use it by including the file that contains the sprite, eg: `!include <material/ma_folder_move>`. When imported, you can use the sprite as normally you would, using `<$ma_sprite_name>`. Notice that this library requires an `ma_` prefix on sprites names, this is to avoid clash of names if multiple sprites have the same name on different libraries.

You may also include the `common.puml` file, eg: `!include <material/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `MA_NAME_OF_SPRITE(parameters...)` macro, note again the use of the prefix `MA_`.

Example of usage:

```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label")
@enduml
```

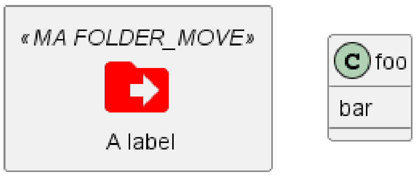


Notes:

When mixing sprites macros with other elements you may get a syntax error if, for example, trying to add a rectangle along with classes. In those cases, add `{` and `}` after the macro to create the empty rectangle.

Example of usage:

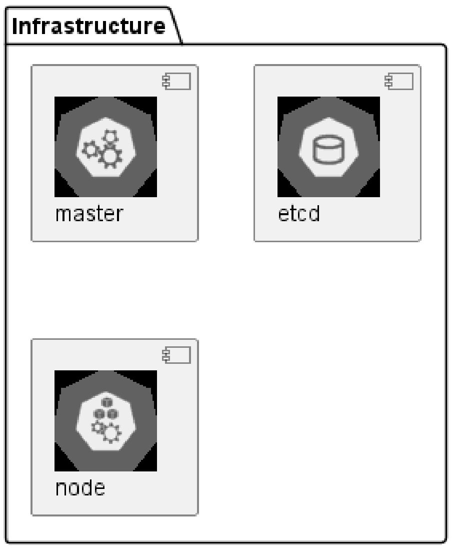
```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label") {
}

class foo {
    bar
}
@enduml
```



27.10 Kubernetes [kubernetes]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/kubernetes
src	https://github.com/michiel/plantuml-kubernetes-sprites
orig	Kubernetes

```
@startuml
!include <kubernetes/k8s-sprites-unlabeled-25pct>
package "Infrastructure" {
  component "<$master>\nmaster" as master
  component "<$etcd>\netcd" as etcd
  component "<$node>\nnode" as node
}
@enduml
```

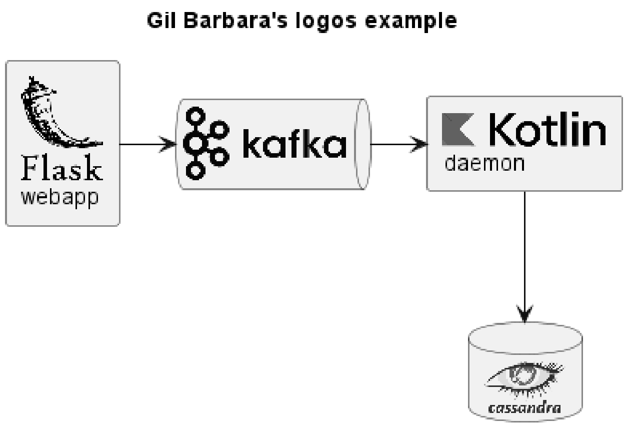


27.11 Logos [logos]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/logos
src	https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites
orig	Gil Barbara's logos

