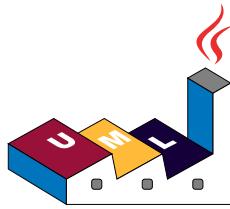


# 使用 PlantUML 绘制的 UML



## PlantUML 语言参考指引

(Version 1.2021.2)

**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 时序图

## 1.1 简单示例

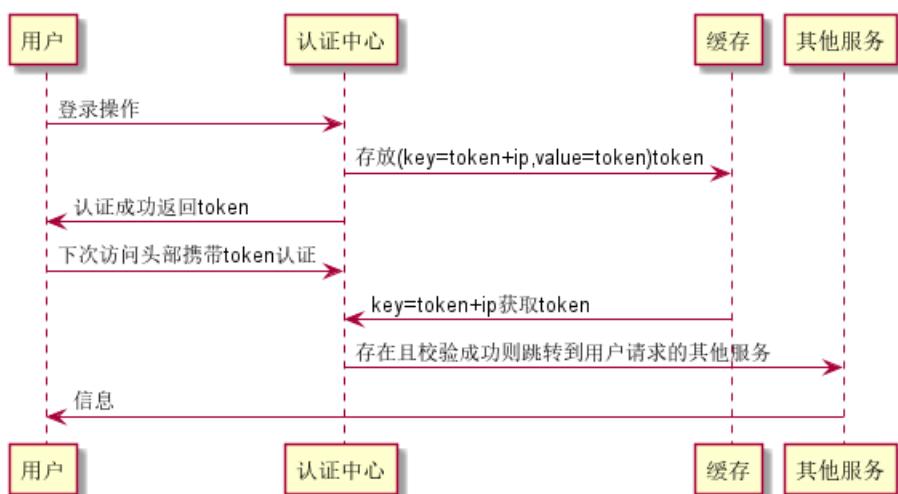
你可以用`->`来绘制参与者之间传递的消息，而不必显式地声明参与者。

你也可以使用`-->`绘制一个虚线箭头。

另外，你还能用`<-`和`<--`，这不影响绘图，但可以提高可读性。注意：仅适用于时序图，对于其它示意图，规则是不同的。

```
@startuml
用户 -> 认证中心: 登录操作
认证中心 -> 缓存: 存放(key=token+ip,value=token)token
```

```
用户 <- 认证中心 : 认证成功返回token
用户 -> 认证中心: 下次访问头部携带token认证
认证中心 <- 缓存: key=token+ip获取token
其他服务 <- 认证中心: 存在且校验成功则跳转到用户请求的其他服务
其他服务 -> 用户: 信息
@enduml
```



## 1.2 声明参与者

使用`participant`关键字来声明一个参与者可以使你对参与者做出更多控制。

关键字`participant`用于改变参与者的先后顺序。

你也可以使用下面这些关键字来声明参与者，这会改变参与者的外观：

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

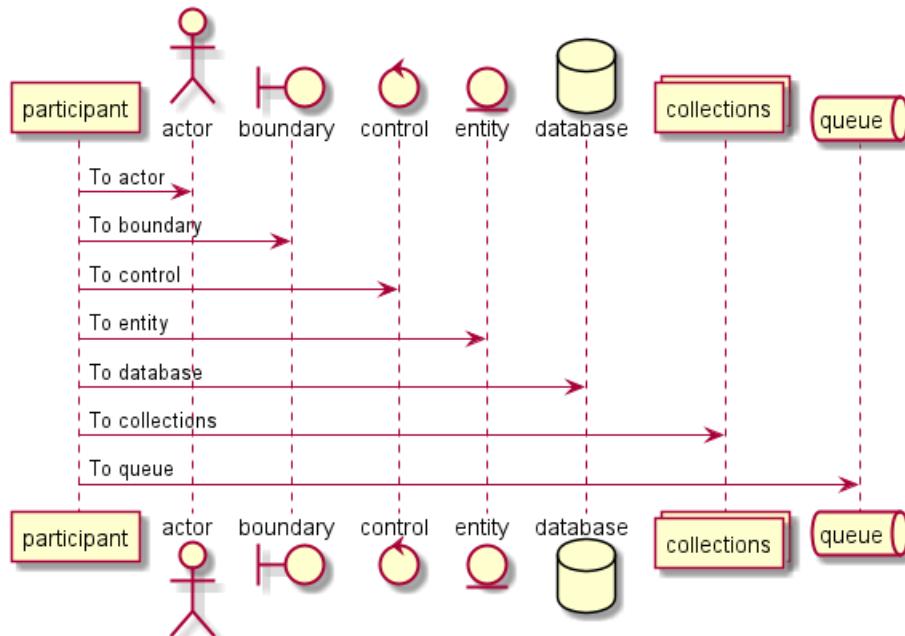
```
@startuml
participant participant as Foo
actor      actor      as Foo1
boundary   boundary   as Foo2
control    control    as Foo3
```



```

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

```



关键字 `as` 用于重命名参与者

你还可以使用 RGB 值或者颜色名修改 actor 或参与者的背景颜色。

```

@startuml
actor Bob #red
' actor 和 participant 只在外观上有区别
participant Alice
participant "很长很长很长\n的名字" as L #99FF99
/' 也可以这样声明:
    participant L as "很长很长很长\n的名字" #99FF99
    /

```

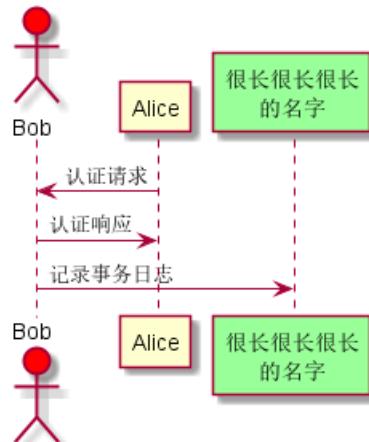
Alice->Bob: 认证请求

Bob->Alice: 认证响应

Bob->L: 记录事务日志

@enduml





您可以使用关键字 `order` 自定义顺序来打印参与者。

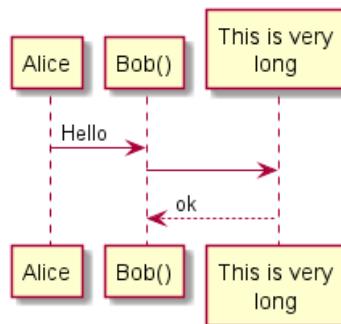
```
@startuml
participant 最后 order 30
participant 中间 order 20
participant 首个 order 10
@enduml
```



### 1.3 在参与者中使用非字母符号

你可以使用引号定义参与者，还可以用关键字 `as` 给参与者定义别名。

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

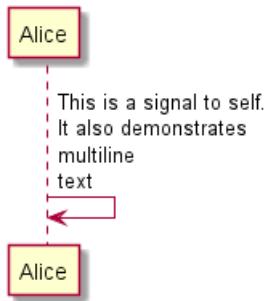


### 1.4 给自己发消息

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 \n\ttext
@enduml
```





## 1.5 文本对齐

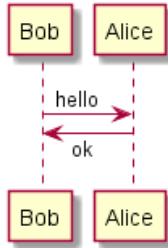
### 1.5.1 让响应信息显示在箭头下面

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

```



**TODO:** TODO 在 skinparam 创建本章节的链接。

## 1.6 修改箭头样式

修改箭头样式的方式有以下几种：

- 表示一条丢失的消息：末尾加 `x`
- 让箭头只有上半部分或者下半部分：将 `<` 和 `>` 替换成 \ 或者 /
- 细箭头：将箭头标记写两次（如 `>>` 或 `//`）
- 虚线箭头：用 `--` 替代 -
- 箭头末尾加圈：`->o`
- 双向箭头：`<->`

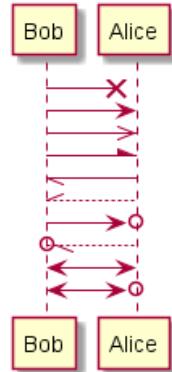
```

@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

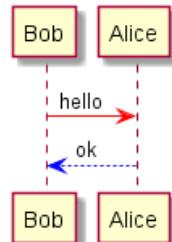
```



## 1.7 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
```



## 1.8 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

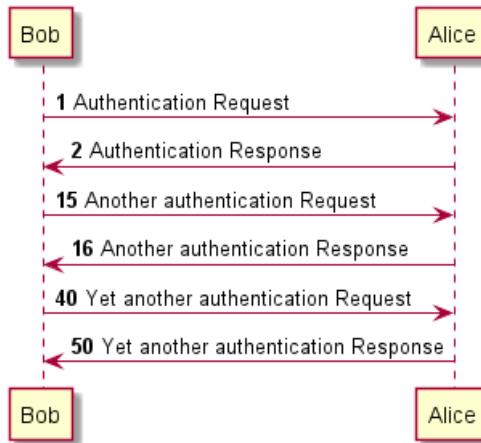
autonumber 15
Bob -> Alice : Another authentication Request
```



```

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

```



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。

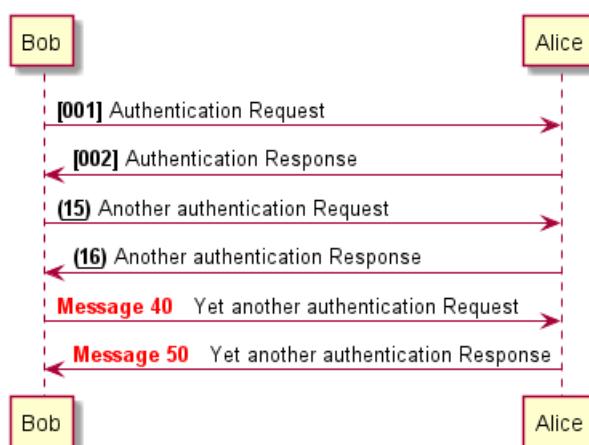
```

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

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

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

```



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]</b>"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

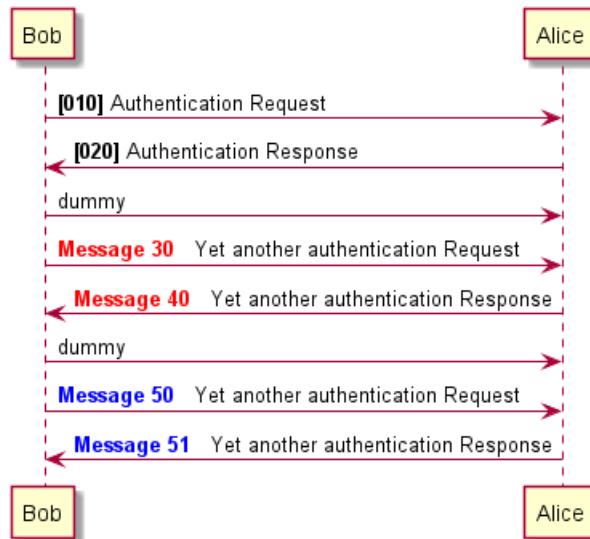
autonumber stop
Bob -> Alice : dummy

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

```



## 1.9 页面标题, 页眉, 页脚

使用 `title` 关键词增加标题  
 使用 `header` 关键词增加页眉  
 使用 `footer` 关键词增加页脚

```

@startuml

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

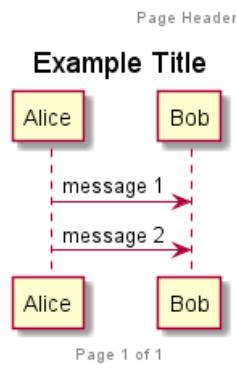
title Example Title

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

@enduml

```





## 1.10 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
```

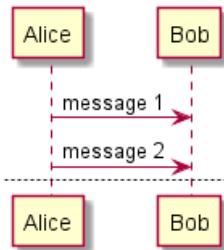
```
Alice -> Bob : message 1
Alice -> Bob : message 2
```

```
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
```



## 1.11 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
```

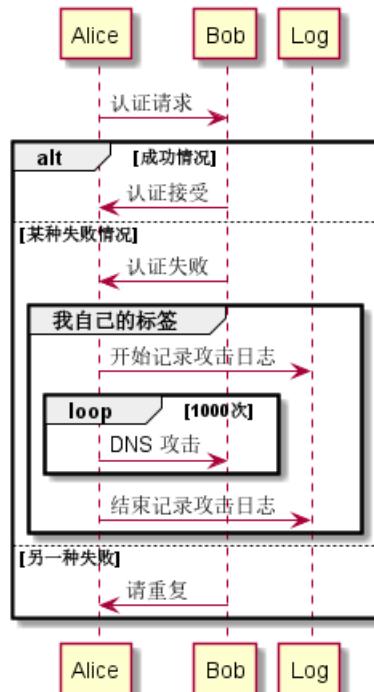
`else` 某种失败情况

```
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
```



## 1.12 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。

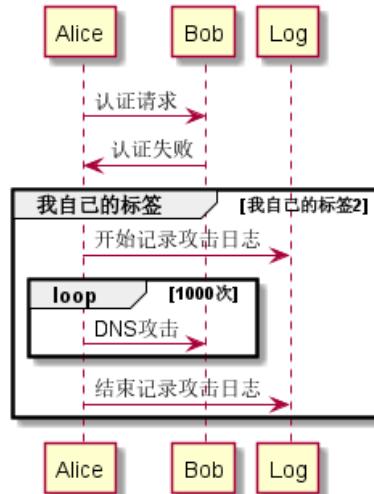
```
@startuml
Alice -> Bob: 认证请求
```



```

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]

### 1.13 给消息添加注释

我们可以通过在消息后面添加 `note left` 或者 `note right` 关键词来给消息添加注释。

你也可以通过使用 `end note` 来添加多行注释。

```

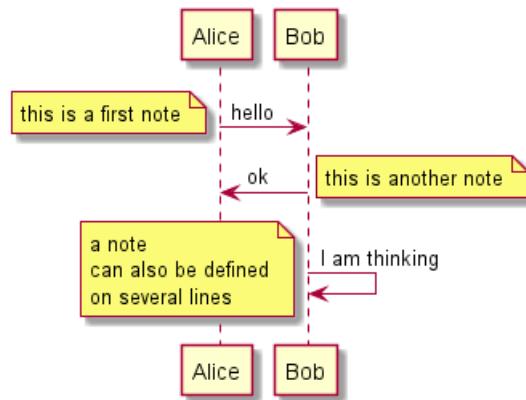
@startuml
Alice->Bob : hello
note left: this is a first note

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

```





## 1.14 其他的注释

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

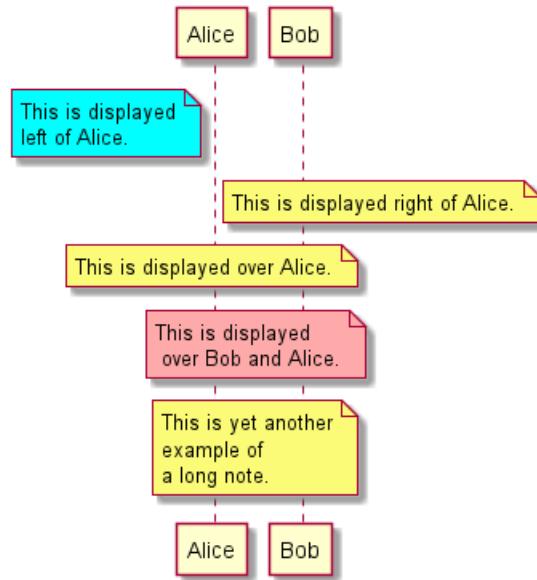
note right of Alice: This is displayed right of Alice.

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
```





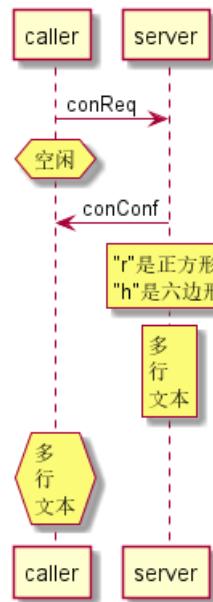
### 1.15 改变备注框的形状 [hnote 和 rnote]

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]

### 1.16 在多个参与者添加备注 [across]

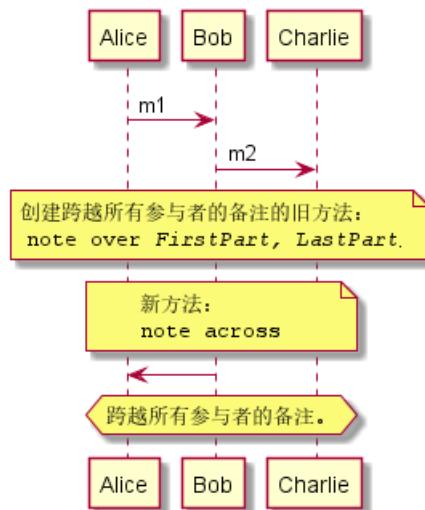
你可以之直接在所有参与者之间添加备注，格式是：

- 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]

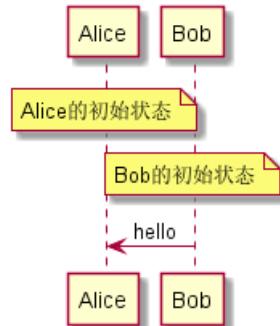
### 1.17 在同一级对齐多个备注 [/]

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。）

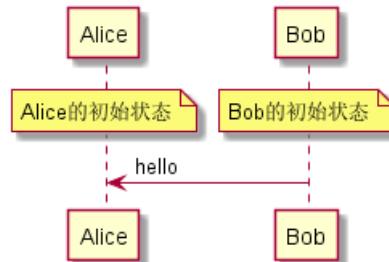


```
@startuml
note over Alice : Alice的初始状态
note over Bob : Bob的初始状态
Bob -> Alice : hello
@enduml
```



- 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]

## 1.18 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

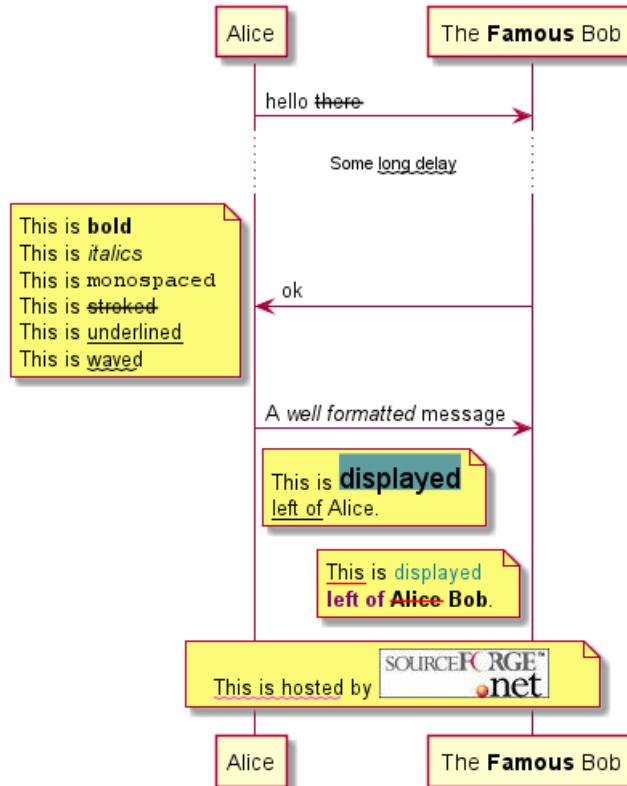
Alice -> Bob : A //well formatted// message
note right of Alice
    This is <back:cadetblue><size:18>displayed</size></back>
    __left of__ Alice.
```



```

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

```



## 1.19 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 ==
```

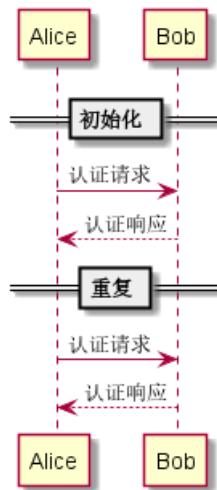
```
Alice -> Bob: 认证请求
Bob --> Alice: 认证响应
```

```
== 重复 ==
```

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

```
@enduml
```





## 1.20 引用

你可以在图中通过使用 `ref over` 关键词来实现引用

```

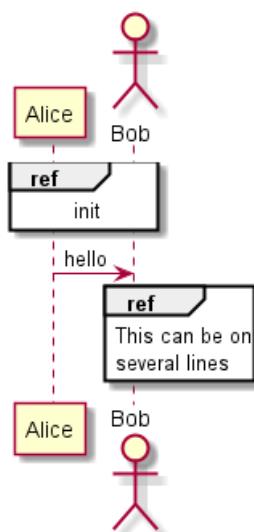
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

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

```



## 1.21 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



## 1.22 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

```



## 1.23 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
```

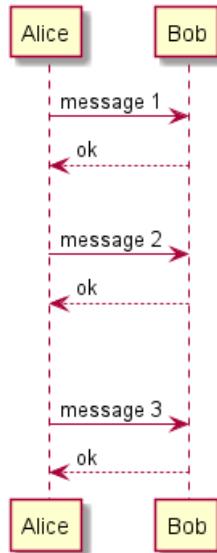


```

Bob --> Alice: ok
|||
Alice -> Bob: message 2
Bob --> Alice: ok
||45||
Alice -> Bob: message 3
Bob --> Alice: ok

@enduml

```



## 1.24 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

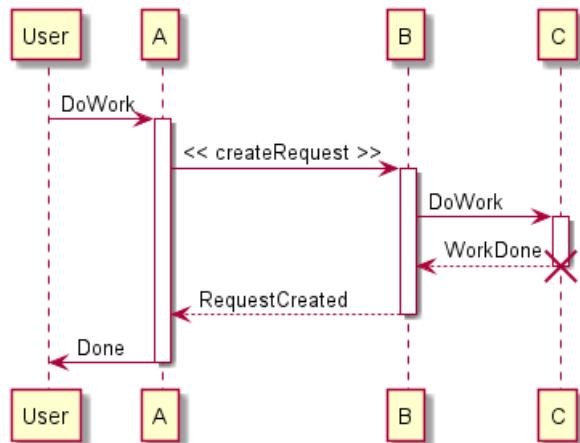
B --> A: RequestCreated
deactivate B

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

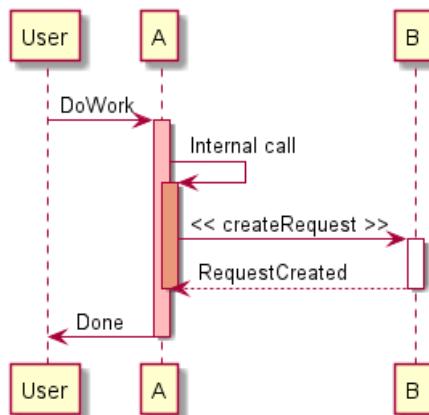
A -> A: Internal call
activate A #DarkSalmon

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

```

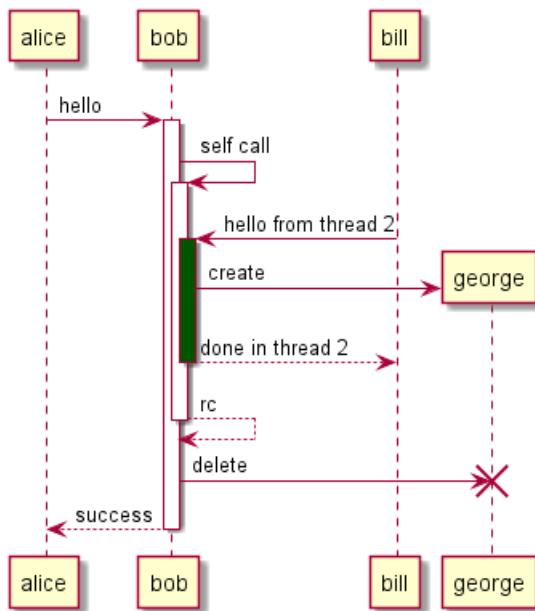


```

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

@enduml

```



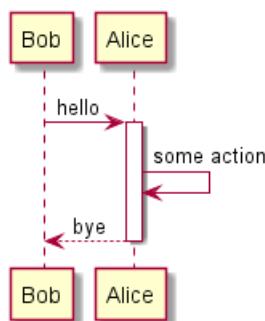
## 1.25 返回

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

```



## 1.26 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

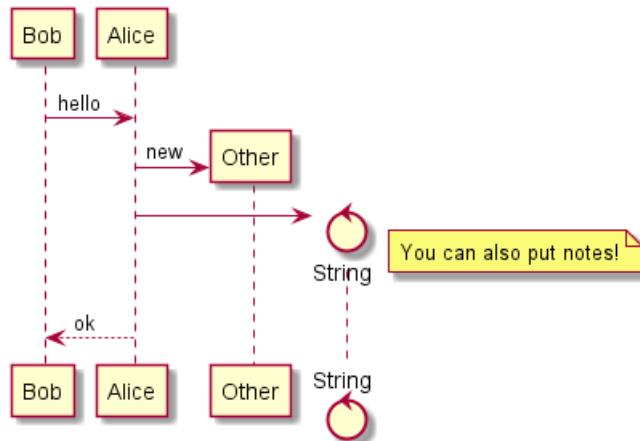
```



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

Alice --> Bob : ok

@enduml



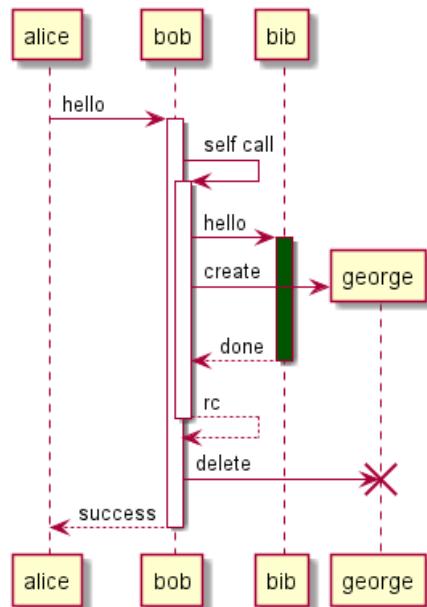
## 1.27 激活、撤销和创建的快捷语法。

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
- !! 崩毁目标实例

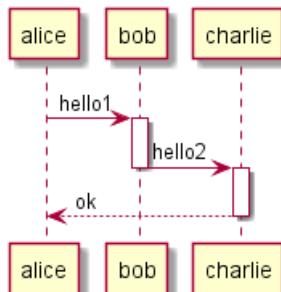
```
@startuml
alice -> bob ++ : hello
bob -> bob ++ : self call
bob -> bib ++ #005500 : hello
bob -> george ** : create
return done
return rc
bob -> george !! : delete
return success
@enduml
```