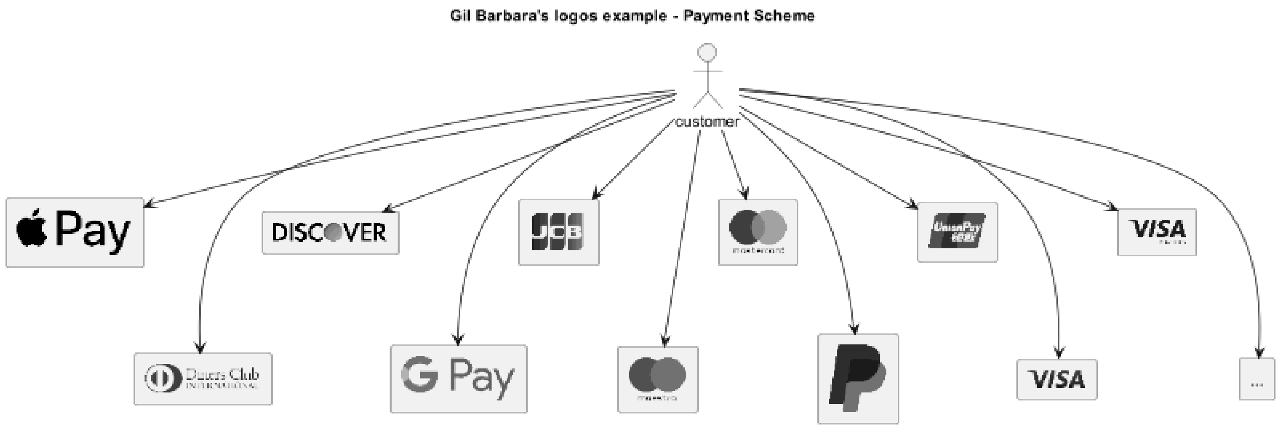
This repository contains PlantUML sprites generated from Gil Barbara's logos, which can easily be used in PlantUML diagrams for nice visual aid.

```
@startuml
!include <logos/flask>
!include <logos/kafka>
!include <logos/kotlin>
!include <logos/cassandra>

title Gil Barbara's logos example

skinparam monochrome true

rectangle "<$flask>\nwebapp" as webapp
queue "<$kafka>" as kafka
rectangle "<$kotlin>\ndaemon" as daemon
database "<$cassandra>" as cassandra

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml
```



```
@startuml
scale 0.7
!include <logos/apple-pay>
!include <logos/dinersclub>
!include <logos/discover>
!include <logos/google-pay>
!include <logos/jcb>
!include <logos/maestro>
!include <logos/mastercard>
!include <logos/paypal>
!include <logos/unionpay>
!include <logos/visaelectron>
!include <logos/visa>
' ...

title Gil Barbara's logos example - **Payment Scheme**

actor customer
rectangle "<$apple-pay>"    as ap
rectangle "<$dinersclub>"   as dc
rectangle "<$discover>"     as d
rectangle "<$google-pay>"   as gp
rectangle "<$jcb>"          as j
rectangle "<$maestro>"      as ma
rectangle "<$mastercard>"   as m
rectangle "<$paypal>"       as p
rectangle "<$unionpay>"     as up
rectangle "<$visa>"         as v
rectangle "<$visaelectron>" as ve
rectangle "..." as etc

customer --> ap
customer ---> dc
customer --> d
customer ---> gp
customer --> j
customer ---> ma
customer --> m
customer ---> p
customer --> up
customer ---> v
customer --> ve
customer ---> etc
```

```
@enduml
```


Gil Barbara's logos example - Payment Scheme
customer
Pay
DISCOVER
JCB
VISA
Diners Club INTERNATIONAL
G Pay
VISA
...

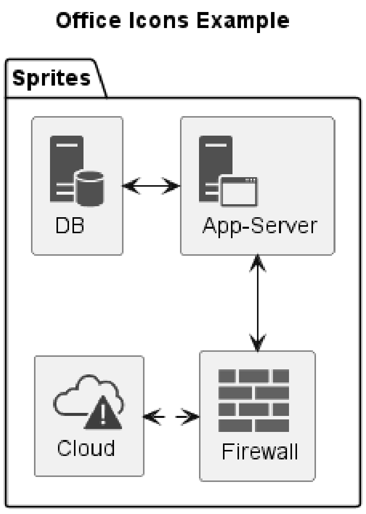
27.12 Office [office]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/office
`src`	https://github.com/Roemer/plantuml-office
`orig`	

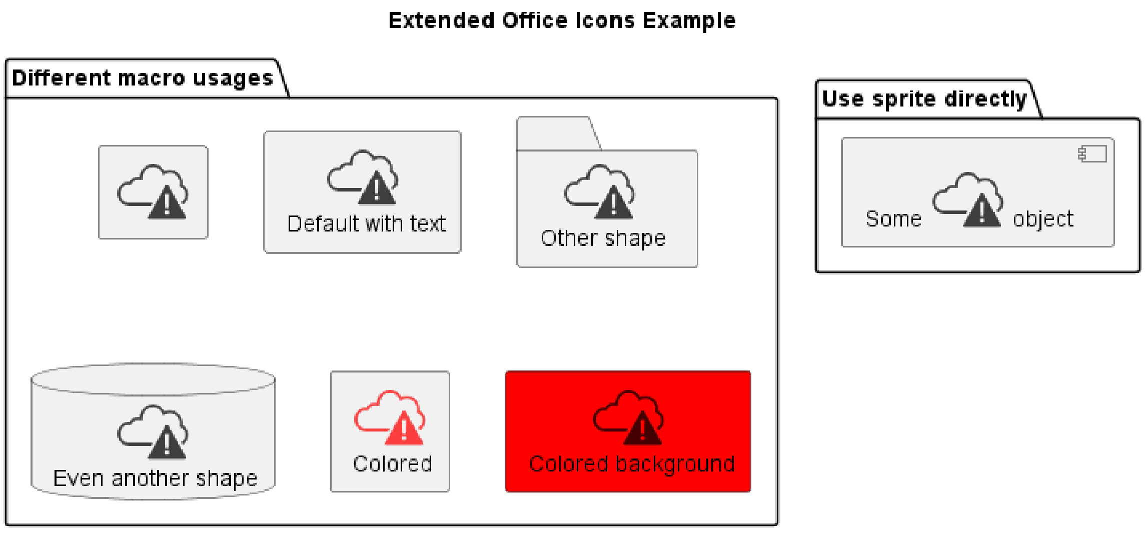
There are sprites (*.puml) and colored png icons available. Be aware that the sprites are all only monochrome even if they have a color in their name (due to automatically generating the files). You can either color the sprites with the macro (see examples below) or directly use the fully colored pngs. See the following examples on how to use the sprites, the pngs and the macros.

Example of usage:

```
@startuml
!include <tupadr3/common>

!include <office/Servers/database_server>
!include <office/Servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

title Office Icons Example

package "Sprites" {
    OFF_DATABASE_SERVER(db,DB)
    OFF_APPLICATION_SERVER(app,App-Server)
    OFF_FIREWALL_ORANGE(fw,Firewall)
    OFF_CLOUD_DISASTER_RED(cloud,Cloud)
    db <-> app
    app <--> fw
    fw <.left.> cloud
}
@enduml
```



```
@startuml
!include <tupadr3/common>

!include <office/servers/database_server>
!include <office/servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

' Used to center the label under the images
skinparam defaultTextAlignment center

title Extended Office Icons Example

package "Use sprite directly" {
    [Some <$cloud_disaster_red> object]
}

package "Different macro usages" {
    OFF_CLOUD_DISASTER_RED(cloud1)
    OFF_CLOUD_DISASTER_RED(cloud2,Default with text)
    OFF_CLOUD_DISASTER_RED(cloud3,Other shape,Folder)
    OFF_CLOUD_DISASTER_RED(cloud4,Even another shape,Database)
    OFF_CLOUD_DISASTER_RED(cloud5,Colored,Rectangle, red)
    OFF_CLOUD_DISASTER_RED(cloud6,Colored background) #red
}
@enduml
```