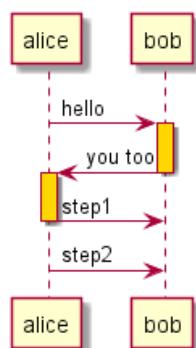


然后你就可以在一行上同时激活和撤销:

```
@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]

## 1.28 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。

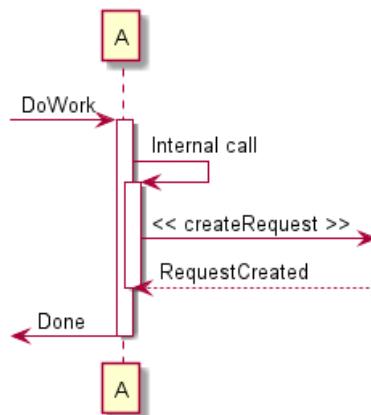
```
@startuml
[-> A: DoWork

activate A

A -> A: Internal call
activate A

A ->] : << createRequest >>

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



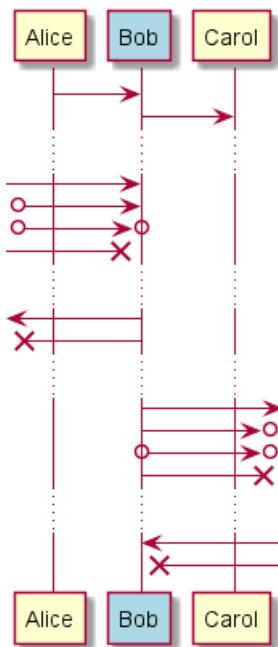
还可以使用下面的语法:

```
@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
```



## 1.29 缩短的进入信息与发出信息箭头

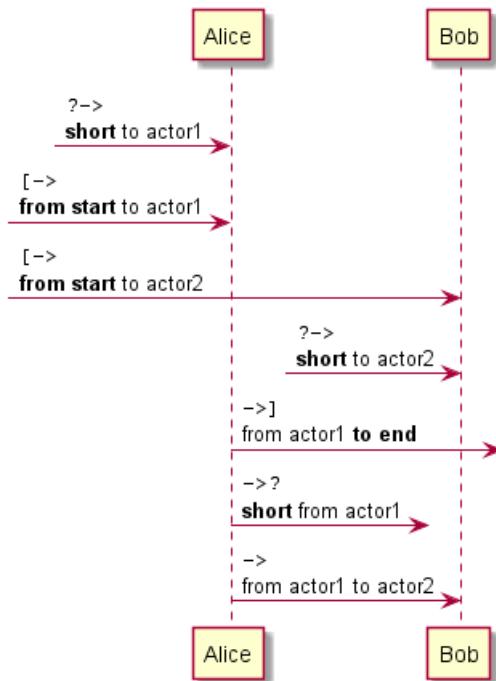
使用? 来显示缩短的箭头。

```

@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]

## 1.30 锚定和持续时间

使用 `teoz` 在图表中添加锚定，从而指定持续时间。

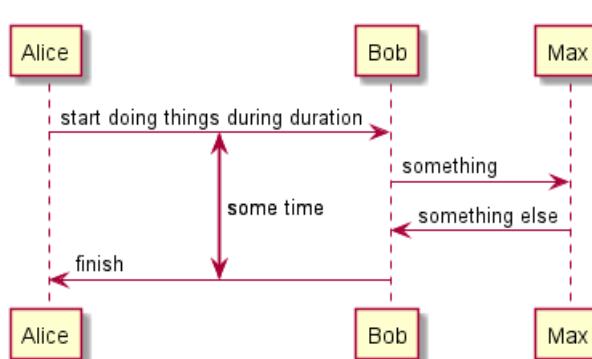
```

@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
  
```



## 1.31 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。

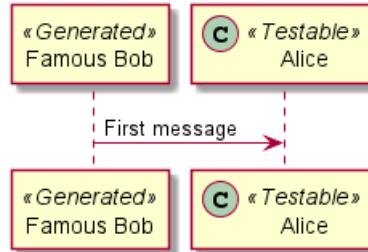
```
@startuml
```



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

Bob->Alice: First message

@enduml



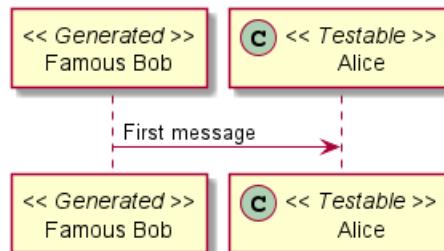
默认使用 *guillemet* 字符来显示构造类型。你可以使用外观参数 `guillemet` 来修改显示行为。

@startuml

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

Bob->Alice: First message

@enduml

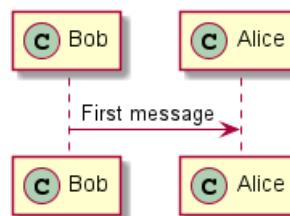


@startuml

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

Bob->Alice: First message

@enduml



## 1.32 更多标题信息

你可以在标题中使用 creole 格式。

@startuml



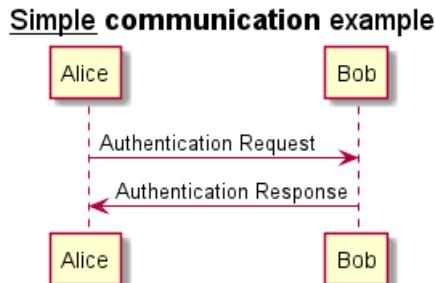
```

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。

```

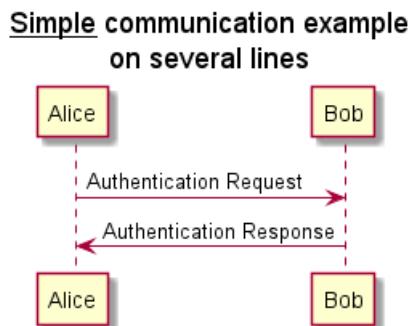
@startuml

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。

```

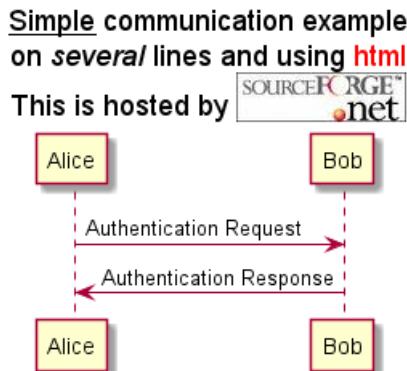
@startuml

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

```



### 1.33 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。

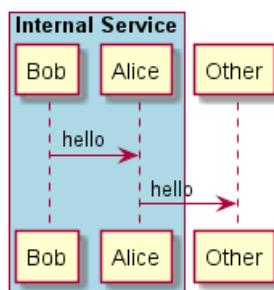
`@startuml`

```

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

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

`@enduml`



### 1.34 移除脚注

使用 `hide footbox` 关键字移除脚注。

`@startuml`

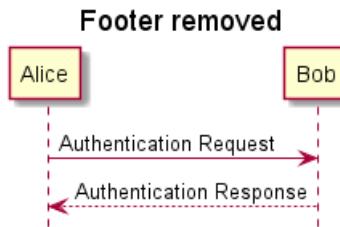
```

hide footbox
title Footer removed

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

`@enduml`





### 1.35 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

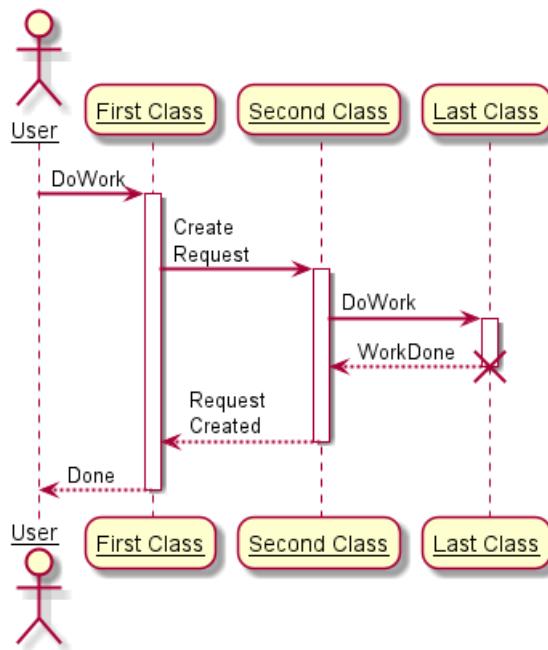
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

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

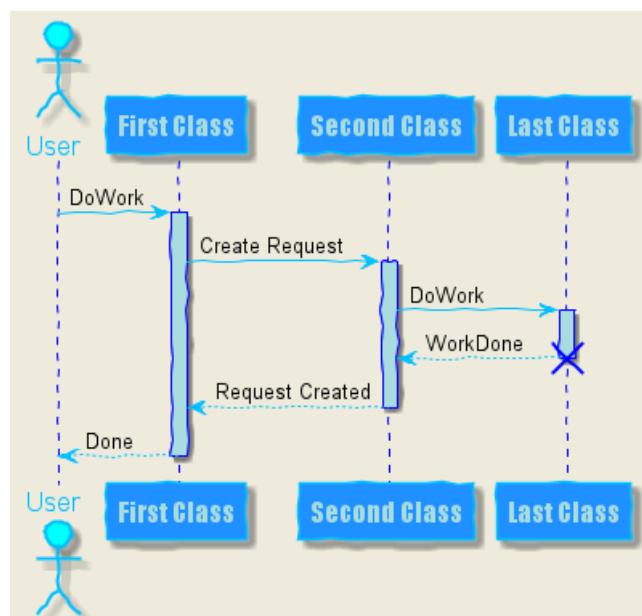
```



```
B --> A: Request Created
deactivate B
```

```
A --> User: Done
deactivate A
```

```
@enduml
```



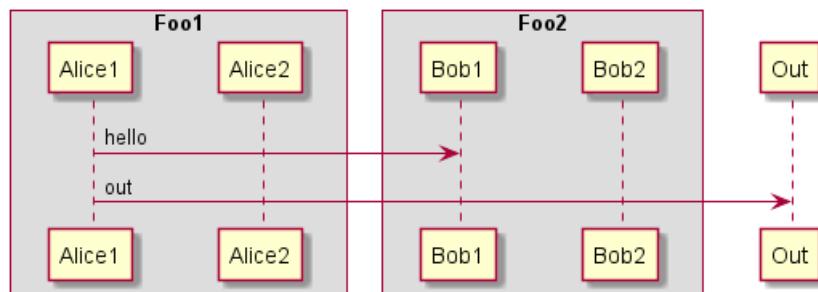
## 1.36 填充区设置

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
  
```

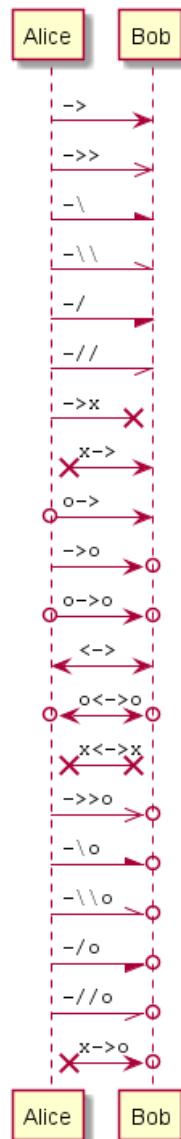


## 1.37 附录：箭头类型大全

### 1.37.1 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
```





### 1.37.2 进入信息和发出信息（使用'[, ]'）

### 1.37.3 进入信息（使用'[']）

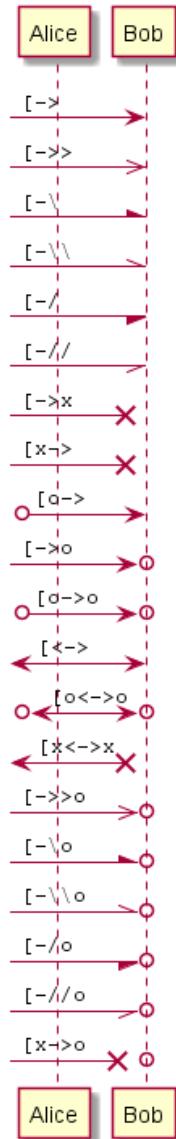
```
@startuml
participant Alice as a
participant Bob as b
[-> b : ""[-> """
[->> b : ""[->> """
[-\ b : ""[-\ """
[-\\\" b : ""[-\\\""""
[-/ b : ""[-/ """
[-// b : ""[-// """
[->x b : ""[->x """
[x-> b : ""[x-> """
[o-> b : ""[o-> """
[->o b : ""[->o """
[o->o b : ""[o->o """
[<-> b : ""[<-> """
[o<->o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

```



#### 1.37.4 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]   """

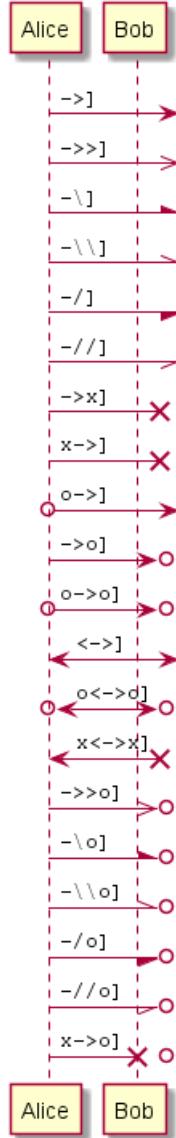
```



```

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

```



### 1.37.5 短进入信息和短发出信息（使用'?'）

### 1.37.6 短进入信息（使用'?'）

```

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

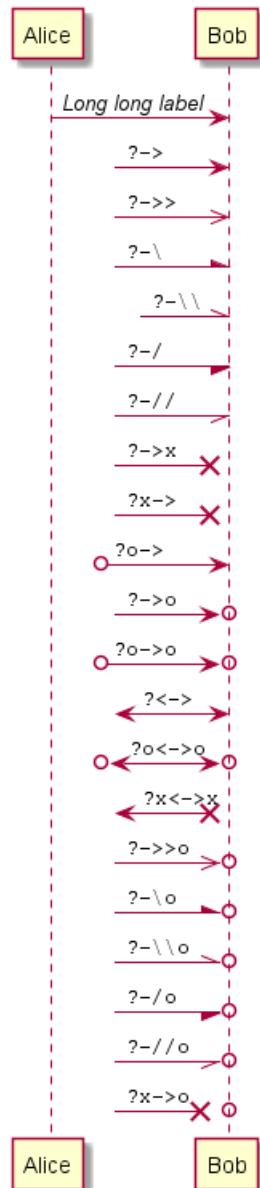
```



```

?-\  
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

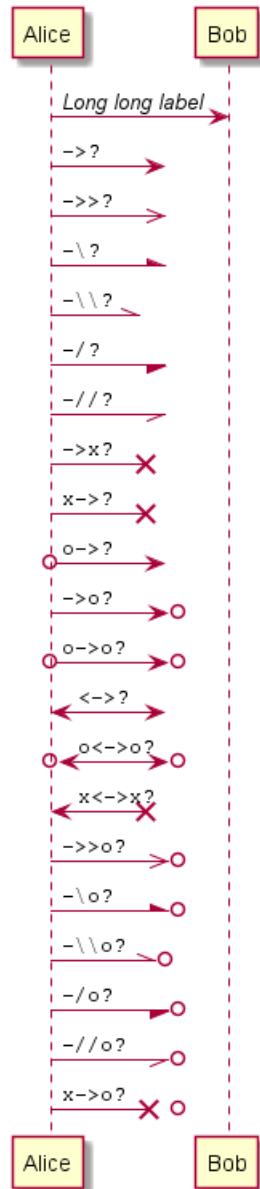
```



### 1.37.7 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
```



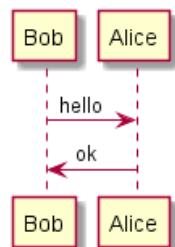


## 1.38 特定外观参数

### 1.38.1 默认情况下

```

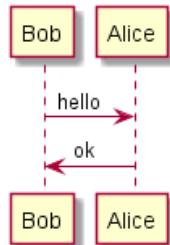
@startuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
  
```



### 1.38.2 生命线策略

- nosolid 虚线（默认情况）

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

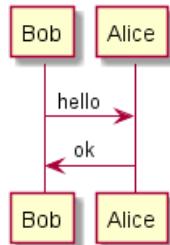


[参见 QA-9016]

- 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
```



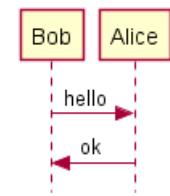
[参见 QA-2794]

### 1.38.3 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
```



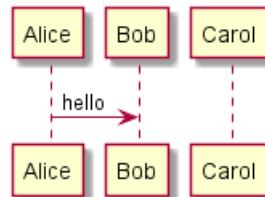
[参见 QA-1047]



### 1.39 隐藏孤立参与者

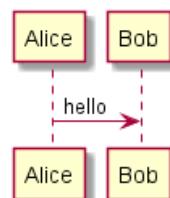
默认情况下会显示所有参与者。

```
@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]



## 2 用例图

举几个例子：

### 2.1 用例

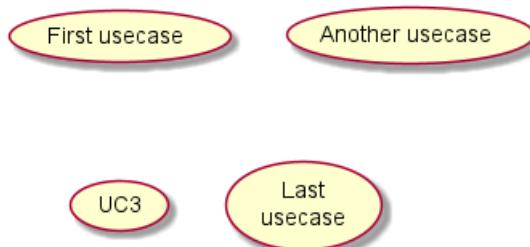
用例用圆括号括起来（两个圆括号看起来就像椭圆）。

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\usecase) as UC4
```

```
@enduml
```



### 2.2 角色

角色用两个冒号包裹起来。

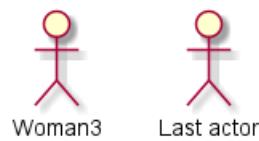
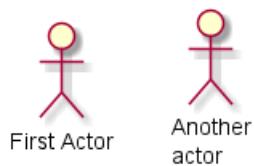
也可以用 `actor` 关键字来定义角色。还可以用关键字 `as` 来定义一个别名，这个别名可以在以后定义关系的时候使用。

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\actor: as Man2
actor Woman3
actor :Last actor: as Person1
```

```
@enduml
```



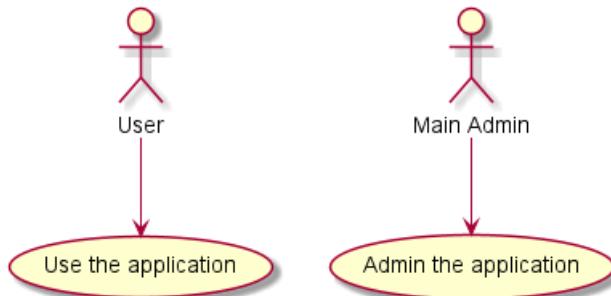
### 2.3 改变角色的样式

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`

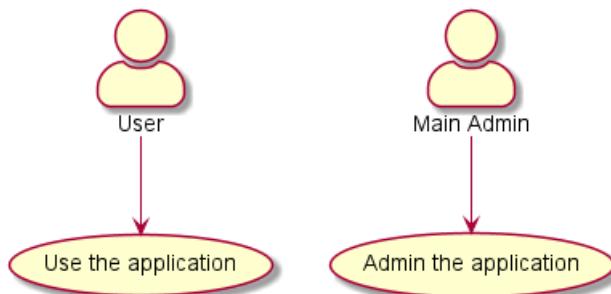
#### 2.3.1 火柴人 默认

```
@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
```

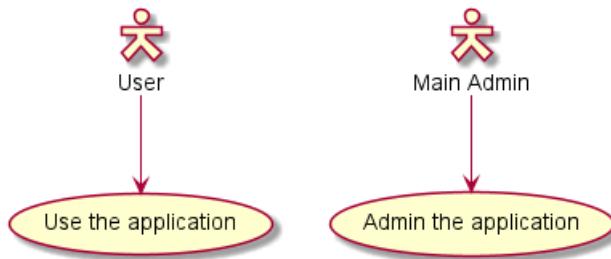


[参见 QA-10493]

#### 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

```

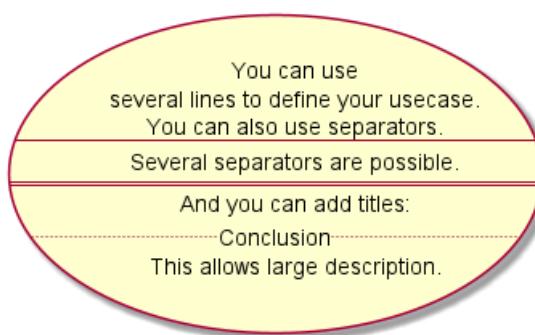
usecase UC1 as "You can use
several lines to define your usecase.
You can also use separators.

--
Several separators are possible.

==

And you can add titles:
..Conclusion..
This allows large description."
  
```

@enduml



## 2.5 使用包

您可以一使用包来对角色或用例进行分组。

```

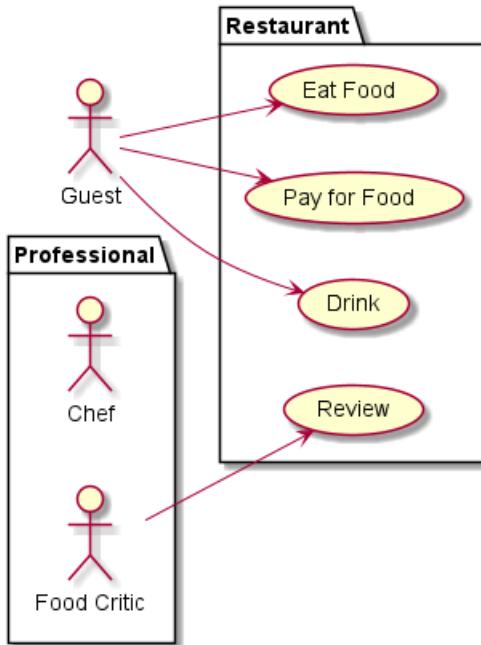
@startuml
left to right direction
actor Guest as g
package Professional {
    actor Chef as c
    actor "Food Critic" as fc
  
```



```

}
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
@genduml

```



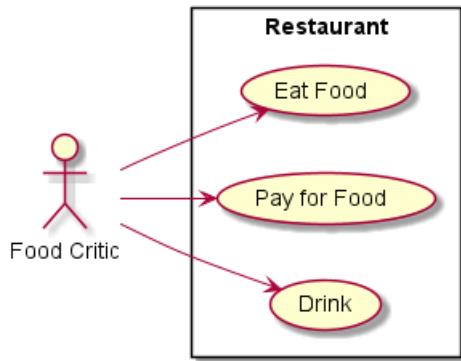
您可以使用 `rectangle` 来改变包的外观。

```

@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
@genduml

```





## 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

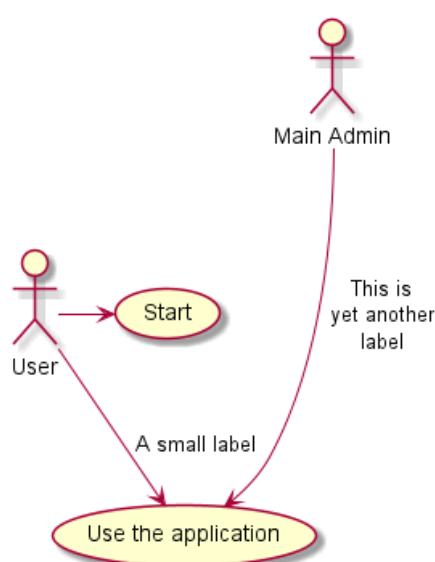
```

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)

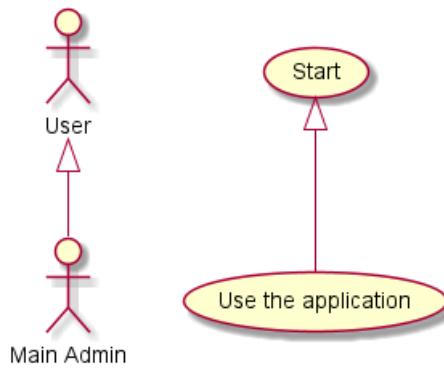
User <|-- Admin
(Start) <|-- (Use)

@enduml

```

@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)
```

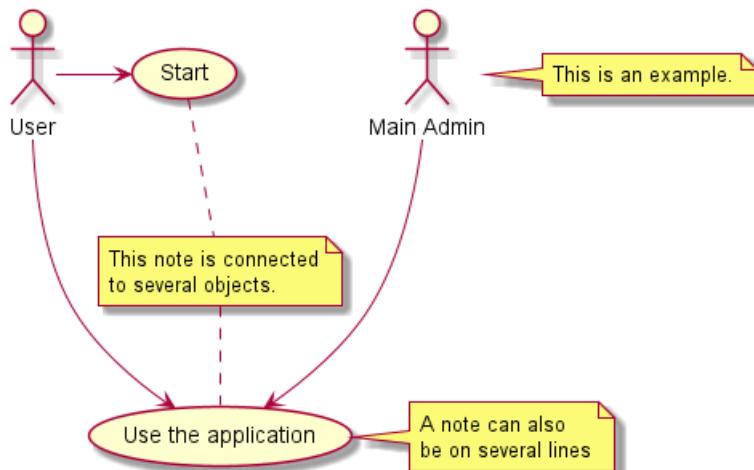
```
User -> (Start)
User --> (Use)
```

```
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\n to 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。

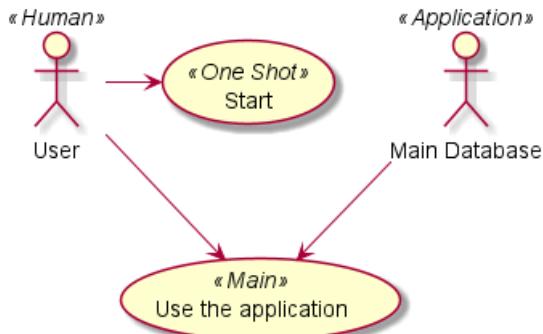


```
@startuml
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

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

MySql --> (Use)

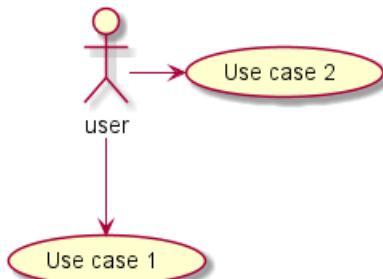
@enduml
```



## 2.10 改变箭头方向

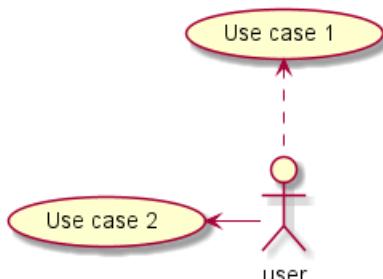
默认连接是竖直方向的，用--表示，可以用一个横杠或点来表示水平连接。

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



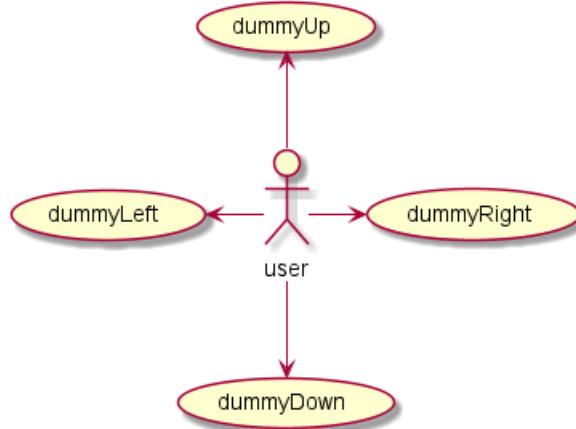
也可以通过翻转箭头来改变方向。

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



还可以通过给箭头添加 `left`, `right`, `up` 或 `down` 等关键字来改变方向。

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



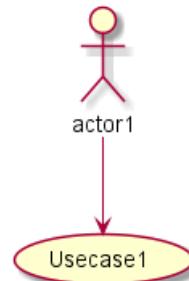
这些方向关键字也可以只是用首字母或者前两个字母的缩写来代替。

但是请注意，这样的缩写不要乱用，Graphviz 不喜欢这样。

## 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
```



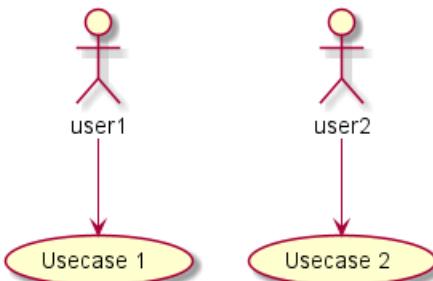
## 2.12 从左向右方向

默认从上往下构建图示。

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

@enduml
```





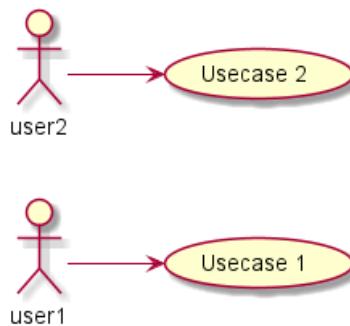
你可以用 `left to right direction` 命令改变图示方向。

`@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
}
  
```

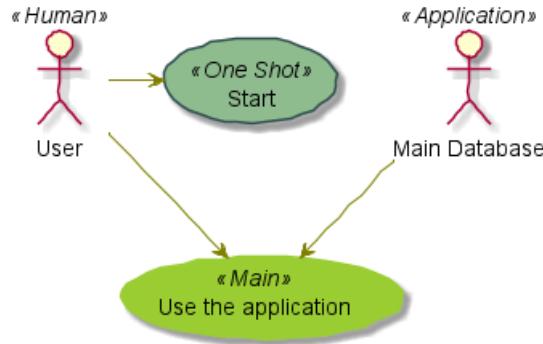


```
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>
```

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

```
MySql --> (Use)
```

```
@enduml
```

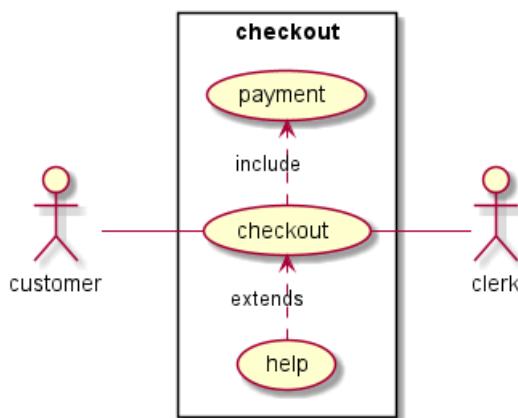


## 2.14 一个完整的例子

```

@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 Business Use Case

You can add / to make Business Use Case.

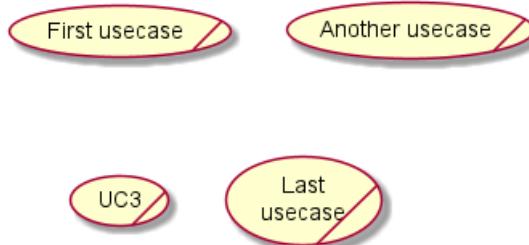


### 2.15.1 Business Usecase

```
@startuml
```

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

```
@enduml
```

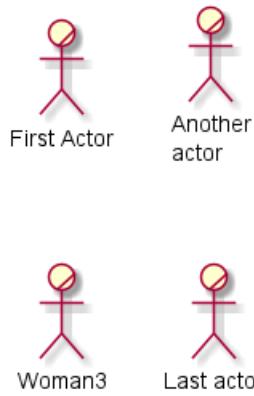


### 2.15.2 Business Actor

```
@startuml
```

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

```
@enduml
```



[Ref. QA-12179]

## 2.16 Change arrow color and style (inline style)

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

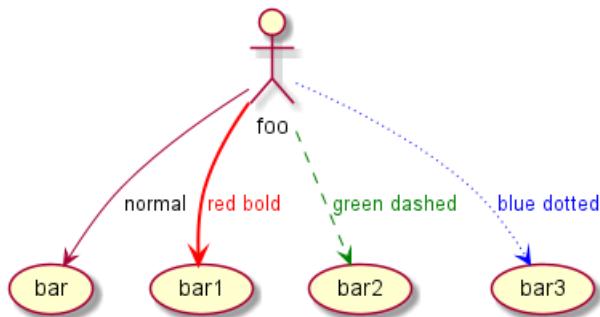
- #color;line.[bold|dashed|dotted];text:color

```
@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
```





[Ref. QA-3770 and QA-3816] [See similar feature on deployment-diagram or class diagram]

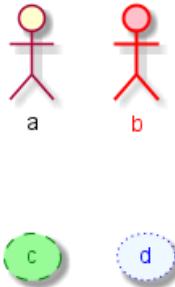
## 2.17 Change element color and style (inline style)

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

- # [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
  
```



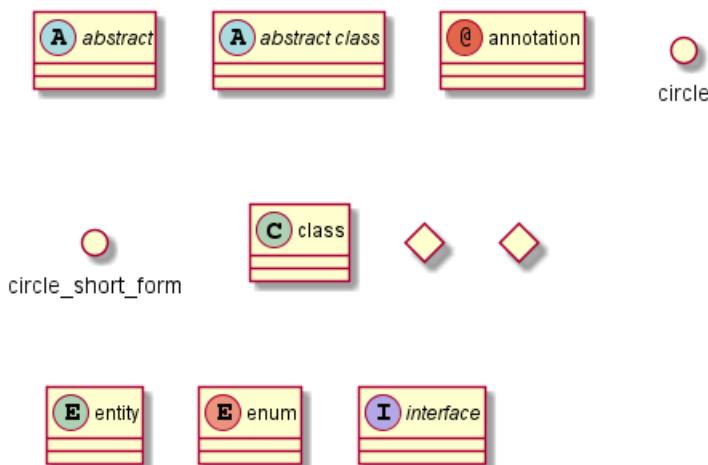
[Ref. QA-5340 and adapted from QA-6852]



## 3 类图

### 3.1 元素声明

```
@startuml
abstract      abstract
abstract class "abstract class"
annotation    annotation
circle        circle
()            circle_short_form
class         class
diamond       diamond
<>           diamond_short_form
entity        entity
enum          enum
interface     interface
@enduml
```



### 3.2 类之间的关系

类之间的关系通过下面的符号定义:

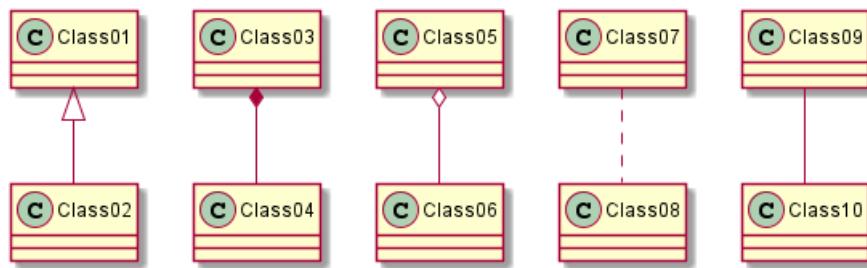
Type	Symbol	Drawing
Extension (扩展)	< --	←—
Composition (组合)	*--	◆—
Aggregation (聚合)	o--	◇—

使用.. 来代替 -- 可以得到点线.

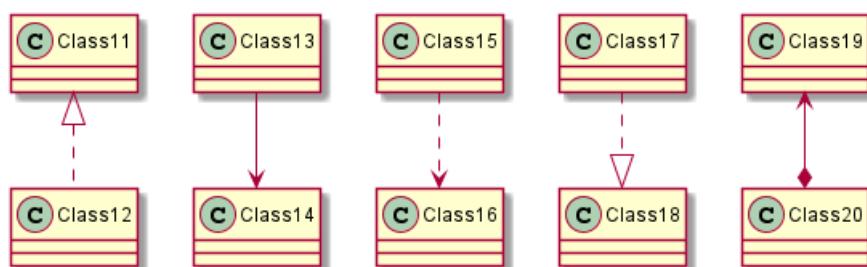
在这些规则下，也可以绘制下列图形

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

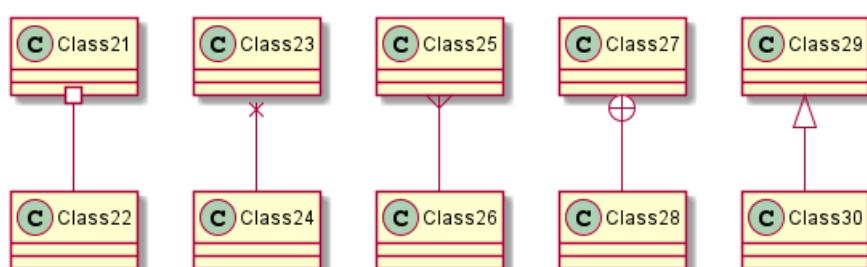




```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```



```
@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

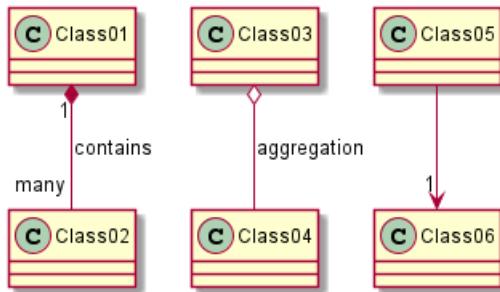
Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation

Class05 --> "1" Class06

@enduml
```





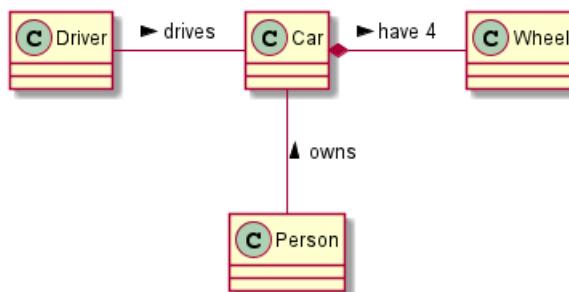
在标签的开始或结束位置添加 < 或 > 以表明是哪个对象作用到哪个对象上。

```

@startuml
class Car

Driver -> Car : drives >
Car *-- Wheel : have 4 >
Car --> Person : < owns
  
```

@enduml



## 3.4 添加方法

为了声明字段 (对象属性) 或者方法, 你可以使用后接字段名或方法名。

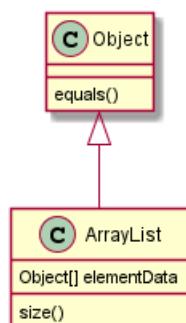
系统检查是否有括号来判断是方法还是字段。

```

@startuml
Object <|-- ArrayList

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

@enduml
  
```



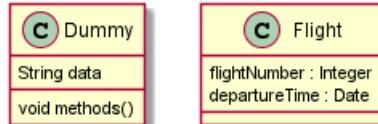
也可以使用 {} 把字段或者方法括起来

注意, 这种语法对于类型/名字的顺序是非常灵活的。



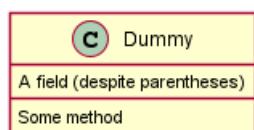
```
@startuml
class Dummy {
    String data
    void methods()
}

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



你可以（显式地）使用 {field} 和 {method} 修饰符来覆盖解析器的对于字段和方法的默认行为

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



## 3.5 定义可访问性

一旦你定义了域或者方法，你可以定义相应条目的可访问性质。

Character	Icon for field	Icon for method	Visibility
-	□	■	private
#	◊	◊	protected
~	△	△	package private
+	○	●	public

```
@startuml
```

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

```
@enduml
```



你可以采用以下命令停用这些特性 `skinparam classAttributeIconSize 0` :

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

@enduml
```

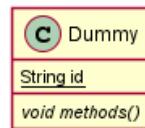


## 3.6 抽象与静态

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
```



## 3.7 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
}
```

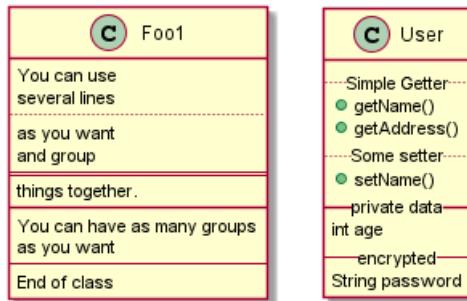


```

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

```

@enduml



## 3.8 备注和模板

模板通过类关键字 ("<<" 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 来添加备注。

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.

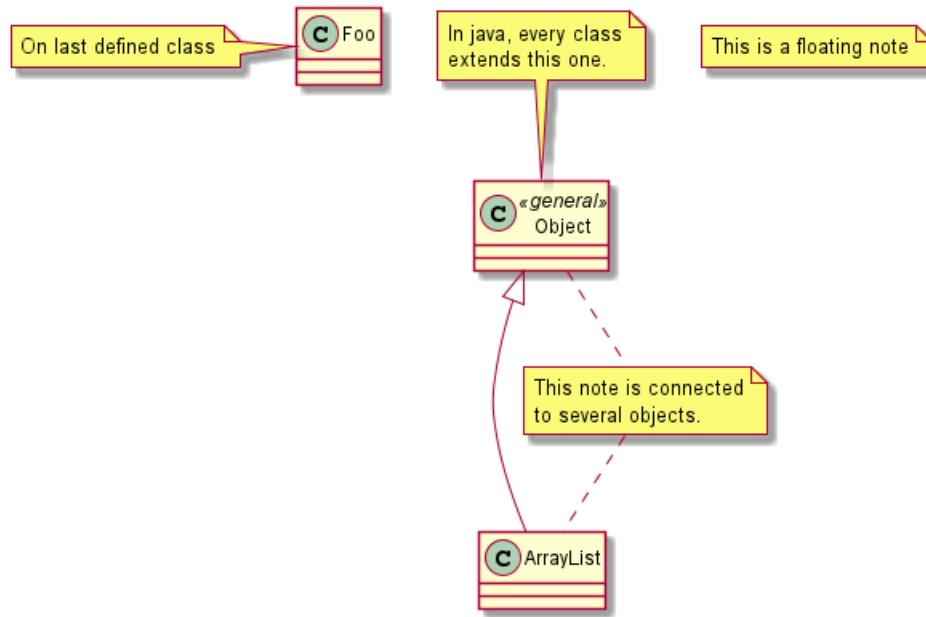
note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2
Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class

@enduml

```





### 3.9 更多注释

可以在注释中使用部分 html 标签:

- <b>
- <u>
- <i>
- <s>, <del>, <strike>
- <font color="#AAAAAA"> or <font color="colorName">
- <color:#AAAAAA> or <color:colorName>
- <size:nn> to change font size
-  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

```

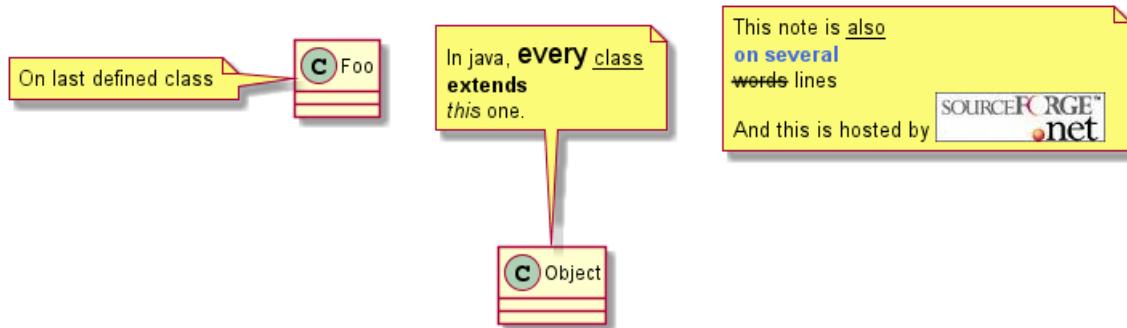
class Foo
note left: On last defined class

note top of Object
  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



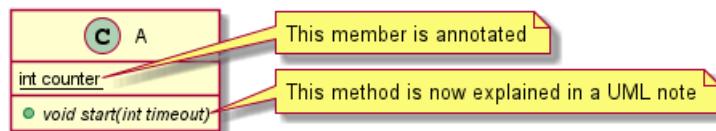


## 3.10 Note on field (field, attribute, member) or method

It is possible to add a note on field (field, attribut, member) or on method.

### 3.10.1 Note on field or method

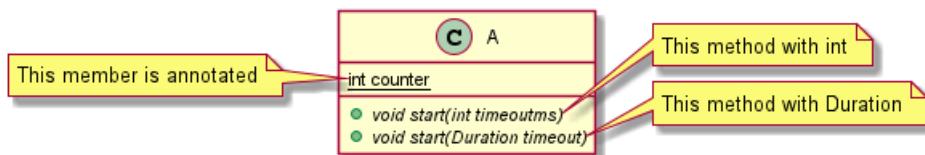
```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeout)
}
note right of A::counter
    This member is annotated
end note
note right of A::start
    This method is now explained in a UML note
end note
@enduml
```



### 3.10.2 Note on method with the same name

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeoutms)
+void {abstract} start(Duration timeout)
}
note left of A::counter
    This member is annotated
end note
note right of A::"start(int timeoutms)"
    This method with int
end note
note right of A::"start(Duration timeout)"
    This method with Duration
end note
@enduml
```





[Ref. QA-3474 and QA-5835]

### 3.11 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`

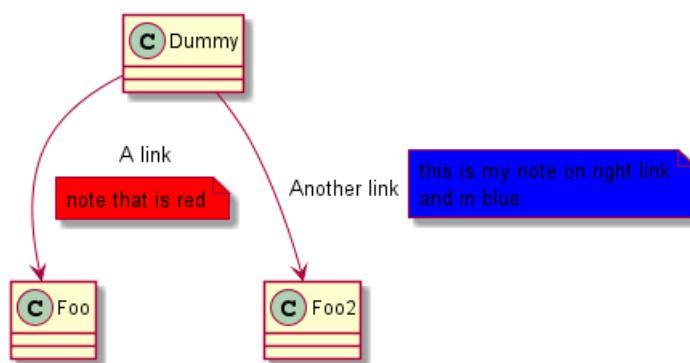
```

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

```



### 3.12 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` 关键字。

`@startuml`

```

abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

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

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

```



```

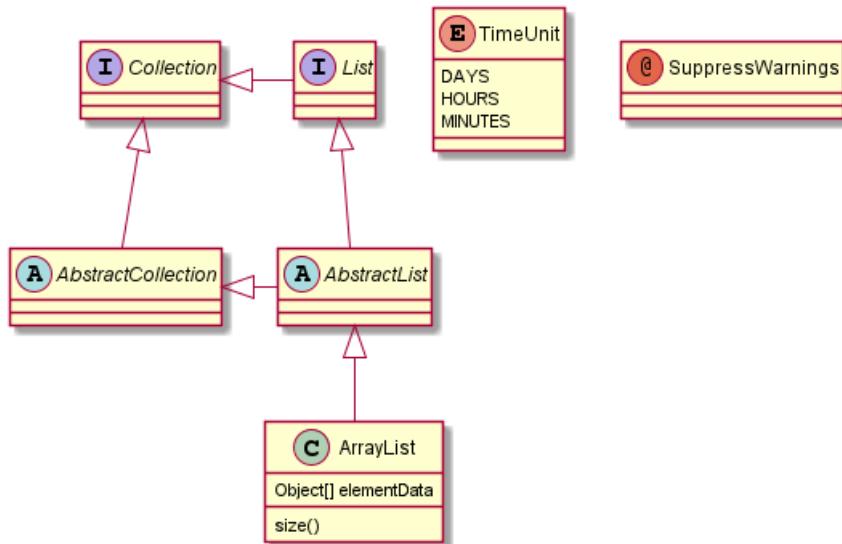
class ArrayList {
    Object[] elementData
    size()
}

enum TimeUnit {
    DAYS
    HOURS
    MINUTES
}

annotation SuppressWarnings

@enduml

```



\*[Ref. 'Annotation with members' [Issue#458](<https://github.com/plantuml/plantuml/issues/458>)]\*

### 3.13 使用非字母字符

如果你想在类（或者枚举）的显示中使用非字母字符，你可以：

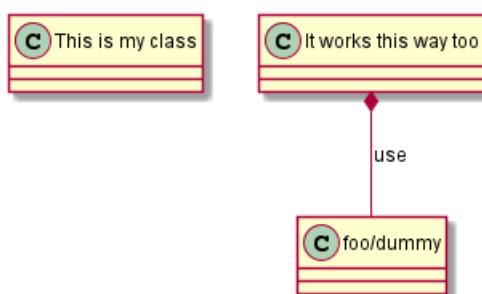
- 在类的定义中使用 `as` 关键字
- 在类名旁边加上 `""`

```

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

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

```



### 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
```

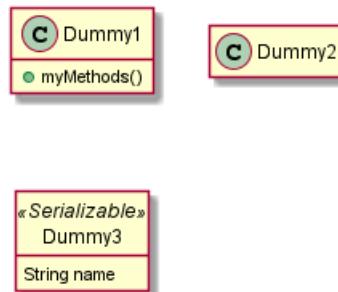
```
class Dummy1 {
    +myMethods()
}

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 Remove classes

You can also use the `remove` commands to remove classes.