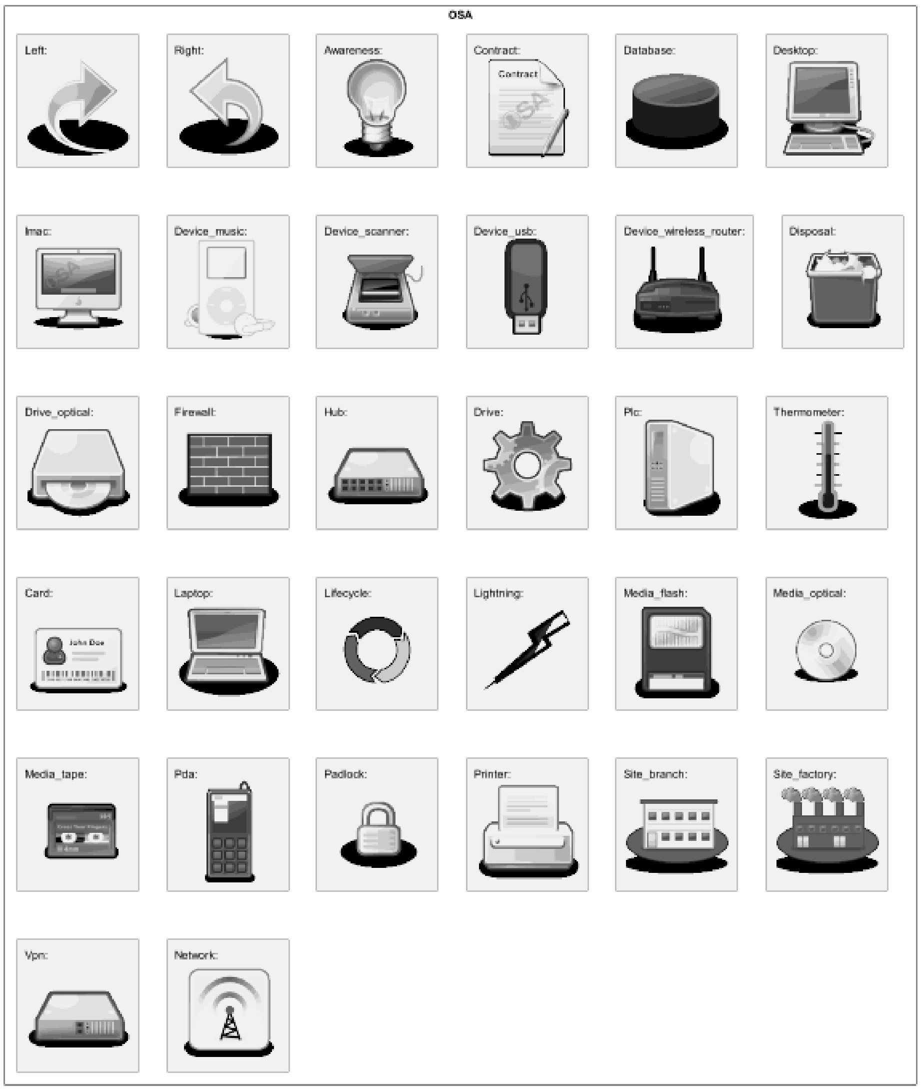


27.13 Open Security Architecture (OSA) [osa]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/osa
src	https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-iconshttps://github.com/Crashedmin
orig	https://www.opensecurityarchitecture.org