This may be useful if you define a large `!included` file, and if you want to remove some classes after file inclusion.

```
@startuml
```

```
class Foo1
class Foo2
```

```
Foo2 *-- Foo1
```

```
remove Foo2
```

```
@enduml
```

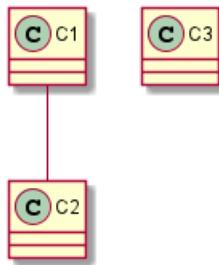


### 3.17 Hide or Remove unlinked class

By default, all classes are displayed:

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





But you can:

- hide @unlinked classes:

```

@startuml
class C1
class C2
class C3
C1 --> C2

hide @unlinked
@enduml

```



- or remove @unlinked classes:

```

@startuml
class C1
class C2
class C3
C1 --> C2

remove @unlinked
@enduml

```



[Adapted from QA-11052]

### 3.18 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.

### 3.19 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) .

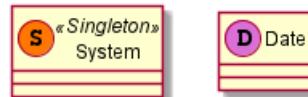
但是当你想定义原型时, 可以增加对应的单个字符及颜色, 来定义自己的标记 (spot), 就像下面一样:

```
@startuml
```

```

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

```



### 3.20 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
}

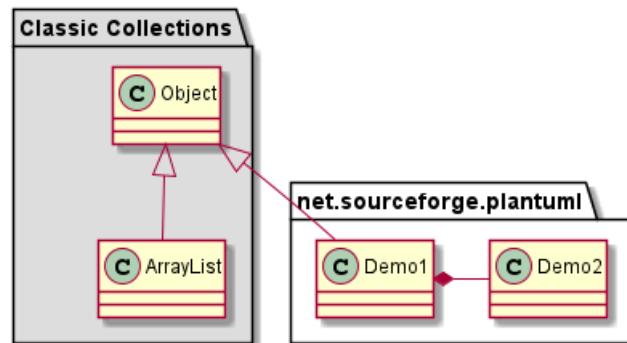
```

```

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

```

```
@enduml
```



## 3.21 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
}

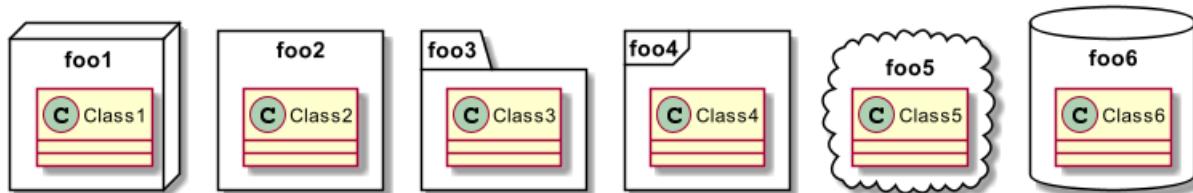
package foo4 <<Frame>> {
    class Class4
}

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

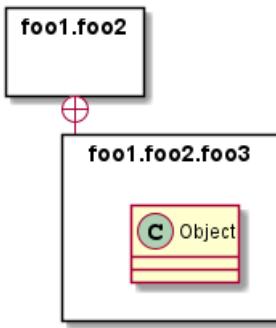
package foo1.foo2 {
}

package foo1.foo2.foo3 {
    class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```





## 3.22 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
}

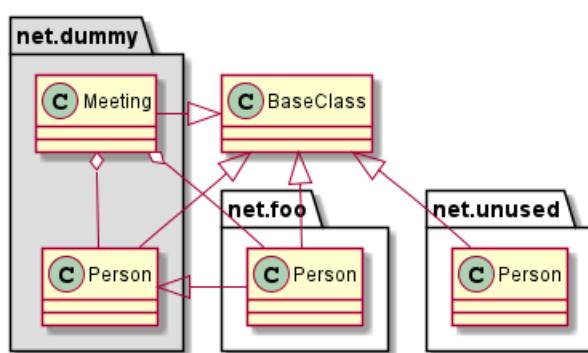
namespace net.foo {
    net.dummy.Person <|- Person
    .BaseClass <|-- Person

    net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

```

@enduml



## 3.23 自动创建命名空间

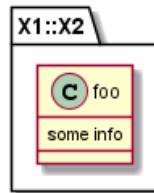
使用命令 `set namespaceSeparator ???` 你可以自定义命名空间分隔符 (为“.”以外的字符) .



```
@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
```



## 3.24 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  
bar ()- foo  
@enduml
```

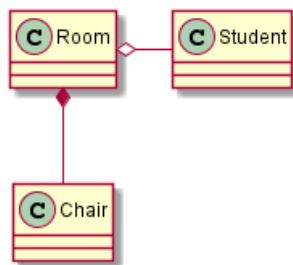


## 3.25 改变箭头方向

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  
Room o- Student  
Room *--- Chair  
@enduml
```

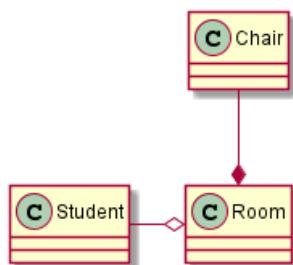




你也可以通过改变倒置链接来改变方向

```

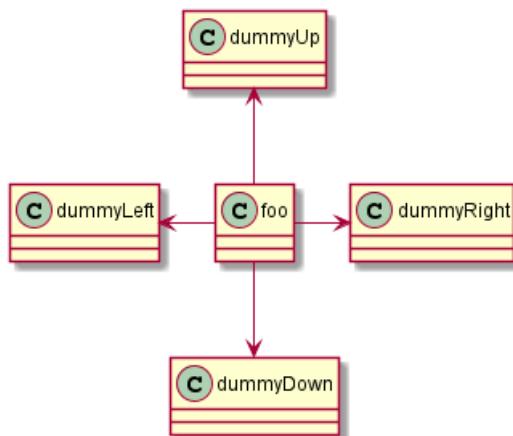
@startuml
Student -o Room
Chair --* Room
@enduml
  
```



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
  
```



You can shorten the arrow by using only the first character of the direction (for example, -d- instead of -down-) or the two first characters (-do-).

Please note that you should not abuse this functionality : *Graphviz* gives usually good results without tweaking.

## 3.26 “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 {
  
```

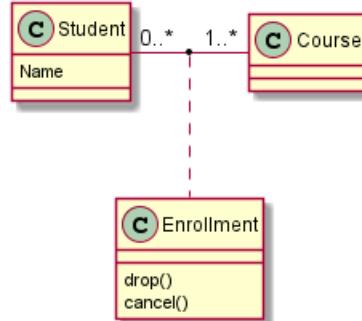


```

Name
}
Student "0..*" - "1..*" Course
(Student, Course) .. Enrollment

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

```



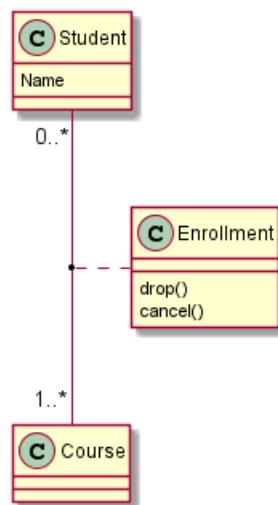
也可以用另一种方式:

```

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

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

```



### 3.27 Association on same classe

```

@startuml
class Station {
    +name: string

```



```

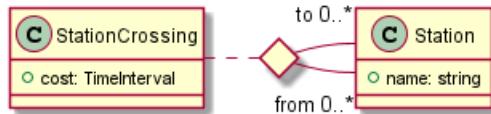
}

class StationCrossing {
    +cost: TimeInterval
}

<> diamond

StationCrossing . diamond
diamond - "from 0..*" Station
diamond - "to 0..*" Station
@enduml

```



[Ref. Incubation: Associations]

## 3.28 皮肤参数

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

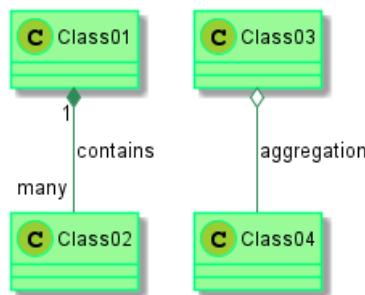
```

skinparam class {
    backgroundColor PaleGreen
    arrowColor SeaGreen
    bordercolor SpringGreen
}
skinparam stereotypeBackgroundColor YellowGreen

Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation

@enduml

```



## 3.29 Skinned Stereotypes

You can define specific color and fonts for stereotyped classes.



```

@startuml

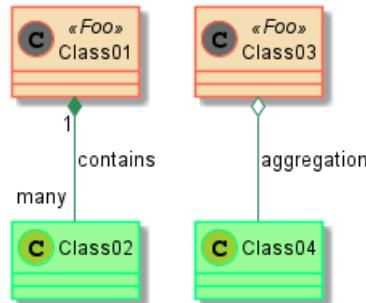
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
BackgroundColor<<Foo>> Wheat
BorderColor<<Foo>> Tomato
}
skinparam stereotypeBackgroundColor YellowGreen
skinparam stereotypeBackgroundColor<< Foo >> DimGray

Class01 <<Foo>>
Class03 <<Foo>>
Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation

@enduml

```



### 3.30 Color gradient

You can declare individual colors for classes, notes etc using the # notation.

You can use standard color names or RGB codes in various notations, see Colors.

You can also use color gradient for background colors, with the following syntax: two colors names separated either by:

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

depending on the direction of the gradient.

For example:

```
@startuml
```

```

skinparam backgroundcolor AntiqueWhite/Gold
skinparam classBackgroundColor Wheat|CornflowerBlue

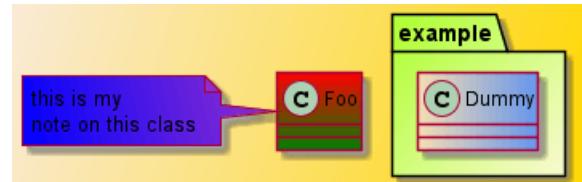
class Foo #red-green
note left of Foo #blue\9932CC
    this is my
    note on this class
end note

```



```
package example #GreenYellow/LightGoldenRodYellow {
    class Dummy
}

@enduml
```



### 3.31 辅助布局

有时候， 默认布局并不完美...

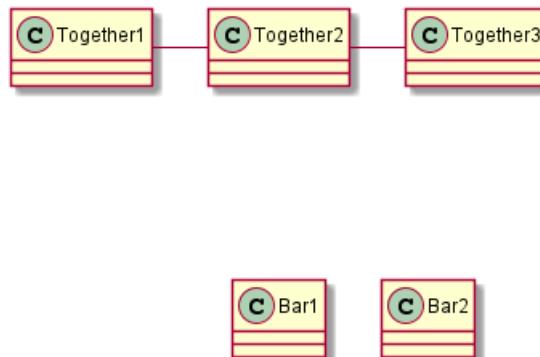
你可以使用 `together` 关键词将某些类进行分组：布局引擎会尝试将它们捆绑在一起（如同在一个包 (package) 内）

你也可以使用建立 隐藏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
```



### 3.32 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

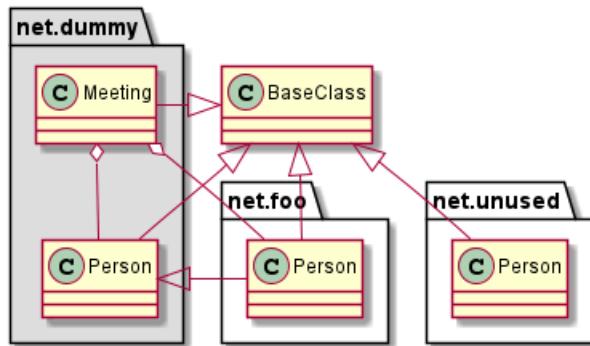
namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

    .BaseClass <|- Meeting
}

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

    net.dummy.Meeting o-- Person
}

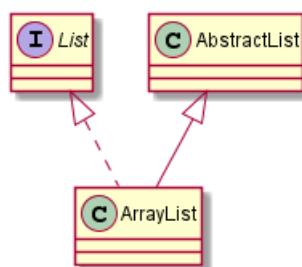
BaseClass <|-- net.unused.Person
@enduml
```



### 3.33 Extends and implements

It is also possible to use `extends` and `implements` keywords.

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



### 3.34 Bracketed relations (linking or arrow) style

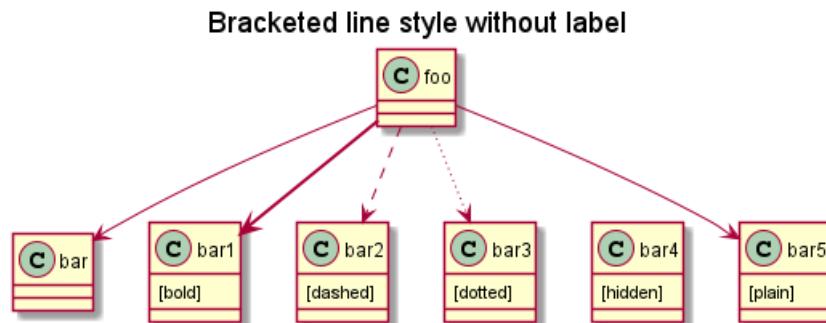
#### 3.34.1 Line style

It's also possible to have explicitly bold, dashed, dotted, hidden or plain relation, links or arrows:

- without label

```
@startuml
title Bracketed line style without label
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml
```



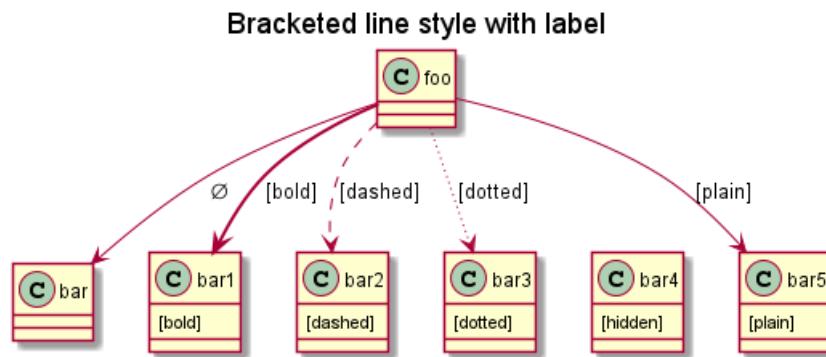
- with label

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

foo --> bar      :
foo -[bold]-> bar1 : [bold]
foo -[dashed]-> bar2 : [dashed]
foo -[dotted]-> bar3 : [dotted]
foo -[hidden]-> bar4 : [hidden]
foo -[plain]-> bar5 : [plain]

@enduml
```





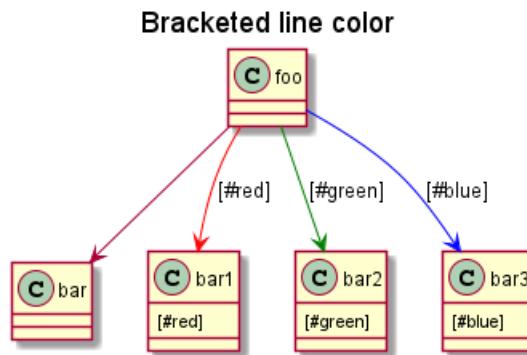
[Adapted from QA-4181]

### 3.34.2 Line color

```

@startuml
title Bracketed line color
class foo
class bar
bar1 : [#red]
bar2 : [#green]
bar3 : [#blue]

foo --> bar
foo -[#red]-> bar1      : [#red]
foo -[#green]-> bar2     : [#green]
foo -[#blue]-> bar3      : [#blue]
'foo -[#blue;#yellow;#green]-> bar4
@enduml
  
```



### 3.34.3 Line thickness

```

@startuml
title Bracketed line thickness
class foo
class bar
bar1 : [thickness=1]
bar2 : [thickness=2]
bar3 : [thickness=4]
bar4 : [thickness=8]
bar5 : [thickness=16]

foo --> bar      :
foo -[thickness=1]-> bar1   : [1]
foo -[thickness=2]-> bar2   : [2]
foo -[thickness=4]-> bar3   : [4]
  
```

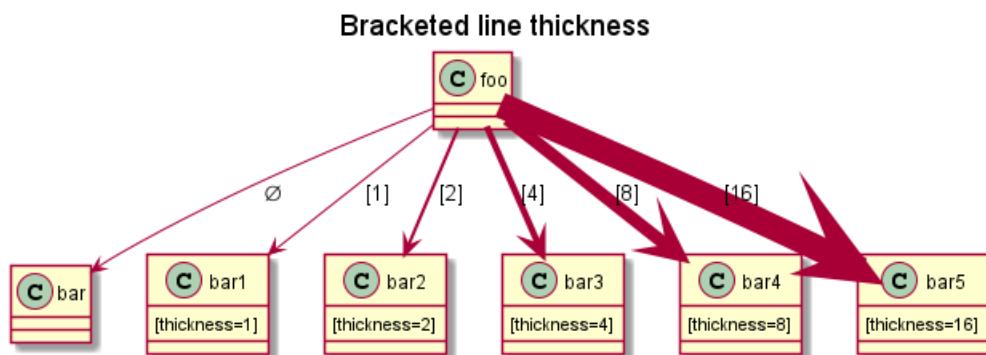


```

foo -[thickness=8]-> bar4 : [8]
foo -[thickness=16]-> bar5 : [16]

@enduml

```



[Ref. QA-4949]

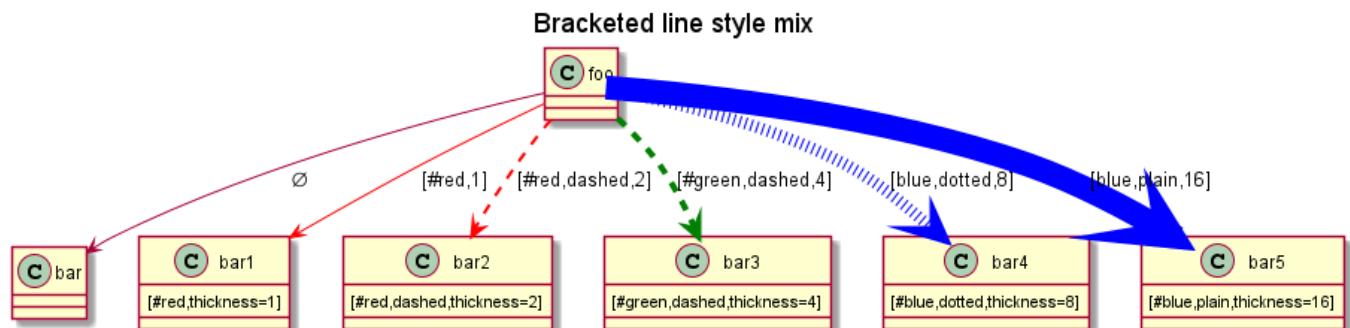
### 3.34.4 Mix

```

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

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]
@enduml

```



### 3.35 Change relation (linking or arrow) color and style (inline style)

You can change the color or style of individual relation or arrows using the inline following notation:

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

```

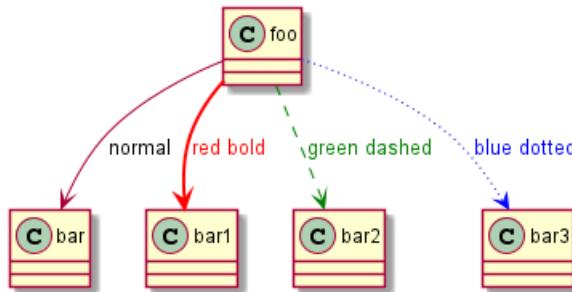
@startuml
class foo
foo --> bar : normal
  
```



```

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

```



*[See similar feature on deployment]*

### 3.36 Change class color and style (inline style)

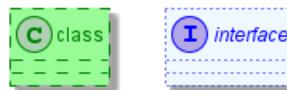
You can change the color or style of individual class using the following notation:

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

```

@startuml
abstract abstract
annotation annotation #pink;line:red;line.bold;text:red
class class      #palegreen;line:green;line.dashed;text:green
interface interface #aliceblue;line:blue;line.dotted;text:blue
@enduml

```



First original example:

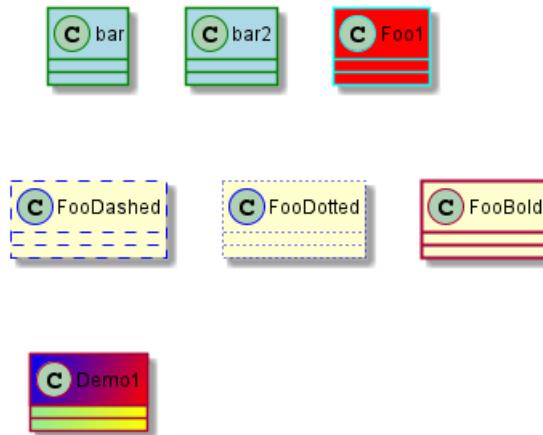
```

@startuml
class bar #line:green;back:lightblue
class bar2 #lightblue;line:green

class Foo1 #back:red;line:00FFFF
class FooDashed #line.dashed:blue
class FooDotted #line.dotted:blue
class FooBold #line.bold
class Demo1 #back:lightgreen|yellow;header:blue/red
@enduml

```





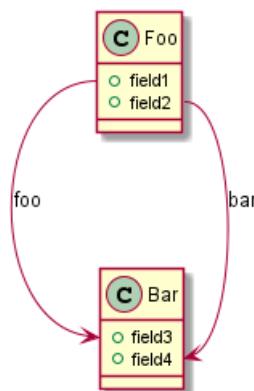
[Ref. QA-3770]

### 3.37 Arrows from/to class members

```
@startuml
class Foo {
+ field1
+ field2
}

class Bar {
+ field3
+ field4
}

Foo::field1 --> Bar::field3 : foo
Foo::field2 --> Bar::field4 : bar
@enduml
```



[Ref. QA-3636]

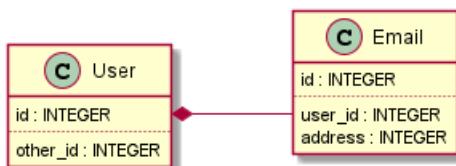
```
@startuml
left to right direction

class User {
    id : INTEGER
    ..
    other_id : INTEGER
}

class Email {
```



```
id : INTEGER  
..  
user_id : INTEGER  
address : INTEGER  
}  
  
User::id *-- Email::user_id  
@enduml
```



[Ref. QA-5261]

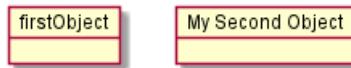


## 4 对象图

### 4.1 对象的定义

使用关键字 `object` 定义实例。

```
@startuml
object firstObject
object "My Second Object" as o2
@enduml
```



### 4.2 对象之间的关系

对象之间的关系用如下符号定义：

Type	Symbol	Image
Extension	< --	
Composition	*---	
Aggregation	o--	

也可以用 `..` 来代替 `--` 以使用点线。

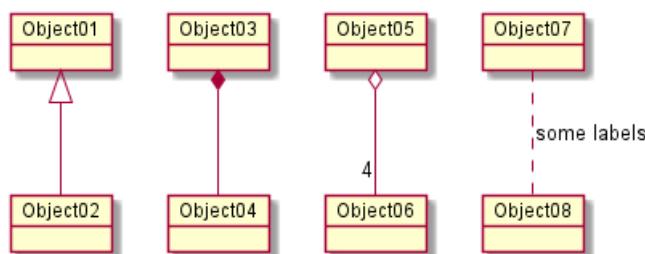
知道了这些规则，就可以画下面的图：

可以用冒号给关系添加标签，标签内容紧跟在冒号之后。

用双引号在关系的两边添加基数。

```
@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 Associations objects

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



```
object o3
```

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



#### 4.4 添加属性

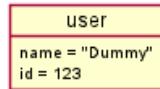
用冒号加属性名的形式声明属性。

```
@startuml
```

```
object user
```

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

```
@enduml
```

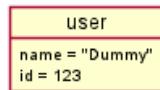


也可以用大括号批量声明属性。

```
@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 **Country => CapitalCity**" as CC {
    UK => London
    USA => Washington
    Germany => Berlin
}
@enduml
```

Map Country => 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
```

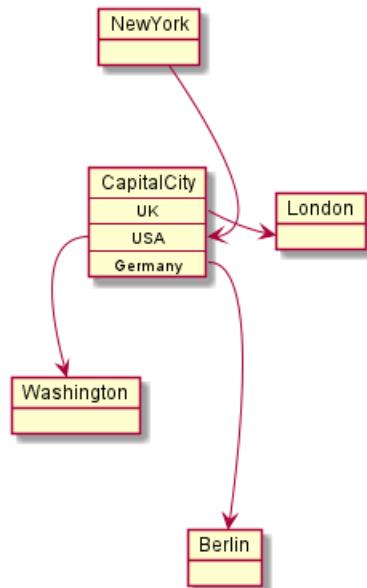


```
@startuml
object London
```

```
object Washington
object Berlin
object NewYork

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

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



[Ref. #307]



## 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。

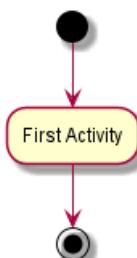
有时，你可能想用 (\*top) 强制开始点位于图示的顶端。

使用--> 绘制箭头。

```
@startuml
```

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

```
@enduml
```



### 5.2 箭头上的标签

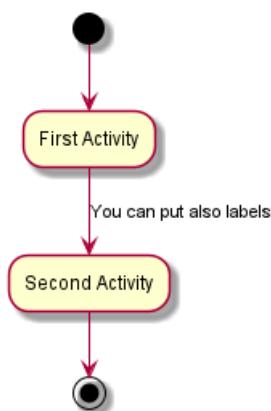
默认情况下，箭头开始于最接近的活动。

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->

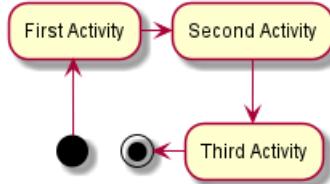


- -up->

```
@startuml
```

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

```
@enduml
```



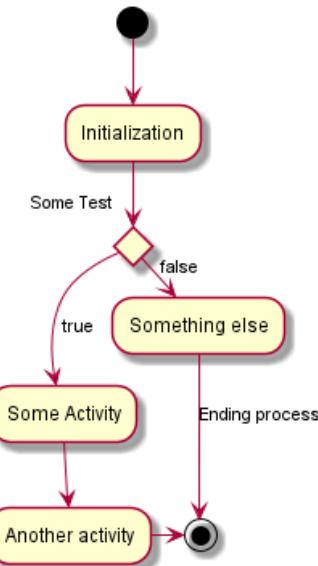
## 5.4 分支

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
```

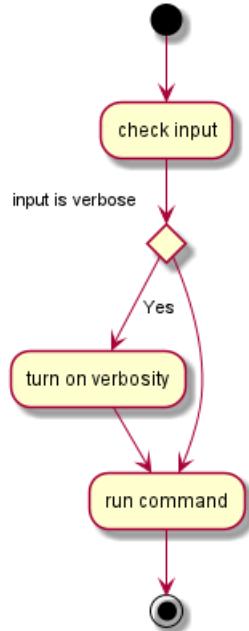


不过，有时你可能需要重复定义同一个活动：

```
@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
```

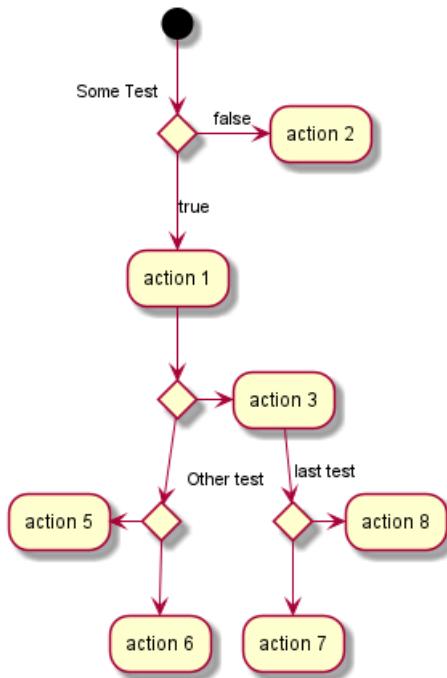


```

-> "action 8"
endif

@enduml

```



## 5.6 同步

你可以使用 === code === 来显示同步条。

```

@startuml

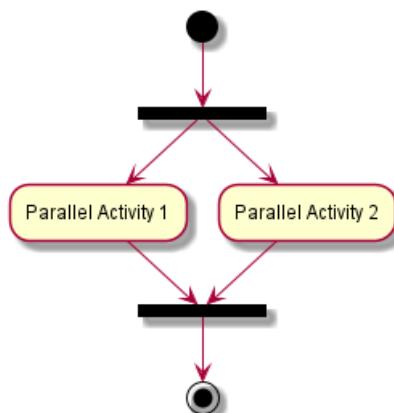
(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

====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。

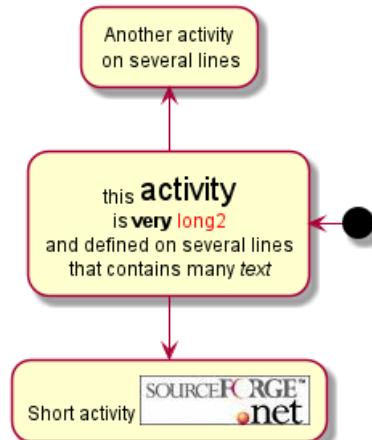
还可以用 `as` 关键字给活动起一个短的别名。这个别名可以在接下来的图示定义中使用。

```
@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。

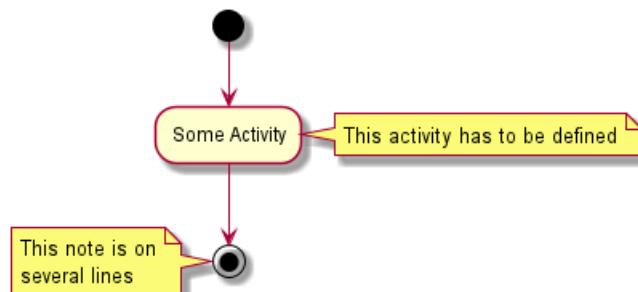
如果想给开始点添加注释，只需把注释的定义放在活动图最开始的地方即可。

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
}
```

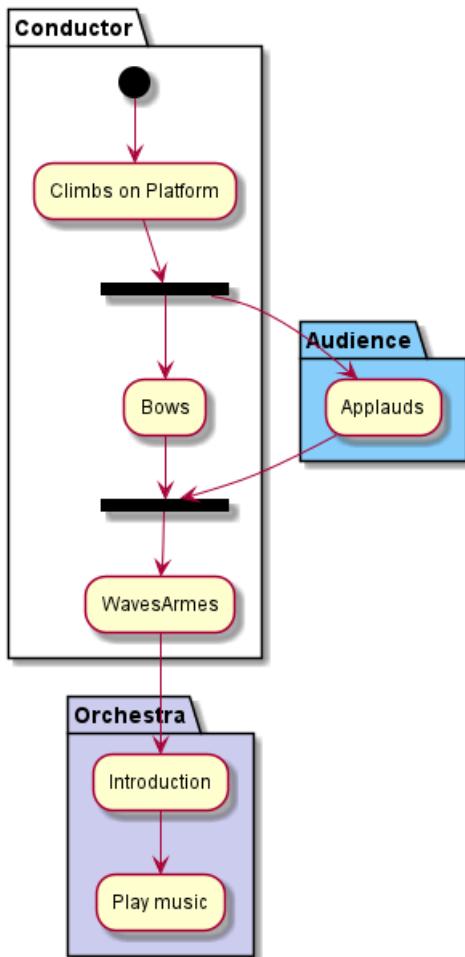
```
partition Audience #LightSkyBlue {
    === S1 === --> Applauds
}
```

```
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`

```

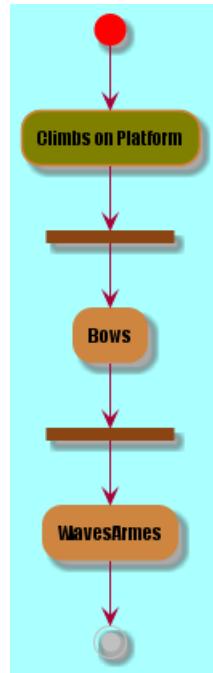
skinparam backgroundColor #AFFFFF
skinparam activity {
    StartColor red
    BarColor SaddleBrown
    EndColor Silver
    BackgroundColor Peru
    BackgroundColor<< Begin >> Olive
    BorderColor Peru
    FontName Impact
}
(*) --> "Climbs on Platform" << Begin >>
--> === S1 ===
--> Bows

```



```
--> === 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

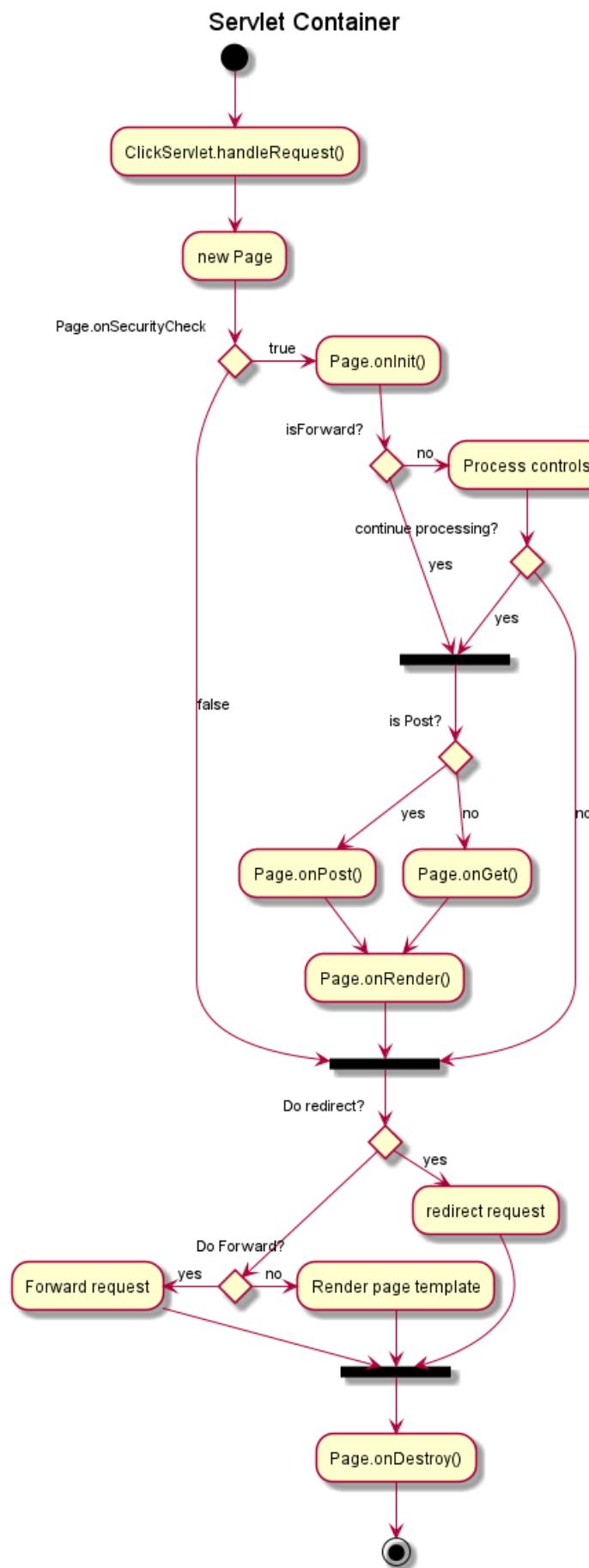
else
-->[false] ===REDIRECT_CHECK===
endif

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





## 6 活动图 (新语法)

当前活动图 (activity diagram) 的语法有诸多限制和缺点，比如代码难以维护。

所以从 V7947 开始提出一种全新的、更好的语法格式和软件实现供用户使用 (beta 版)。

就像序列图一样，新的软件实现的另一个优点是它不再依赖于 Graphviz。

新的语法将会替换旧的语法。然而考虑到兼容性，旧的语法仍被能够使用以确保向前兼容。

但是我们鼓励用户使用新的语法格式。

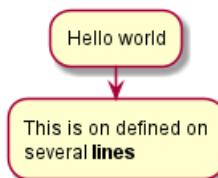
### 6.1 简单活动图

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。

活动默认安装它们定义的顺序就行连接。

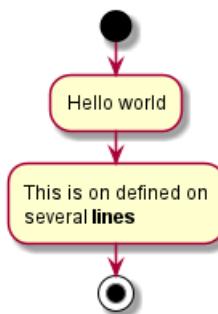
```
@startuml
:Hello world;
:This is on defined on
several **lines**;
@enduml
```



### 6.2 开始/结束

你可以使用关键字 `start` 和 `stop` 表示图示的开始和结束。

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

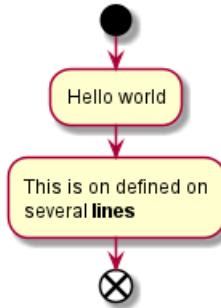


也可以使用 `end` 关键字。

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



```
end
@enduml
```



### 6.3 条件语句

在图示中可以使用关键字 `if`, `then` 和 `else` 设置分支测试。标注文字则放在括号中。

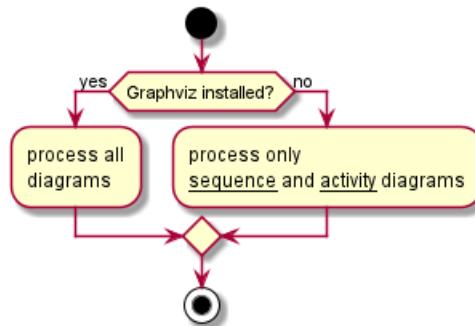
```
@startuml
```

```
start
```

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

```
stop
```

```
@enduml
```

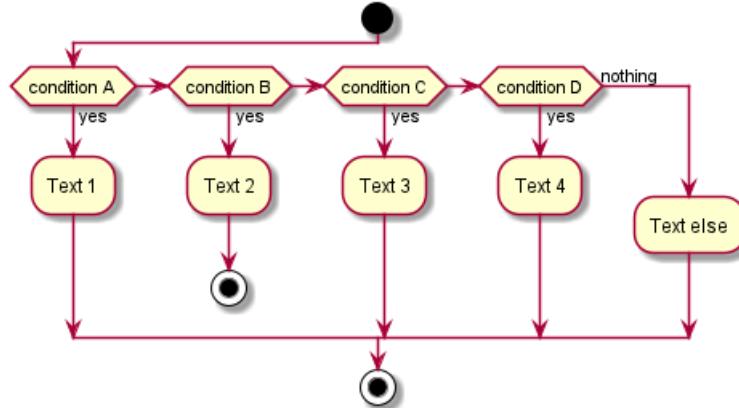


也可以使用关键字 `elseif` 设置多个分支测试。

```
@startuml
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;
```



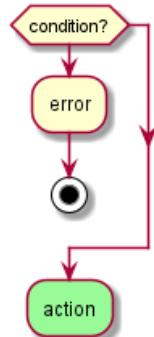
```
endif
stop
@enduml
```



## 6.4 Conditional with stop on an action [kill, detach]

You can stop action on a if loop.

```
@startuml
if (condition?) then
  :error;
  stop
endif
#palegreen:action;
@enduml
```

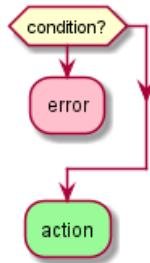


But if you want to stop at an precise action, you can use the `kill` or `detach` keyword:

- kill

```
@startuml
if (condition?) then
  #pink:error;
  kill
endif
#palegreen:action;
@enduml
```



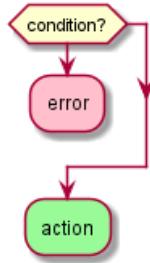


[Ref. QA-265]

- detach

```

@startuml
if (condition?) then
  #pink:error;
  detach
endif
#palegreen:action;
@enduml
  
```



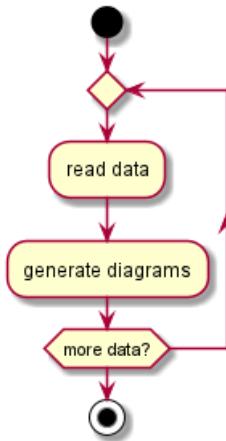
## 6.5 重复循环

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

```

@startuml
start
repeat
  :read data;
  :generate diagrams;
repeat while (more data?)
stop
@enduml
  
```





It is also possible to use a full action as `repeat` target and insert an action in the return path using the `backward` keyword.

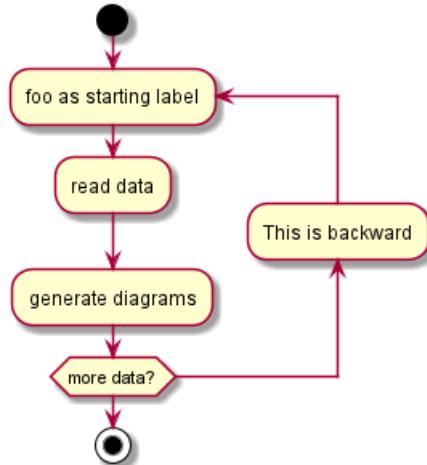
```
@startuml
```

```
start
```

```
repeat :foo as starting label;
    :read data;
    :generate diagrams;
backward:This is backward;
repeat while (more data?)
```

```
stop
```

```
@enduml
```



## 6.6 Break on a repeat loop [break]

You can break after an action on a loop.

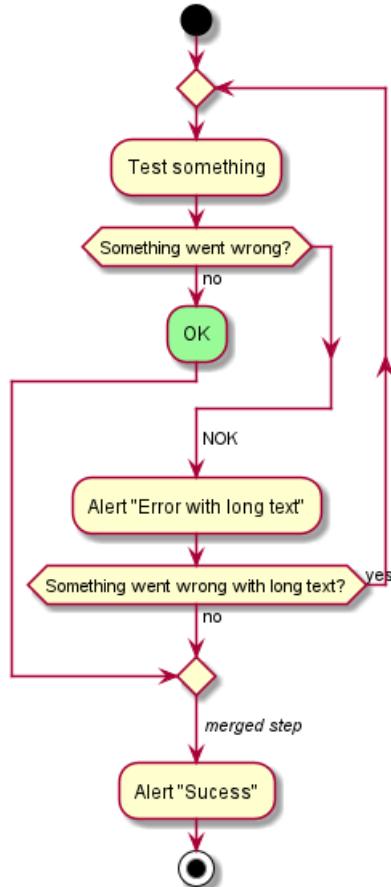
```
@startuml
start
repeat
    :Test something;
    if (Something went wrong?) then (no)
        #palegreen:OK;
        break
```



```

endif
->NOK;
:Alert "Error with long text";
repeat while (Something went wrong with long text?) is (yes) not (no)
->//merged step//;
:Alert "Sucess";
stop
@enduml

```



[Ref. QA-6105]

## 6.7 while 循环

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

```
@startuml
```

start

```

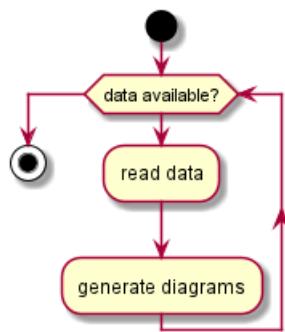
while (data available?)
    :read data;
    :generate diagrams;
endwhile

```

stop

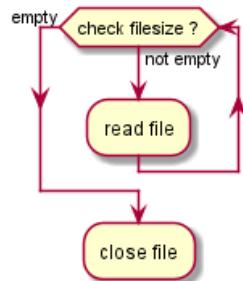
```
@enduml
```





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
```



## 6.8 并行处理

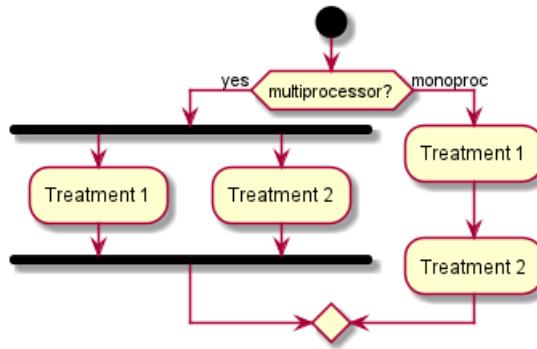
你可以使用关键字 `fork`, `fork again` 和 `end fork` 表示并行处理。

```
@startuml
start

if (multiprocessor?) then (yes)
  fork
    :Treatment 1;
  fork again
    :Treatment 2;
  end fork
else (monoproc)
  :Treatment 1;
  :Treatment 2;
endif

@enduml
```





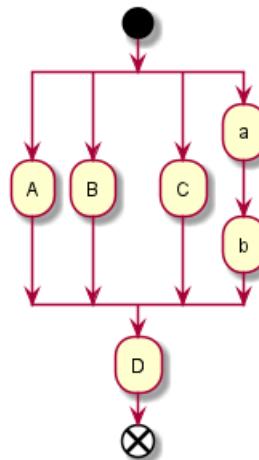
## 6.9 Split processing

### 6.9.1 Split

You can use `split`, `split again` and `end split` keywords to denote split processing.

```

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



### 6.9.2 Input split (multi-start)

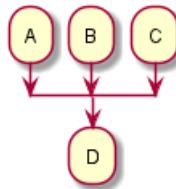
You can use hidden arrows to make an input split (multi-start):

```

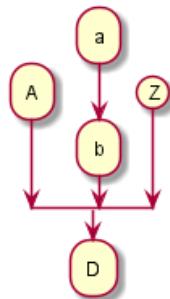
@startuml
split
  -[hidden]->
  :A;
split again
  -[hidden]->
  
```



```
: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
```



[Ref. QA-8662]

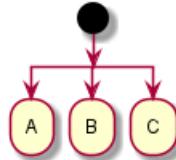
### 6.9.3 Output split (multi-end)

You can use `kill` or `detach` to make an output split (multi-end):

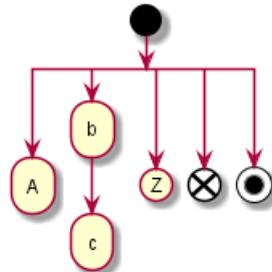
```
@startuml
start
split
:A;
kill
split again
:B;
detach
split again
:C;
kill
```



```
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
```



## 6.10 注释

文本格式支持 creole wiki 语法。

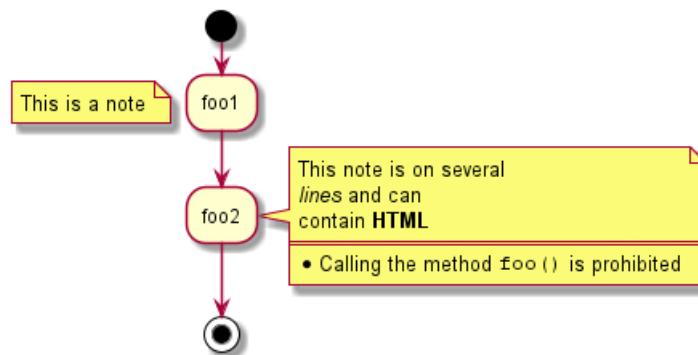
A note can be floating, using `floating` keyword.

```
@startuml

start
:foo1;
floating note left: This is a note
:foo2;
note right
  This note is on several
  //lines// and can
  contain <b>HTML</b>
=====
  * Calling the method ""foo()"" is prohibited
end note
stop
```



@enduml



## 6.11 颜色

你可以为活动 (activity) 指定一种颜色。

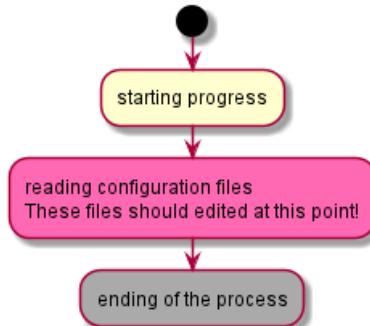
@startuml

```

start
:starting progress;
#HotPink:reading configuration files
These files should edited at this point!;
#AAAAAA:ending of the process;

```

@enduml



## 6.12 Lines without arrows

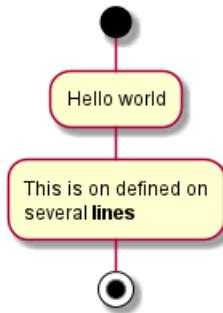
You can use `skinparam ArrowHeadColor none` in order to connect activities using lines only, without arrows.

```

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

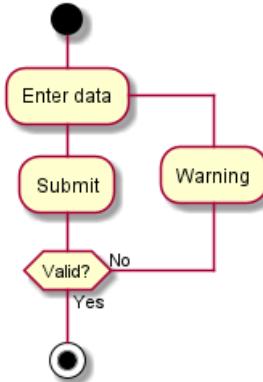
```





```

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



## 6.13 箭头

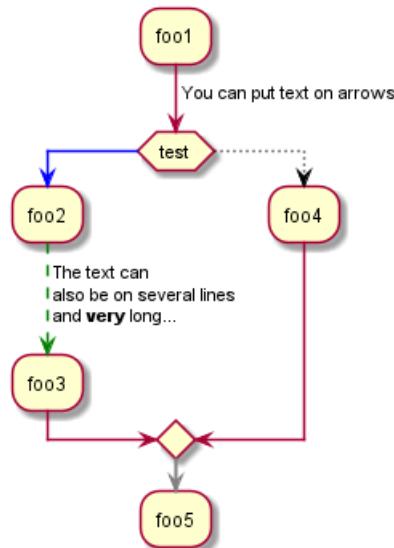
使用-> 标记，你可以给箭头添加文字或者修改箭头颜色。

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
  
```

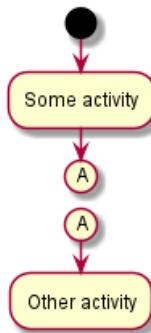




## 6.14 连接器 (Connector)

你可以使用括号定义连接器。

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

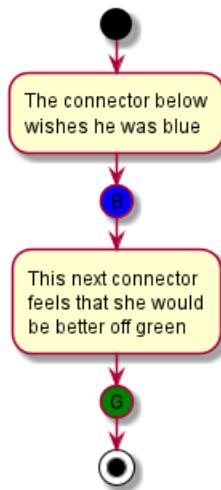


## 6.15 Color on connector

You can add color on connector.

```
@startuml
start
:The connector below
wishes he was blue;
#blue:(B)
:This next connector
feels that she would
be better off green;
#green:(G)
stop
@enduml
```





[Ref. QA-10077]

## 6.16 组合 (grouping)

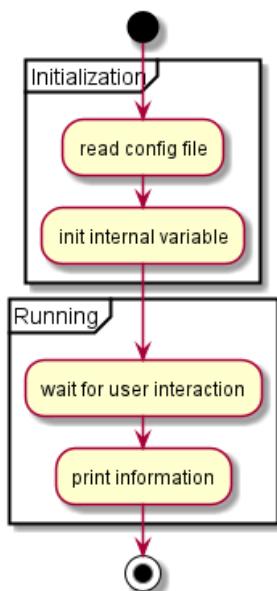
通过定义分区 (partition)，你可以把多个活动组合 (group) 在一起。

```

@startuml
start
partition Initialization {
    :read config file;
    :init internal variable;
}
partition Running {
    :wait for user interaction;
    :print information;
}

stop
@enduml

```

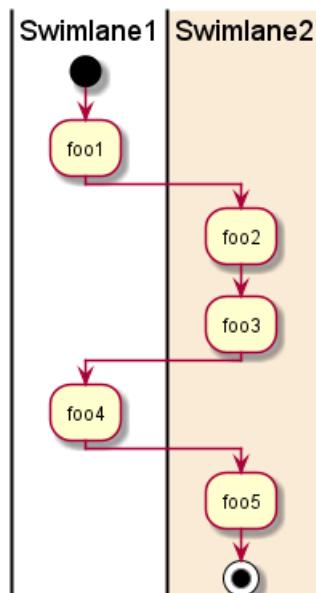


### 6.17 泳道 (Swimlanes)

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
```



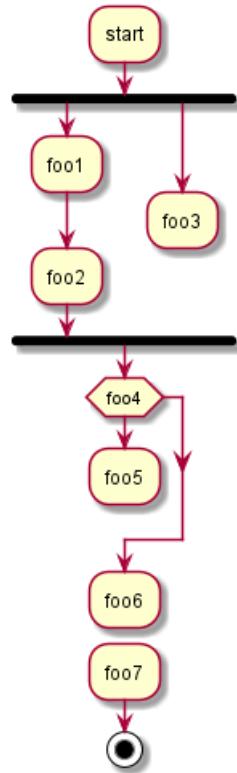
### 6.18 分离 (detach)

可以使用关键字 `detach` 移除箭头。

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



```
stop
@enduml
```



## 6.19 特殊领域语言 (SDL)

通过修改活动标签最后的分号分隔符 (;)，可以为活动设置不同的形状。

- |
- <
- >
- /
- ]
- }

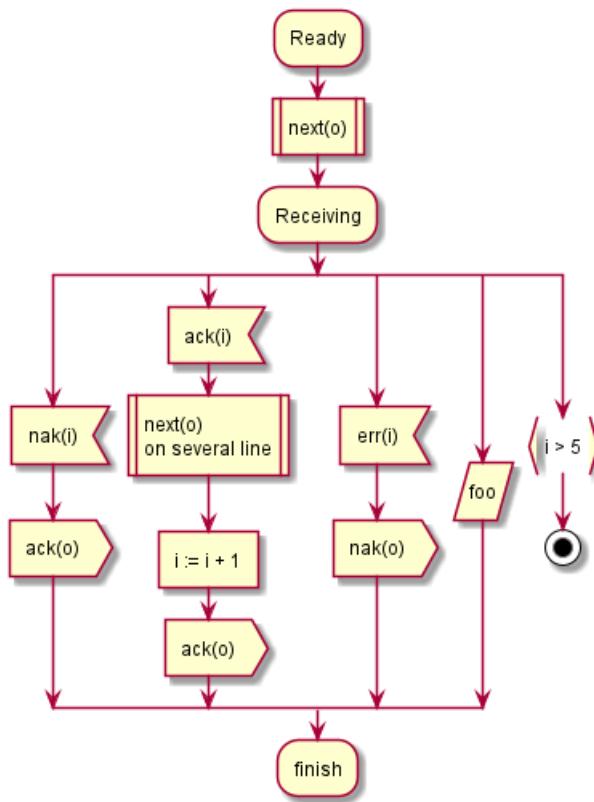
```
@startuml
:Ready;
:next(o)| 
:Receiving;
split
  :nak(i)<
  :ack(o)>
split again
  :ack(i)<
  :next(o)
on several line|
  :i := i + 1]
  :ack(o)>
split again
  :err(i)<
  :nak(o)>
split again
```



```

:foo/
split again
:i > 5}
stop
end split
:finish;
@enduml

```



## 6.20 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

```

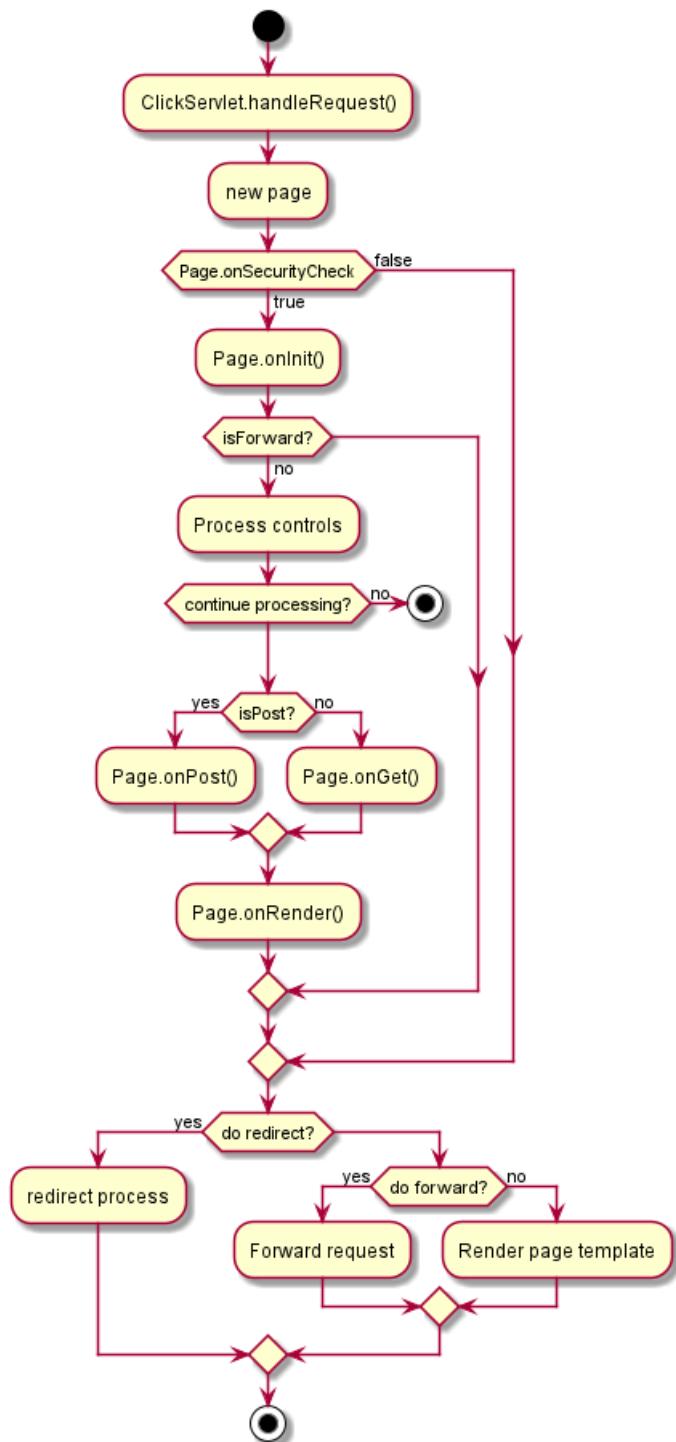


```
if (do redirect?) then (yes)
    :redirect process;
else
    if (do forward?) then (yes)
        :Forward request;
    else (no)
        :Render page template;
    endif
endif

stop

@enduml
```