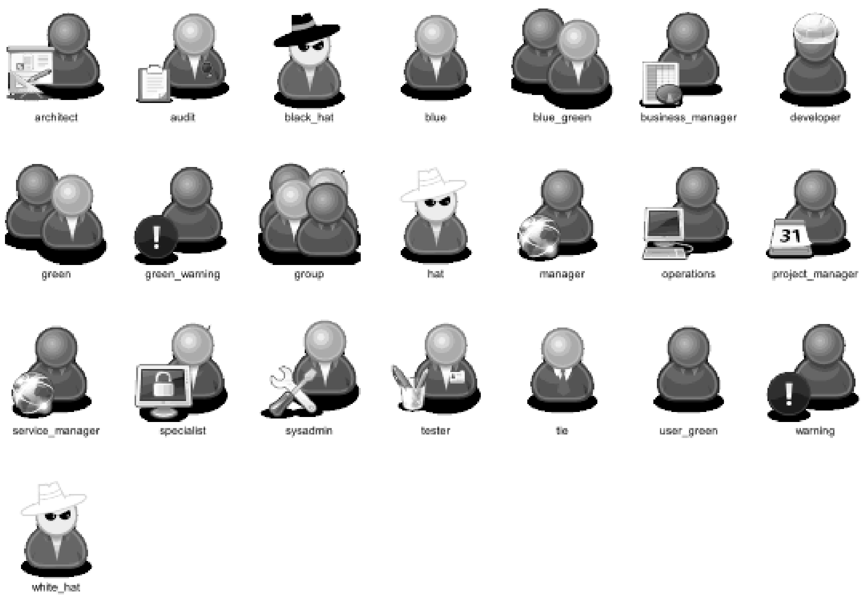
```
@startuml
'Adapted from https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons/blob/master/all
scale .5
!include <osa/arrow/green/left/left>
!include <osa/arrow/yellow/right/right>
!include <osa/awareness/awareness>
!include <osa/contract/contract>
!include <osa/database/database>
!include <osa/desktop/desktop>
!include <osa/desktop/imac/imac>
!include <osa/device_music/device_music>
!include <osa/device_scanner/device_scanner>
!include <osa/device_usb/device_usb>
!include <osa/device_wireless_router/device_wireless_router>
!include <osa/disposal/disposal>
!include <osa/drive_optical/drive_optical>
!include <osa/firewall/firewall>
!include <osa/hub/hub>
!include <osa/ics/drive/drive>
!include <osa/ics/plc/plc>
!include <osa/ics/thermometer/thermometer>
!include <osa/id/card/card>
!include <osa/laptop/laptop>
!include <osa/lifecycle/lifecycle>
!include <osa/lightning/lightning>
!include <osa/media_flash/media_flash>
!include <osa/media_optical/media_optical>
!include <osa/media_tape/media_tape>
!include <osa/mobile/pda/pda>
!include <osa/padlock/padlock>
!include <osa/printer/printer>
!include <osa/site_branch/site_branch>
!include <osa/site_factory/site_factory>
!include <osa/vpn/vpn>
```

```
!include <osa/wireless/network/network>

rectangle "OSA" {
rectangle "Left:\n <$left>"
rectangle "Right:\n <$right>"
rectangle "Awareness:\n <$awareness>"
rectangle "Contract:\n <$contract>"
rectangle "Database:\n <$database>"
rectangle "Desktop:\n <$desktop>"
rectangle "Imac:\n <$imac>"
rectangle "Device_music:\n <$device_music>"
rectangle "Device_scanner:\n <$device_scanner>"
rectangle "Device_usb:\n <$device_usb>"
rectangle "Device_wireless_router:\n <$device_wireless_router>"
rectangle "Disposal:\n <$disposal>"
rectangle "Drive_optical:\n <$drive_optical>"
rectangle "Firewall:\n <$firewall>"
rectangle "Hub:\n <$hub>"
rectangle "Drive:\n <$drive>"
rectangle "Plc:\n <$plc>"
rectangle "Thermometer:\n <$thermometer>"
rectangle "Card:\n <$card>"
rectangle "Laptop:\n <$laptop>"
rectangle "Lifecycle:\n <$lifecycle>"
rectangle "Lightning:\n <$lightning>"
rectangle "Media_flash:\n <$media_flash>"
rectangle "Media_optical:\n <$media_optical>"
rectangle "Media_tape:\n <$media_tape>"
rectangle "Pda:\n <$pda>"
rectangle "Padlock:\n <$padlock>"
rectangle "Printer:\n <$printer>"
rectangle "Site_branch:\n <$site_branch>"
rectangle "Site_factory:\n <$site_factory>"
rectangle "Vpn:\n <$vpn>"
rectangle "Network:\n <$network>"
}
@enduml
```



```
@startuml
scale .5
!include <osa/user/audit/audit>
'beware of 'hat-sprite'
!include <osa/user/black/hat/hat-sprite>
!include <osa/user/blue/blue>
!include <osa/user/blue/security/specialist/specialist>
!include <osa/user/blue/sysadmin/sysadmin>
!include <osa/user/blue/tester/tester>
!include <osa/user/blue/tie/tie>
!include <osa/user/green/architect/architect>
!include <osa/user/green/business/manager/manager>
!include <osa/user/green/developer/developer>
!include <osa/user/green/green>
!include <osa/user/green/operations/operations>
!include <osa/user/green/project/manager/manager>
!include <osa/user/green/service/manager/manager>
!include <osa/user/green/warning/warning>
!include <osa/user/large/group/group>
!include <osa/users/blue/green/green>
!include <osa/user/white/hat/hat>

listsprites
```

```
@enduml
```



27.14 Tupadr3 library [tupadr3]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/tupadr3
`src`	https://github.com/tupadr3/plantuml-icon-font-sprites
`orig`	https://github.com/tupadr3/plantuml-icon-font-sprites#icon-sets

This library contains several libraries of icons (including Devicons and Font Awesome).

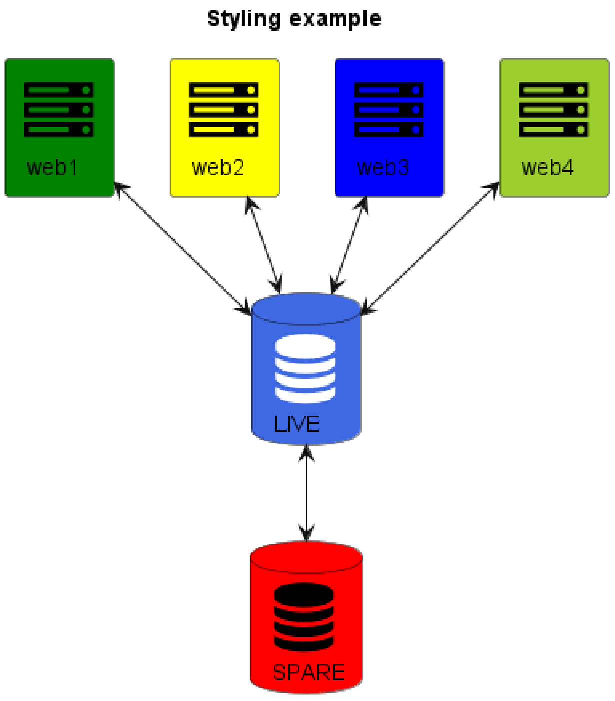
Use it by including the file that contains the sprite, eg: `!include <font-awesome/align_center>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <font-awesome/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

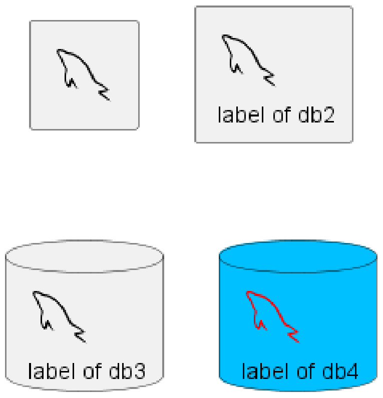
Example of usage:

```
@startuml
!include <tupadr3/common>
!include <tupadr3/font-awesome/server>
!include <tupadr3/font-awesome/database>

title Styling example

FA_SERVER(web1,web1) #Green
FA_SERVER(web2,web2) #Yellow
FA_SERVER(web3,web3) #Blue
FA_SERVER(web4,web4) #YellowGreen

FA_DATABASE(db1,LIVE,database,white) #RoyalBlue
FA_DATABASE(db2,SPARE,database) #Red

db1 <--> db2

web1 <--> db1
web2 <--> db1
web3 <--> db1
web4 <--> db1
```

```
@enduml
```



```
@startuml
!include <tupadr3/common>
!include <tupadr3/devicons/mysql>

DEV_MYSQL(db1)
DEV_MYSQL(db2,label of db2)
DEV_MYSQL(db3,label of db3,database)
DEV_MYSQL(db4,label of db4,database,red) #DeepSkyBlue
@enduml
```



27.15 AWS library [aws]

Type	Link
`stdlib`	https://github.com/plantuml/plantuml-stdlib/tree/master/aws
`src`	https://github.com/milo-minderbinder/AWS-PlantUML
`orig`	https://aws.amazon.com/en/architecture/icons/

Warning: We are thinking about deprecating this library.

So you should probably use `<awslib>` instead (see above).

hr

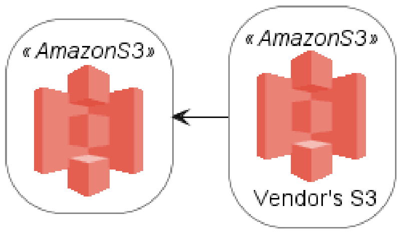
The AWS library consists of Amazon AWS icons, it provides icons of two different sizes (normal and large).

Use it by including the file that contains the sprite, eg: `!include <aws/Storage/AmazonS3/AmazonS3>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <aws/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

```
@startuml
!include <aws/common>
!include <aws/Storage/AmazonS3/AmazonS3>

AMAZONS3(s3_internal)
AMAZONS3(s3_partner,"Vendor's S3")
s3_internal <- s3_partner
@enduml
```



Contents