## 6.21 Condition Style

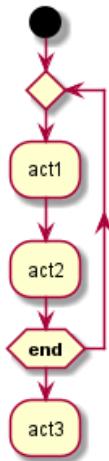
### 6.21.1 Inside style (by default)

```

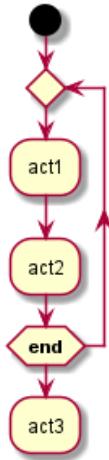
@startuml
skinparam conditionStyle inside
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end</b>)
  :act3;
  
```



@enduml



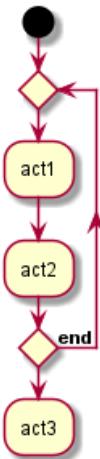
```
@startuml
start
repeat
:act1;
:act2;
repeatwhile (<b>end</b>)
:act3;
@enduml
```



### 6.21.2 Diamond style

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

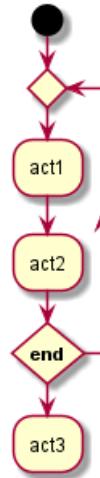




### 6.21.3 InsideDiamond (or *Foo1*) style

```

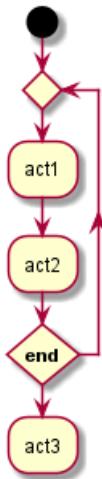
@startuml
skinparam conditionStyle InsideDiamond
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end</b>)
  :act3;
@enduml
  
```



```

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





[Ref. QA-1290 and #400]

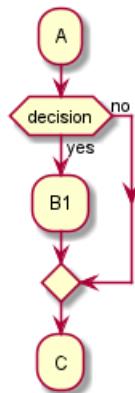
## 6.22 Condition End Style

### 6.22.1 Diamond style (by default)

- With one branch

```

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



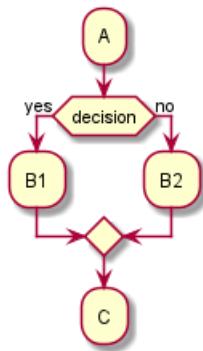
- With two branches (B1, B2)

```

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



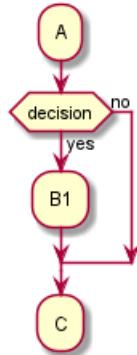
```
@enduml
```



### 6.22.2 Horizontal line (hline) style

- With one branch

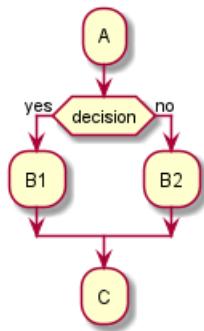
```
@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
  :B1;
else (no)
endif
:C;
@enduml
```



- With two branches (B1, B2)

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





[Ref. QA-4015]



## 7 组件图

我们来看几个例子。

### 7.1 组件

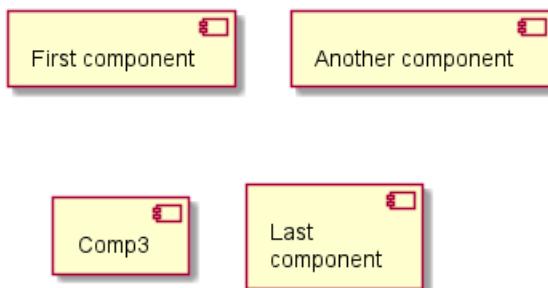
组件必须用中括号括起来。

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
```



### 7.2 接口

接口可以使用 `()` 来定义 (因为这个看起来像个圆)。

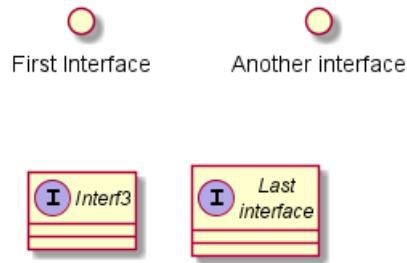
还可以使用关键字 `interface` 关键字来定义接口。并且还可以使用关键字 `as` 定义一个别名。这个别名可以在稍后定义关系的时候使用。

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\ncinterface" as Interf4
```

```
@enduml
```

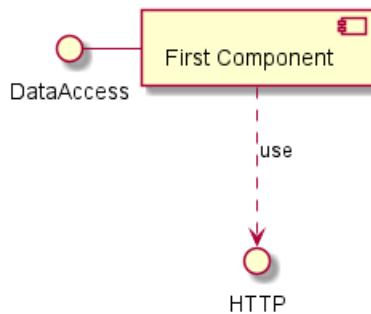


### 7.3 基础的示例

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。

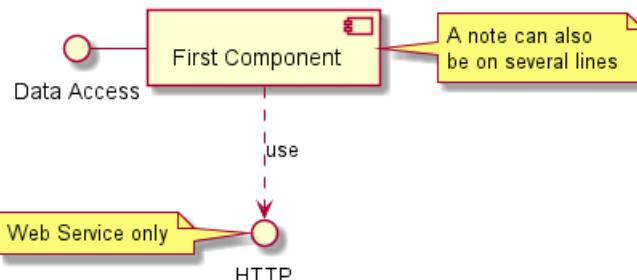
```
@startuml
interface "Data Access" as DA

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
}

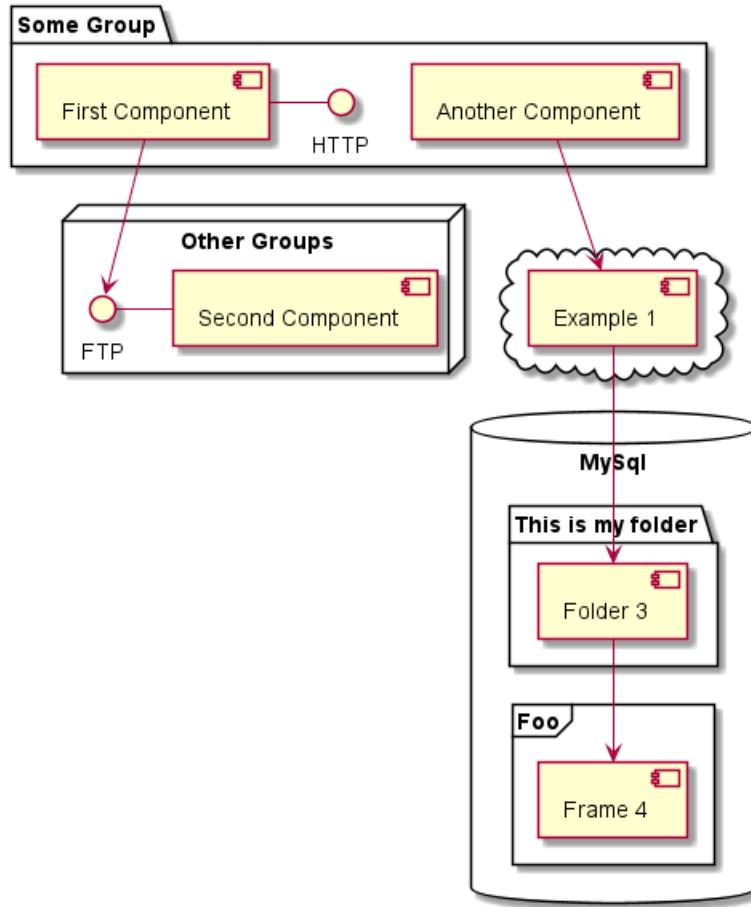
cloud {
    [Example 1]
}

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
```

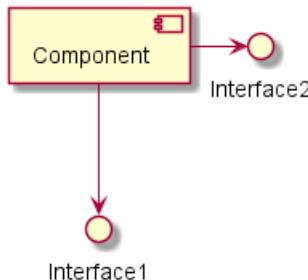




## 7.6 改变箭头方向

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：

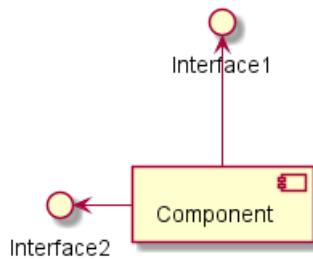
```
@startuml
[Component] --> Interface1
[Component] -> Interface2
@enduml
```



也可以使用反向连接：

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

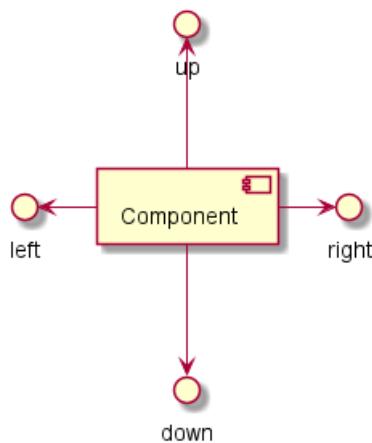




还可以使用关键字 `left`, `right`, `up` or `down` 改变箭头方向。

```

@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 Use UML2 notation

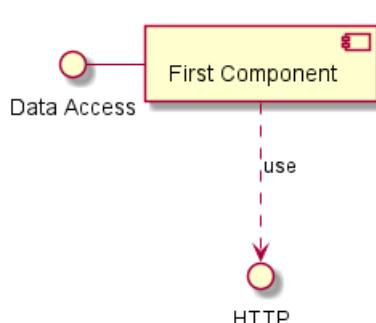
By default (*from v1.2020.13-14*), UML2 notation is used.

```
@startuml
```

```

interface "Data Access" as DA
DA - [First Component]
[First Component] ..> HTTP : use
  
```

```
@enduml
```



## 7.8 使用 UML1 标记符

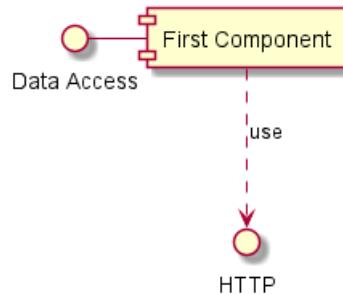
命令 `skinparam componentStyle uml1` 可以切换到 UML1 标记符。

```
@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)

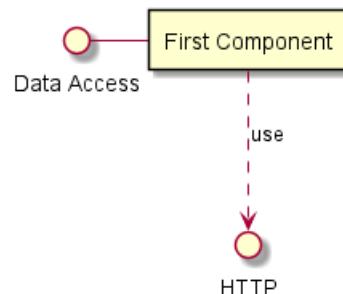
The `skinparam componentStyle rectangle` command is used to switch to rectangle notation (*without any UML notation*).

```
@startuml
skinparam componentStyle rectangle
```

```
interface "Data Access" as DA
```

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

```
@enduml
```

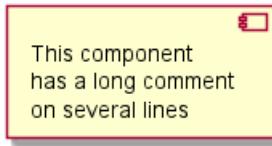


## 7.10 长描述

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
```

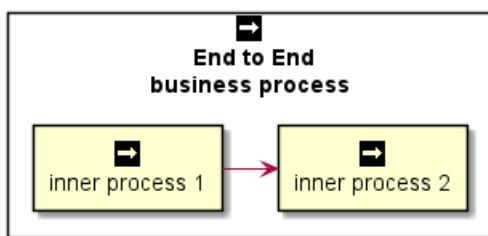


## 7.12 在定型组件中使用精灵图

你可以在定型组件中使用精灵图 (sprite)。

```
@startuml
sprite $businessProcess [16x16/16] {
    FFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFF
    FFFFFFFFFFFFOFFFF
    FFFFFFFFFFFFOOOFFF
    FF000000000000FF
    FF0000000000000FF
    FF0000000000000FF
    FFFFFFFFFFFFOOFFFF
    FFFFFFFFFFFFOFFFFF
    FFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFF
}
```

```
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
```

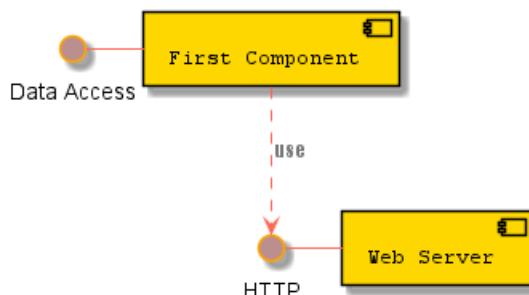
```
skinparam interface {
    backgroundColor RosyBrown
    borderColor orange
}

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

() "Data Access" as DA

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server] << Apache >>
```

```
@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
```



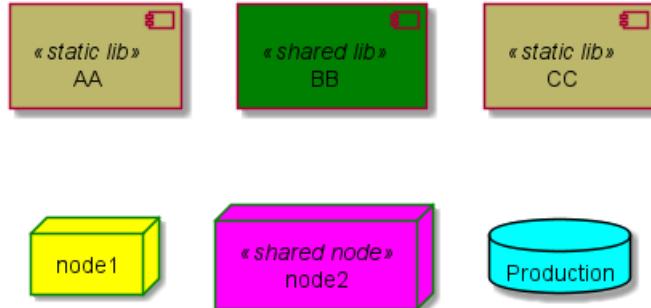
```

    backgroundColor<<shared lib>> Green
}

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

@enduml

```



## 7.14 Specific SkinParameter

### 7.14.1 componentStyle

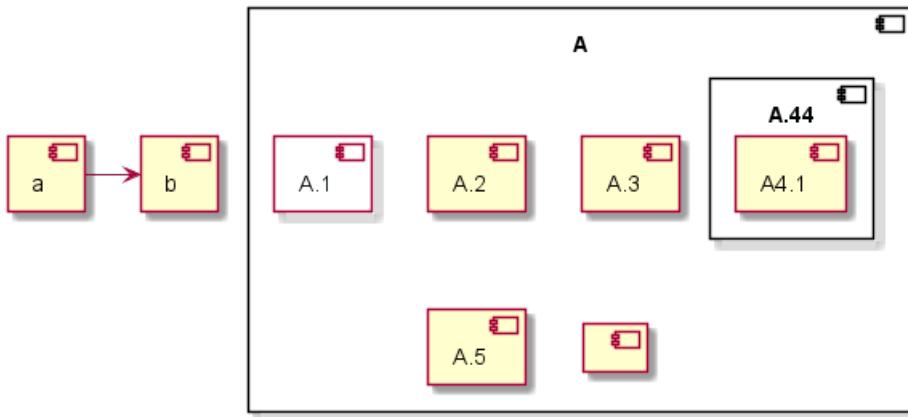
- By default (or with `skinparam componentStyle uml2`), you have an icon for component

```

@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

```

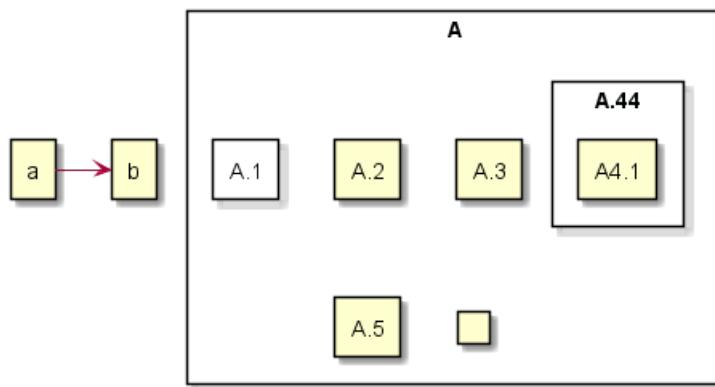




- If you want to suppress it, and to have only the rectangle, you can use `skinparam componentStyle rectangle`

```

@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 Hide or Remove unlinked component

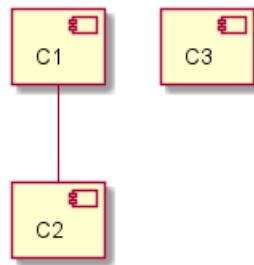
By default, all components are displayed:

```

@startuml
component C1
component C2
component C3
  
```



```
C1 -- C2
@enduml
```



But you can:

- hide @unlinked components:

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

```
hide @unlinked
@enduml
```



- or remove @unlinked components:

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

```
remove @unlinked
@enduml
```



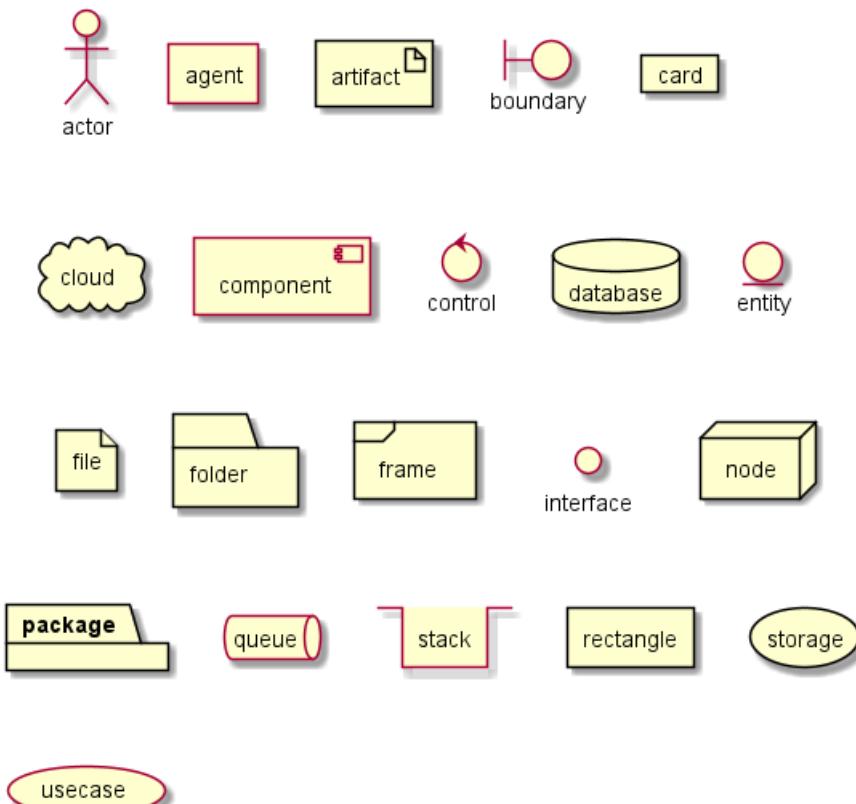
[Ref. QA-11052]



## 8 部署图

### 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</b>
```



----  
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 Declaring element (using short form)

We can declare element using some short forms.

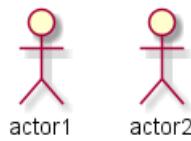
Long form Keyword	Short form Keyword	Long form example	Short form example	Ref.
actor	: a :	actor actor1	:actor2:	Actors
component	[ c ]	component component1	[component2]	Components
interface	() i	interface interface1	() "interface2"	Interfaces
usecase	( u )	usecase usecase1	(usecase2)	Usecases

### 8.2.1 Actor

```
@startuml
```

```
actor actor1
:actor2:
```

```
@enduml
```



**NB:** There is an old syntax for actor with guillemet which is now deprecated and will be removed some days. Please do not use in your diagram.

### 8.2.2 Component

```
@startuml
```

```
component component1
[component2]
```

```
@enduml
```



### 8.2.3 Interface

```
@startuml
```

```
interface interface1
() "interface2"

label "//interface example//"
@enduml
```



*interface example*

### 8.2.4 Usecase

```
@startuml
```

```
usecase usecase1  
(usecase2)
```

```
@enduml
```



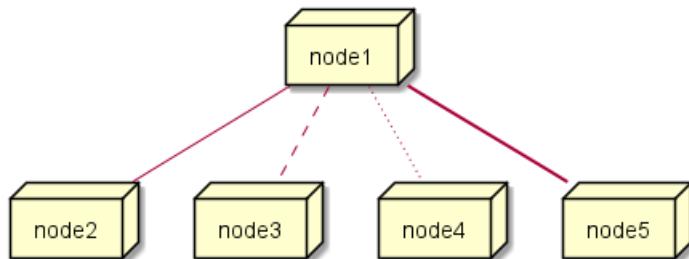
### 8.3 链接

您可以在元素之间创建简单链接:

```
@startuml
```

```
node node1  
node node2  
node node3  
node node4  
node node5  
node1 -- node2  
node1 .. node3  
node1 ~~ node4  
node1 == node5
```

```
@enduml
```

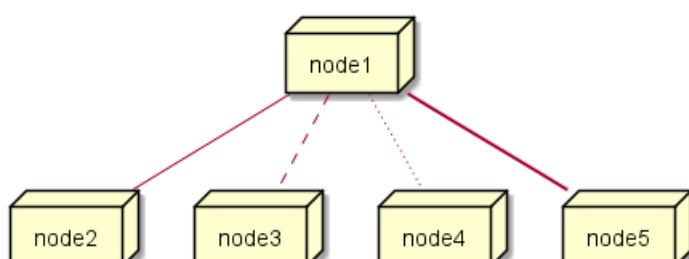


您可以在元素之间创建简单链接:

```
@startuml
```

```
node node1  
node node2  
node node3  
node node4  
node node5  
node1 -- node2  
node1 .. node3  
node1 ~~ node4  
node1 == node5
```

```
@enduml
```

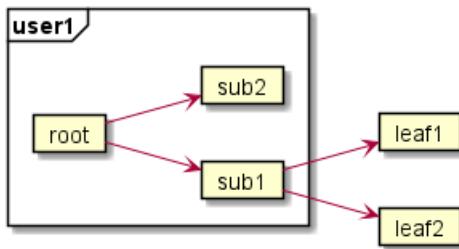


横向的链接:

```
@startuml
left to right direction
frame user1{
card root
card sub1
card sub2
}

card leaf1
card leaf2

root-->sub1
root-->sub2
sub1-->leaf1
sub1-->leaf2
@enduml
```

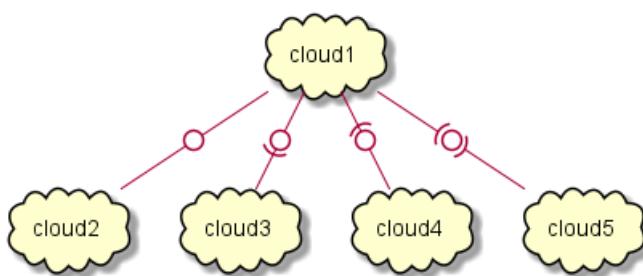


您还可以使用以下类型:

```
@startuml

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

@enduml
```



See all type on [Appendix](#).

## 8.4 Bracketed arrow style

*Similar as Bracketed class relations (linking or arrow) style*

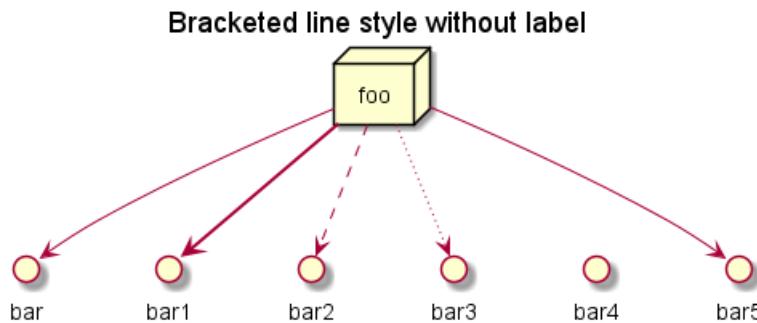


### 8.4.1 Line style

It's also possible to have explicitly bold, dashed, dotted, hidden or plain arrows:

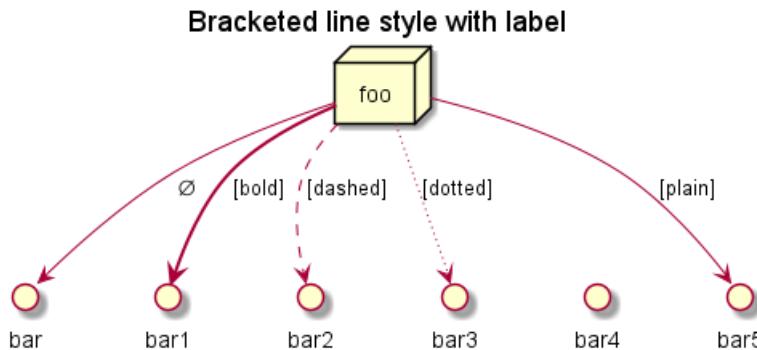
- without label

```
@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
```



- with label

```
@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
```



[Adapted from QA-4181]

### 8.4.2 Line color

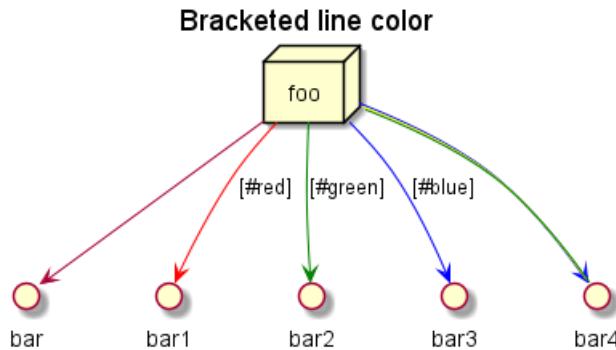
```
@startuml
title Bracketed line color
node foo
foo --> bar
```



```

foo -[#red]-> bar1      : [#red]
foo -[#green]-> bar2    : [#green]
foo -[#blue]-> bar3     : [#blue]
foo -[#blue;#yellow;#green]-> bar4
@enduml

```

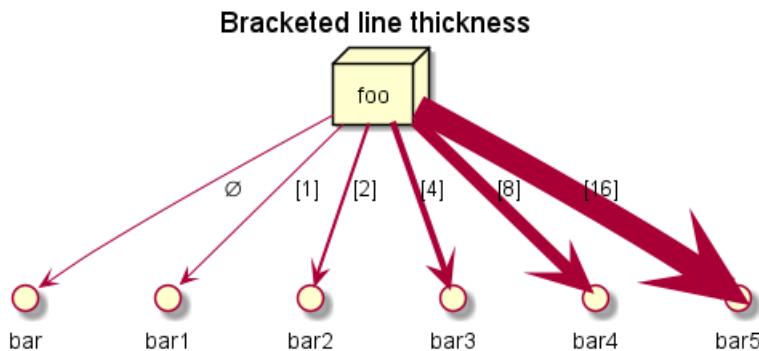


#### 8.4.3 Line thickness

```

@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

```



[Adapted from QA-4949]

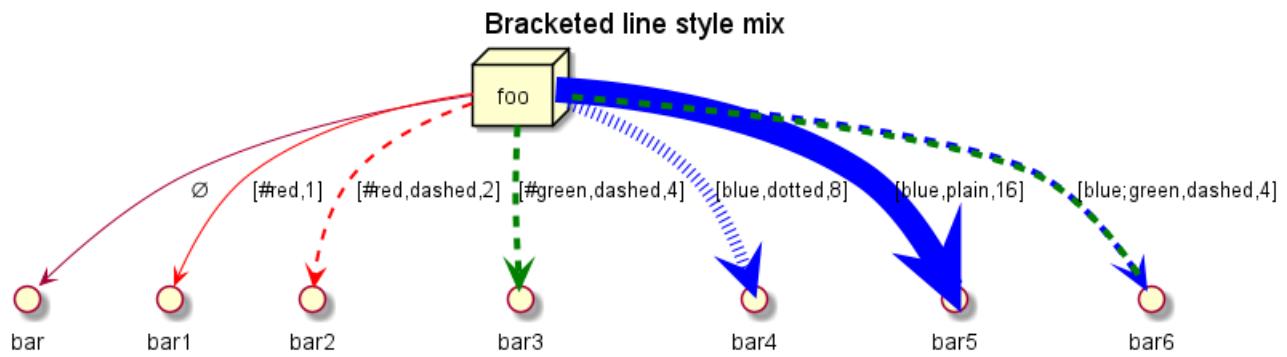
#### 8.4.4 Mix

```

@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

```



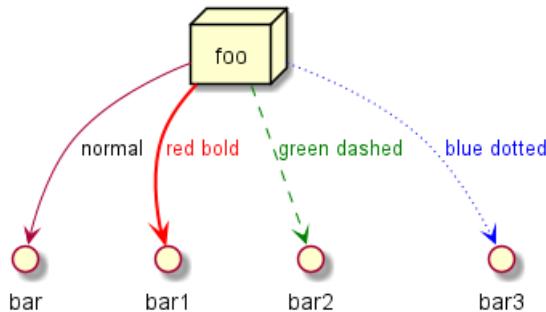


## 8.5 Change arrow color and style (inline style)

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

- `#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
```



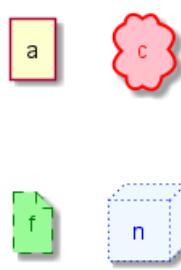
[Ref. QA-3770 and QA-3816] [See similar feature on class diagram]

## 8.6 Change element color and style (inline style)

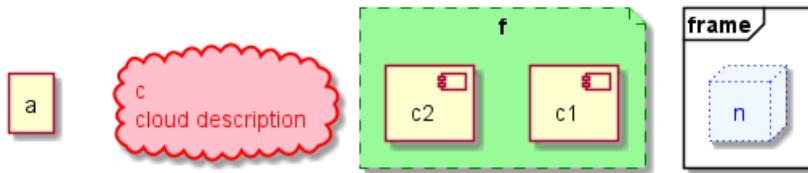
You can change the color or style of individual element using the following notation:

- `#*[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 Nestable elements

Here are the nestable elements:

```
@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 包装

```
@startuml
artifact Foo1 {
    folder Foo2
}

folder Foo3 {
    artifact Foo4
}

frame Foo5 {
    database Foo6
}

@enduml
```



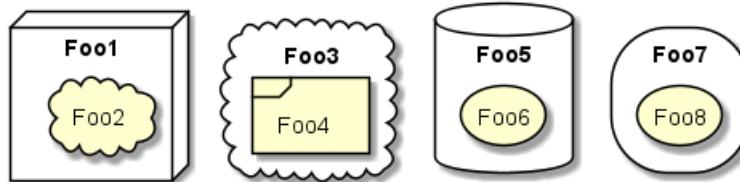
```
@startuml
node Foo1 {
    cloud Foo2
}

cloud Foo3 {
    frame Foo4
}

database Foo5 {
    storage Foo6
}

storage Foo7 {
    storage Foo8
}

@enduml
```



## 8.9 Alias

### 8.9.1 Simple alias with as

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

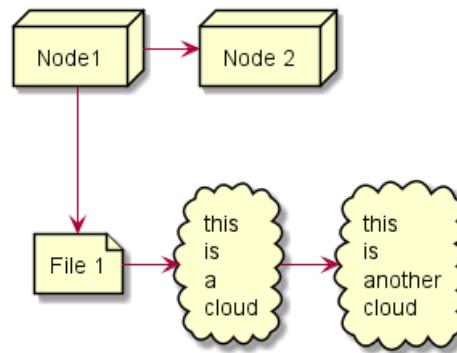


```

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

```



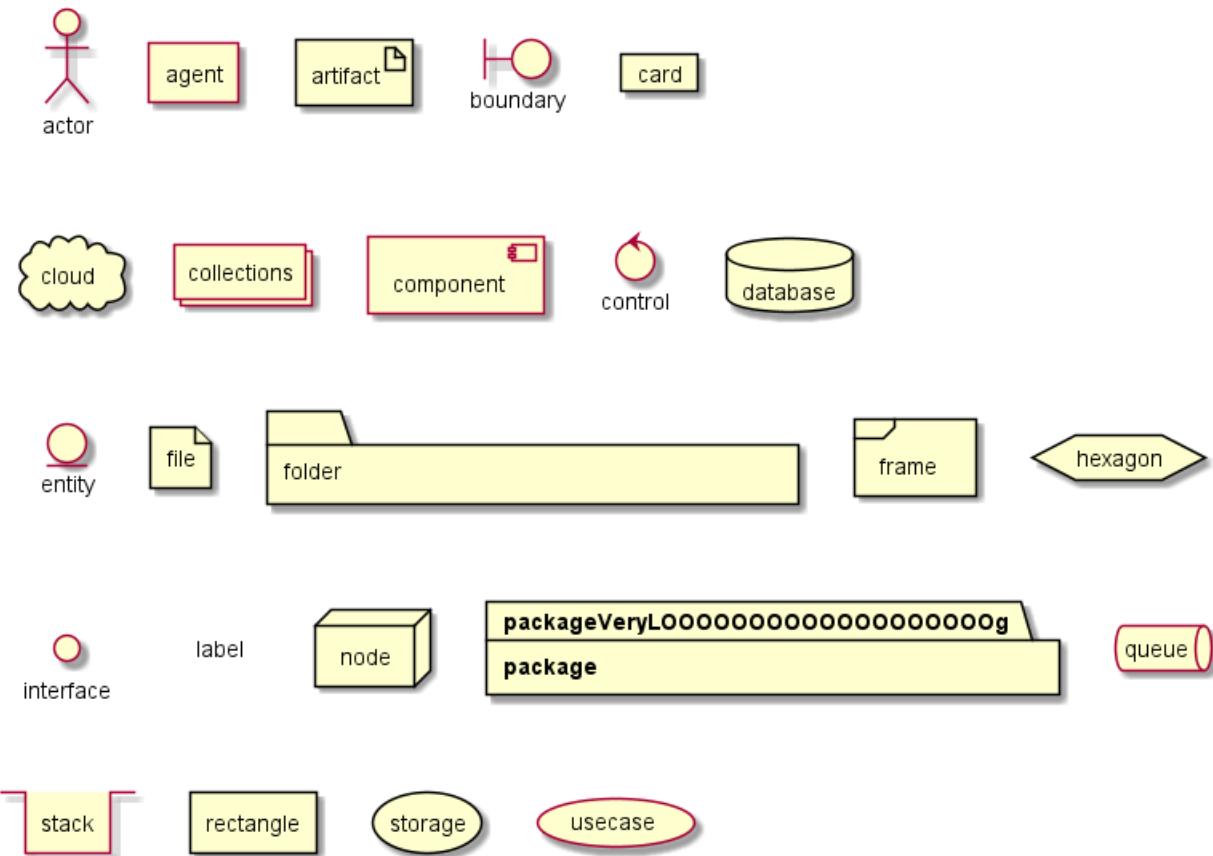
### 8.9.2 Examples of long alias

```

@startuml
actor      "actor"      as actorVeryL00000000000000000000g
agent      "agent"      as agentVeryL00000000000000000000g
artifact   "artifact"   as artifactVeryL00000000000000000000g
boundary   "boundary"   as boundaryVeryL00000000000000000000g
card       "card"       as cardVeryL00000000000000000000g
cloud      "cloud"      as cloudVeryL00000000000000000000g
collections "collections" as collectionsVeryL00000000000000000000g
component   "component"  as componentVeryL00000000000000000000g
control     "control"    as controlVeryL00000000000000000000g
database   "database"   as databaseVeryL00000000000000000000g
entity      "entity"     as entityVeryL00000000000000000000g
file        "file"       as fileVeryL00000000000000000000g
folder      "folder"     as folderVeryL00000000000000000000g
frame       "frame"      as frameVeryL00000000000000000000g
hexagon     "hexagon"    as hexagonVeryL00000000000000000000g
interface   "interface"  as interfaceVeryL00000000000000000000g
label       "label"      as labelVeryL00000000000000000000g
node        "node"       as nodeVeryL00000000000000000000g
package     "package"    as packageVeryL00000000000000000000g
queue       "queue"      as queueVeryL00000000000000000000g
stack       "stack"      as stackVeryL00000000000000000000g
rectangle   "rectangle"  as rectangleVeryL00000000000000000000g
storage     "storage"   as storageVeryL00000000000000000000g
usecase     "usecase"   as usecaseVeryL00000000000000000000g
@enduml

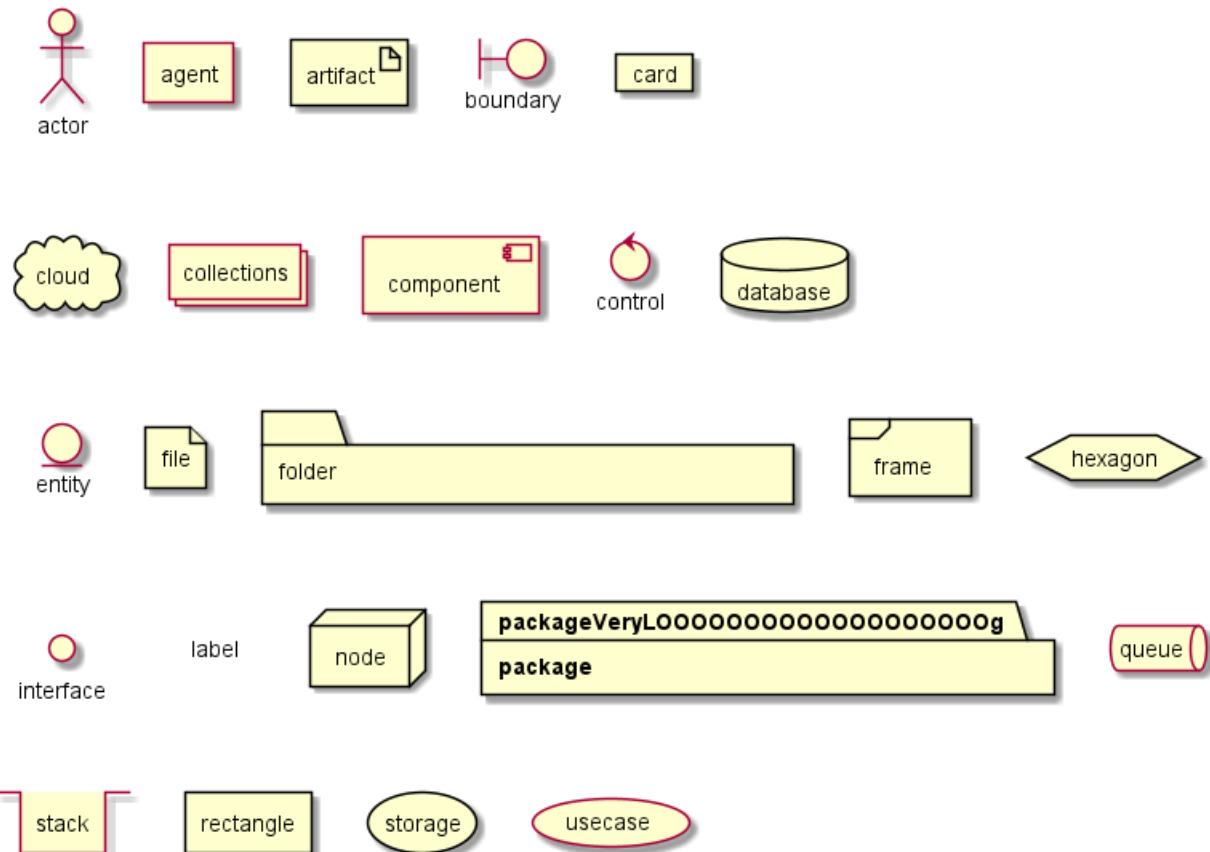
```





```
@startuml
actor      actorVeryLooooooooooooo0g          as "actor"
agent      agentVeryLooooooooooooo0g           as "agent"
artifact    artifactVeryLooooooooooooo0g        as "artifact"
boundary   boundaryVeryLooooooooooooo0g        as "boundary"
card       cardVeryLooooooooooooo0g            as "card"
cloud      cloudVeryLooooooooooooo0g           as "cloud"
collections collectionsVeryLooooooooooooo0g   as "collections"
component   componentVeryLooooooooooooo0g       as "component"
control     controlVeryLooooooooooooo0g         as "control"
database   databaseVeryLooooooooooooo0g        as "database"
entity      entityVeryLooooooooooooo0g          as "entity"
file        fileVeryLooooooooooooo0g            as "file"
folder      folderVeryLooooooooooooo0g          as "folder"
frame       frameVeryLooooooooooooo0g           as "frame"
hexagon    hexagonVeryLooooooooooooo0g          as "hexagon"
interface  interfaceVeryLooooooooooooo0g        as "interface"
label      labelVeryLooooooooooooo0g            as "label"
node       nodeVeryLooooooooooooo0g             as "node"
package    packageVeryLooooooooooooo0g          as "package"
queue      queueVeryLooooooooooooo0g            as "queue"
stack      stackVeryLooooooooooooo0g             as "stack"
rectangle  rectangleVeryLooooooooooooo0g         as "rectangle"
storage    storageVeryLooooooooooooo0g           as "storage"
usecase   usecaseVeryLooooooooooooo0g           as "usecase"
@enduml
```



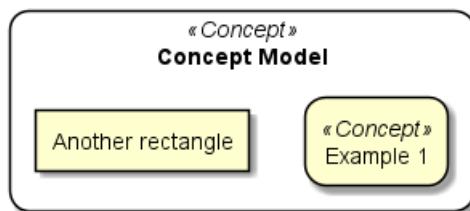


[Ref. QA-12082]

## 8.10 圆角

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

rectangle "Concept Model" <<Concept>> {
    rectangle "Example 1" <<Concept>> as ex1
    rectangle "Another rectangle"
}
@enduml
```



## 8.11 Specific SkinParameter

### 8.11.1 roundCorner

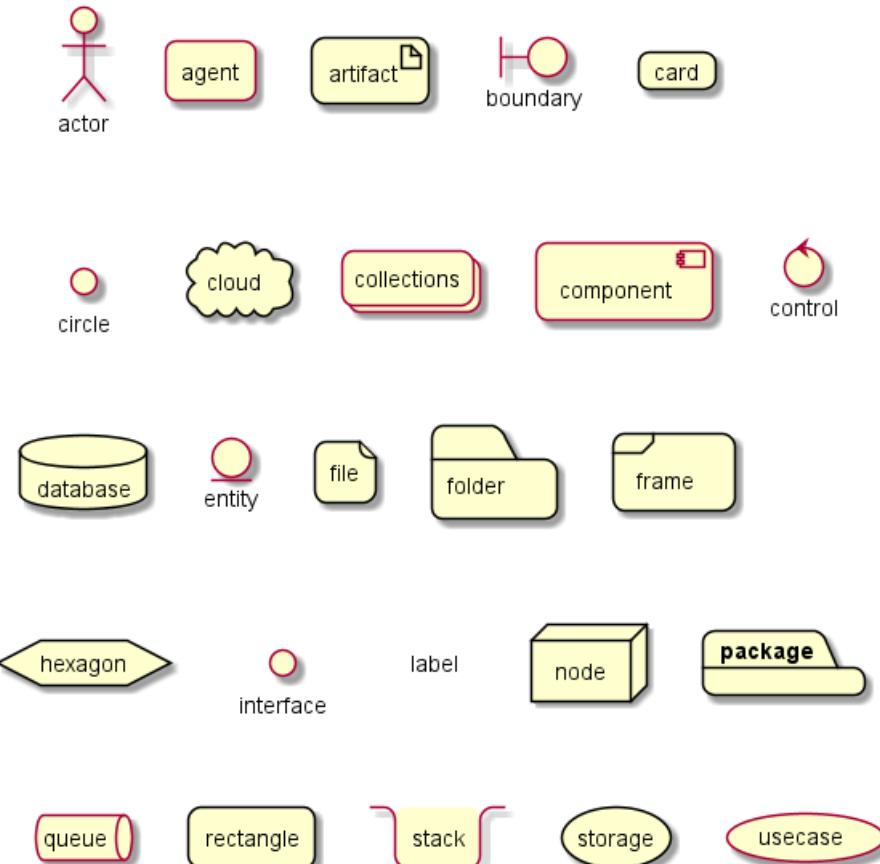
```
@startuml
skinparam roundCorner 15
actor actor
agent agent
```



```

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
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
@enduml

```



[Ref. QA-5299, QA-6915, QA-11943]

## 8.12 Appendix: All type of arrow line

```

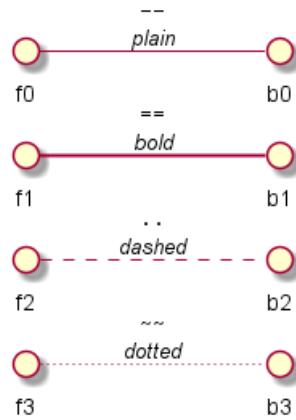
@startuml
left to right direction

```



```
skinparam nodesep 5
```

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



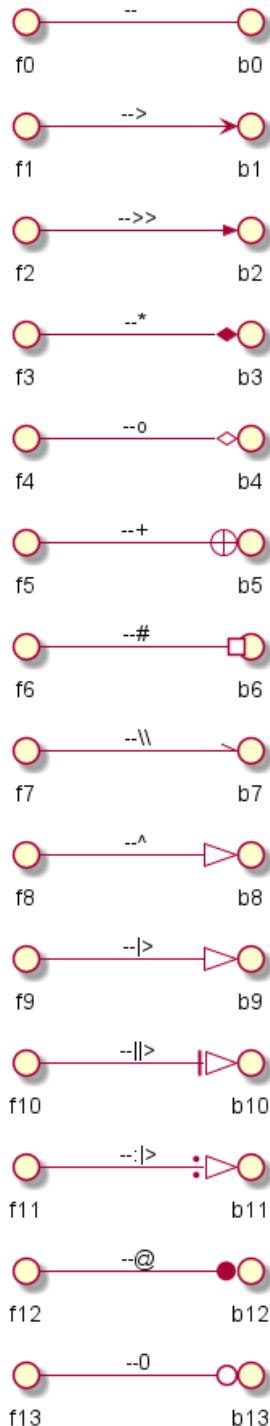
## 8.13 Appendix: All type of arrow head or '0' arrow

### 8.13.1 Type of arrow head

```
@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
```





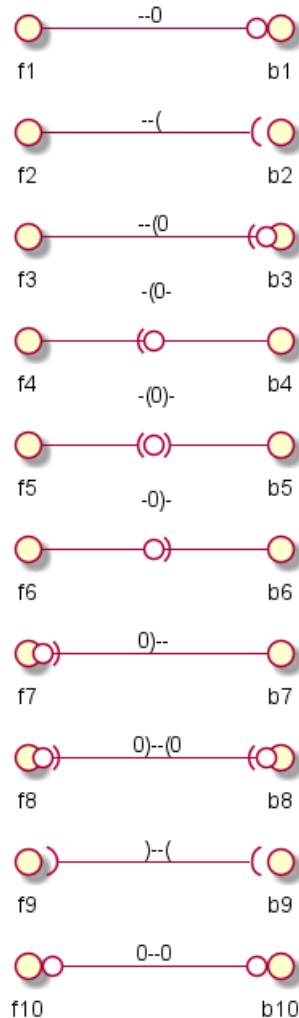
### 8.13.2 Type of '0' arrow or circle arrow

```
@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)-\n ""
f5 -(0)- b5 : "" -(0)-\n ""
```



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



## 8.14 Appendix: Test of inline style on all element

### 8.14.1 Simple element

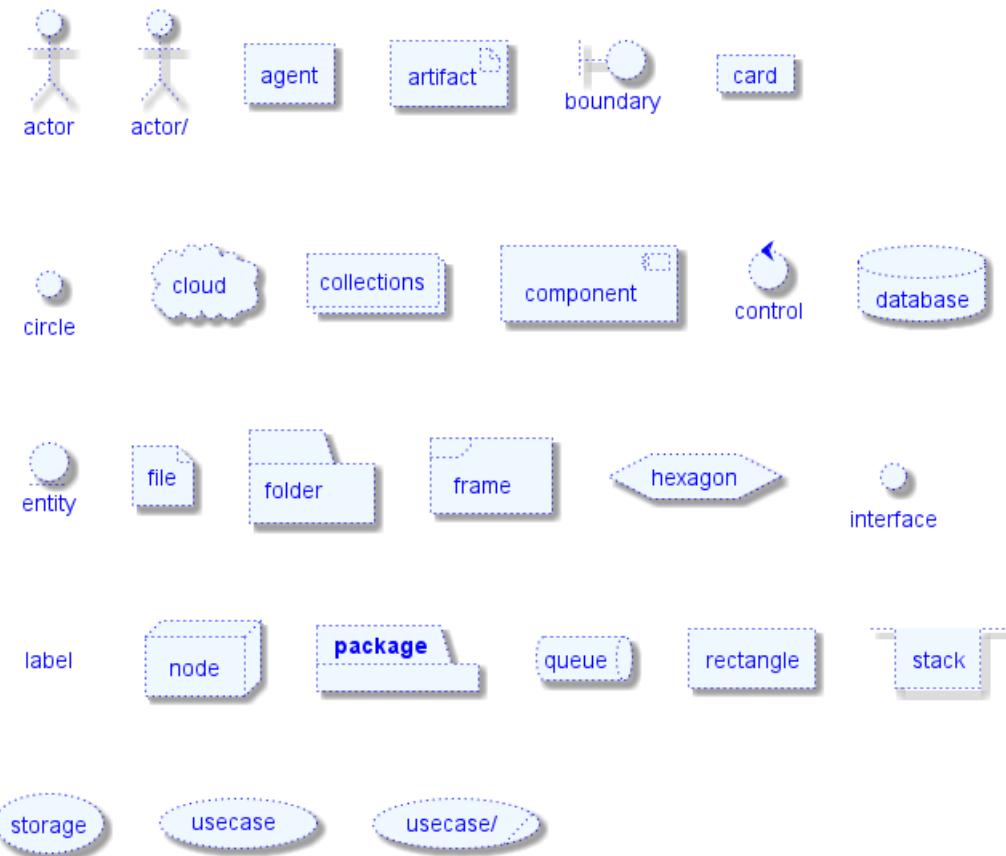
```
@startuml
actor actor #aliceblue;line:blue;line.dotted;text:blue
actor/ "actor/" #aliceblue;line:blue;line.dotted;text:blue
agent agent #aliceblue;line:blue;line.dotted;text:blue
artifact artifact #aliceblue;line:blue;line.dotted;text:blue
boundary boundary #aliceblue;line:blue;line.dotted;text:blue
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
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 Nested element

#### 8.14.3 Without sub-element

```

@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 With sub-element

```

@startuml
artifact      artifactVeryL00000000000000000000g      as "artifact" #aliceblue;line:blue;line.dotted;text:blue
file f1
}
card         cardVeryL00000000000000000000g      as "card" #aliceblue;line:blue;line.dotted;text:blue
file f2
}
cloud        cloudVeryL00000000000000000000g      as "cloud" #aliceblue;line:blue;line.dotted;text:blue
file f3
}
component    componentVeryL00000000000000000000g   as "component" #aliceblue;line:blue;line.dotted;text:blue
file f4
}
database     databaseVeryL00000000000000000000g    as "database" #aliceblue;line:blue;line.dotted;text:blue
file f5
}
file         fileVeryL00000000000000000000g      as "file" #aliceblue;line:blue;line.dotted;text:blue
file f6
}
folder        folderVeryL00000000000000000000g     as "folder" #aliceblue;line:blue;line.dotted;text:blue
file f7
}
frame         frameVeryL00000000000000000000g      as "frame" #aliceblue;line:blue;line.dotted;text:blue
file f8
}
hexagon       hexagonVeryL00000000000000000000g     as "hexagon" #aliceblue;line:blue;line.dotted;text:blue
file f9
}
node          nodeVeryL00000000000000000000g      as "node" #aliceblue;line:blue;line.dotted;text:blue
file f10
}
package       packageVeryL00000000000000000000g     as "package" #aliceblue;line:blue;line.dotted;text:blue
file f11
}

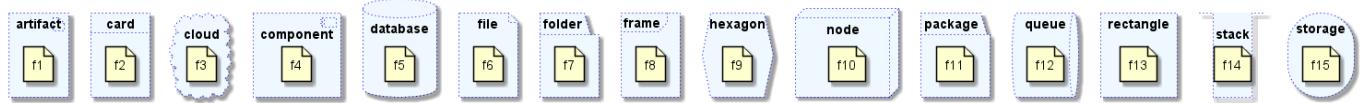
```



```

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

```



## 8.15 Appendix: Test of style on all element

### 8.15.1 Simple element

### 8.15.2 Global style (on componentDiagram)

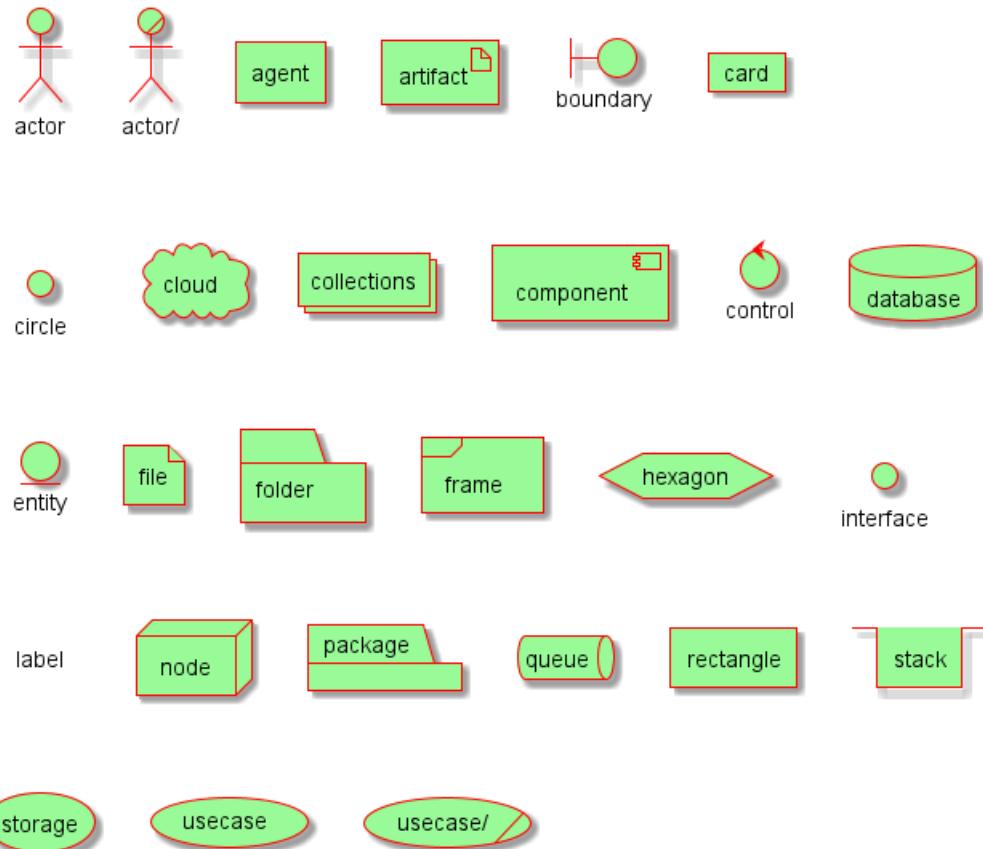
```

@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 1
    LineColor red
}
</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
queue queue
rectangle rectangle
stack stack
storage storage

```



```
usecase usecase
usecase/ "usecase/"
@enduml
```



### 8.15.3 Style for each element

```
@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 {
```



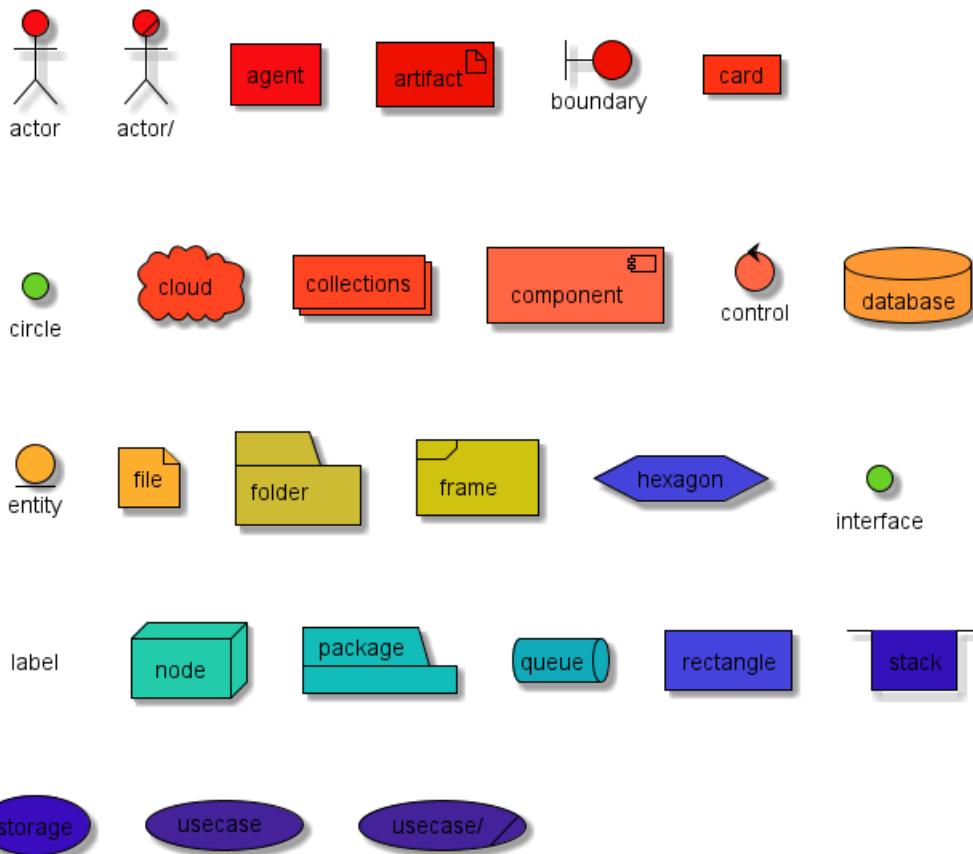
```
BackGroundColor #69d025
LineThickness 1
LineColor black
}
label {
    BackGroundColor black
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    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
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml

```



[Ref. QA-13261]

#### 8.15.4 Nested element (without level)

#### 8.15.5 Global style (on componentDiagram)

```

@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 2
    LineColor red
}
</style>
artifact artifact {
}

```

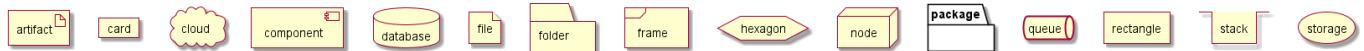


```

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 Style for each nested element

```

@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 Nested element (with one level)

### 8.15.8 Global style (on componentDiagram)

```

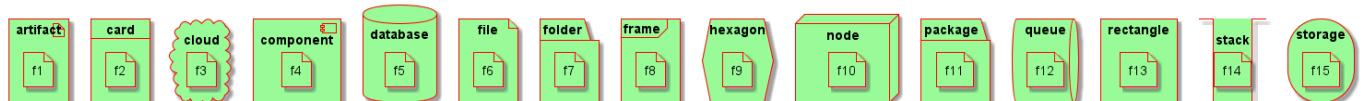
@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 1
    LineColor red
}
</style>
artifact e1 as "artifact" {
    file f1
}
card e2 as "card" {
    file f2
}
cloud e3 as "cloud" {
    file f3
}
component e4 as "component" {
    file f4
}
database e5 as "database" {
    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 Style for each nested element

```

@startuml
<style>
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {

```



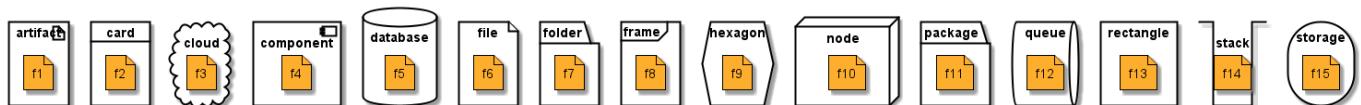
```
BackGroundColor #ff6644
LineThickness 1
LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    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

```



## 9 状态图

### 9.1 简单状态

使用 [\*] 开始和结束状态图。

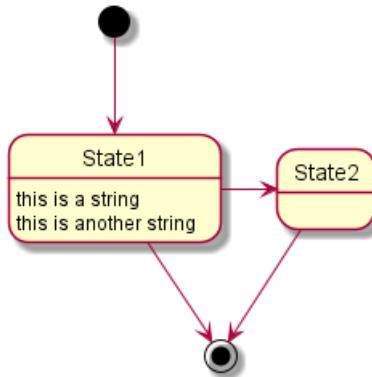
使用--> 添加箭头。

```
@startuml
```

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

```
State1 -> State2
State2 --> [*]
```

```
@enduml
```



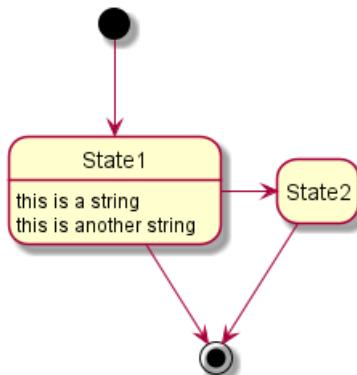
### 9.2 Change state rendering

You can use hide empty description to render state as simple box.

```
@startuml
```

```
hide empty description
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string
```

```
State1 -> State2
State2 --> [*]
@enduml
```



### 9.3 合成状态

一个状态也可能是合成的，必须使用关键字 `state` 和花括号来定义合成状态。

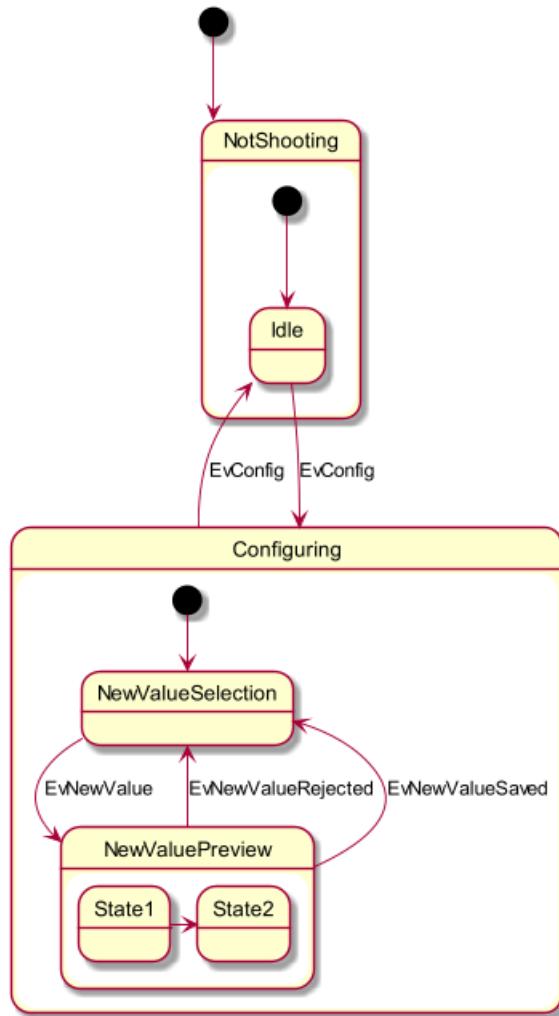
```
@startuml
scale 350 width
[*] --> NotShooting

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
```



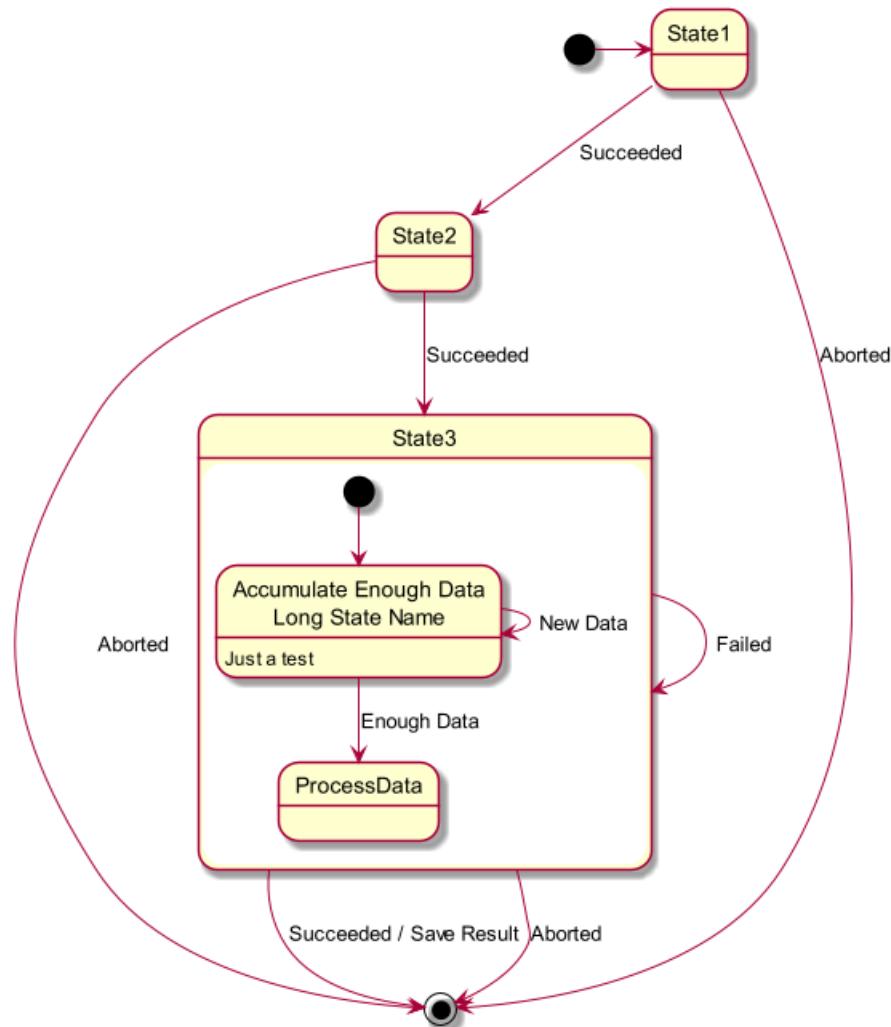
## 9.4 长名字

也可以使用关键字 `state` 定义长名字状态。

```
@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

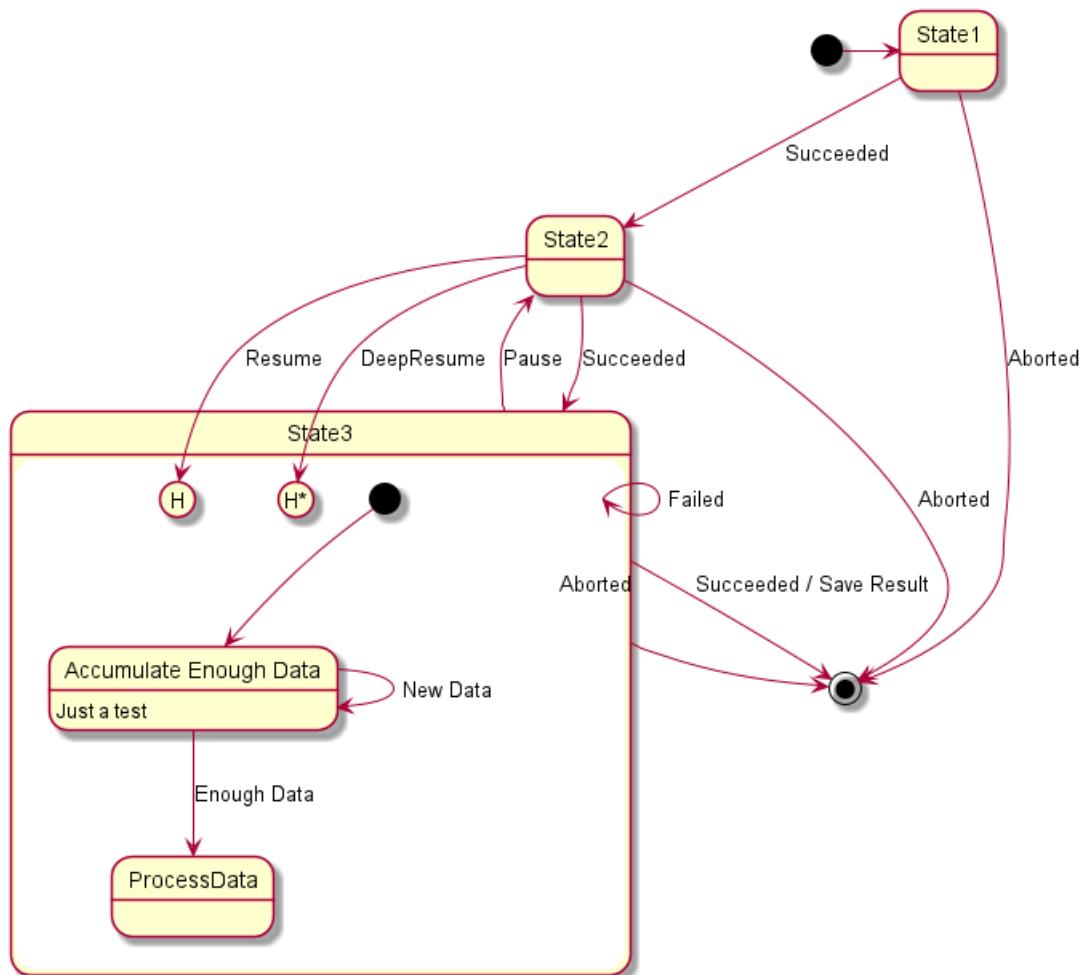
@enduml
```



## 9.5 History [[H], [H\*]]

You can use [H] for the history and [H\*] for the deep history of a substate.

```
@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 Fork [fork, join]

You can also fork and join using the <<fork>> and <<join>> stereotypes.



```

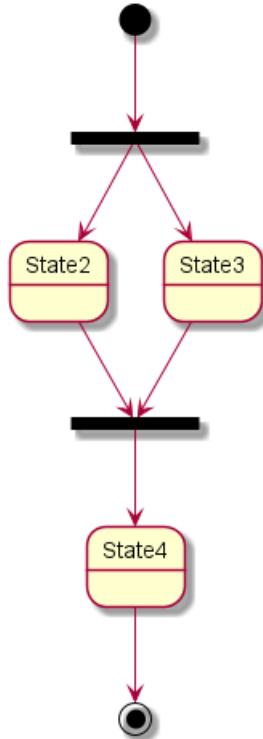
@startuml

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。

```

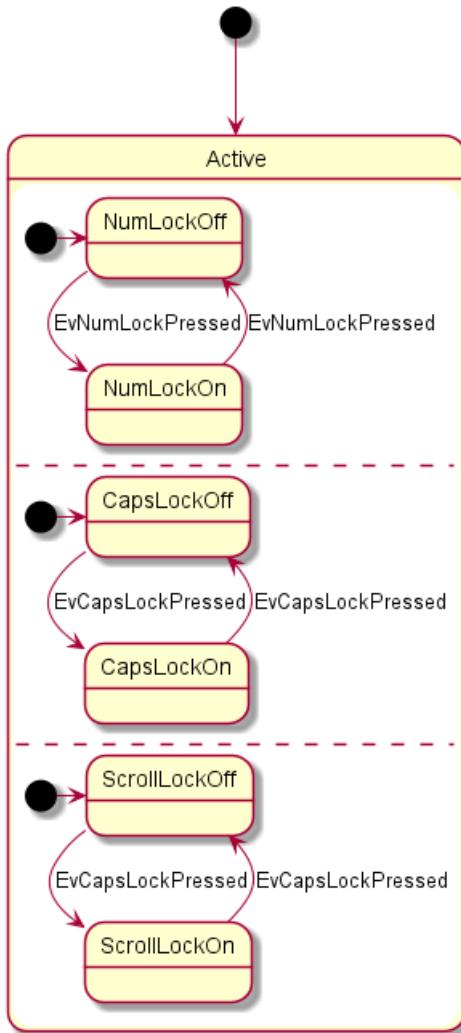
@startuml
[*] --> Active

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

```



@enduml



## 9.8 Conditional [choice]

The stereotype <<choice>> can be used to use conditional state.

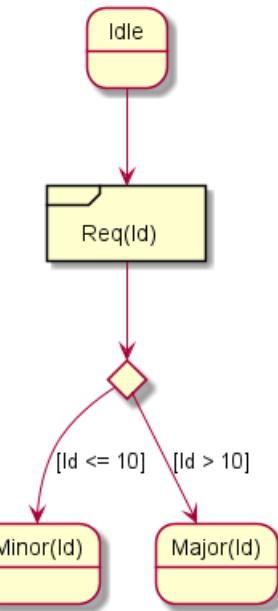
```

@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 Stereotypes full example [choice, fork, join, end]

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

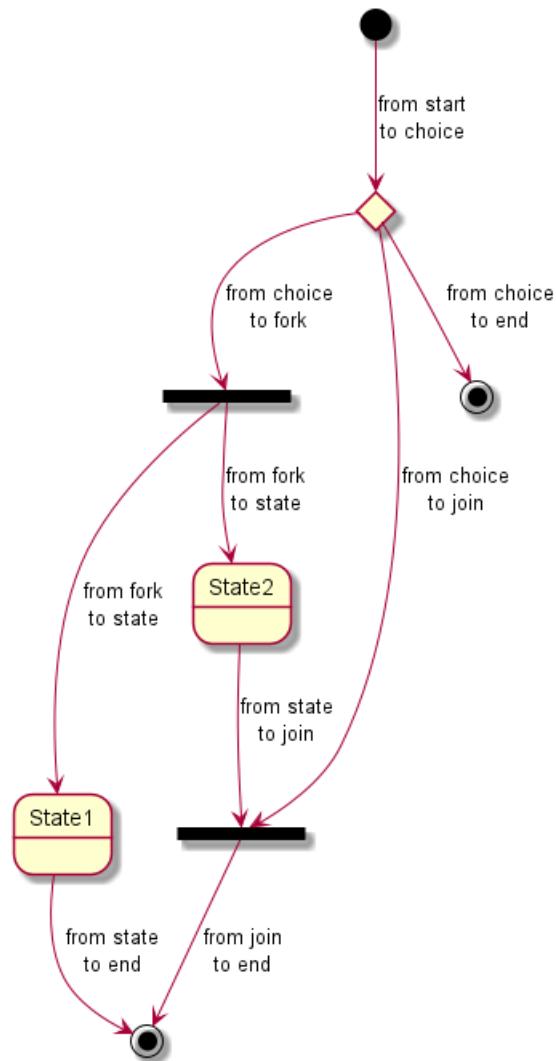
[*]      --> choice1 : from start\nto choice
choice1 --> fork1   : from choice\nto fork
choice1 --> join2   : from choice\nto join
choice1 --> end3    : from choice\nto end

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
```





[Ref. QA-404 and QA-1159]

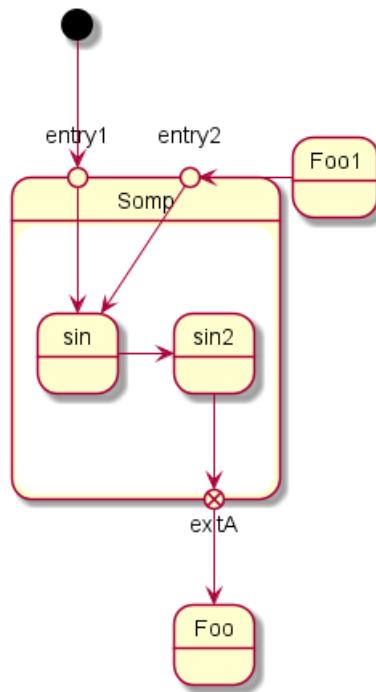
## 9.10 Point [entryPoint, exitPoint]

You can add **point** with <<entryPoint>> and <<exitPoint>> stereotypes:

```

@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
  
```



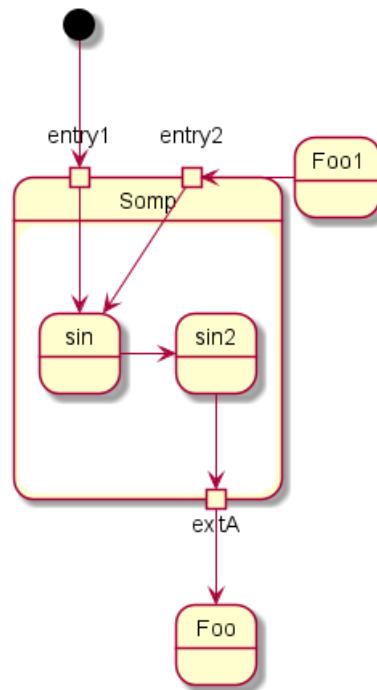


## 9.11 Pin [inputPin, outputPin]

You can add **pin** with `<<inputPin>>` and `<<outputPin>>` stereotypes:

```
@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 Expansion [expansionInput, expansionOutput]

You can add **expansion** with <<expansionInput>> and <<expansionOutput>> stereotypes:

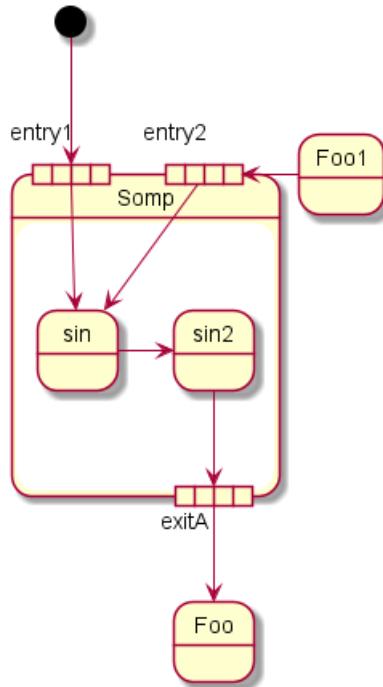
```

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

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

```





[Ref. QA-4309]

### 9.13 箭头方向

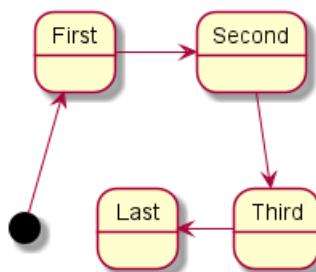
使用`->` 定义水平箭头，也可以使用下列格式强制设置箭头方向：

- `-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 Change line color and style

You can change line color and/or line style.

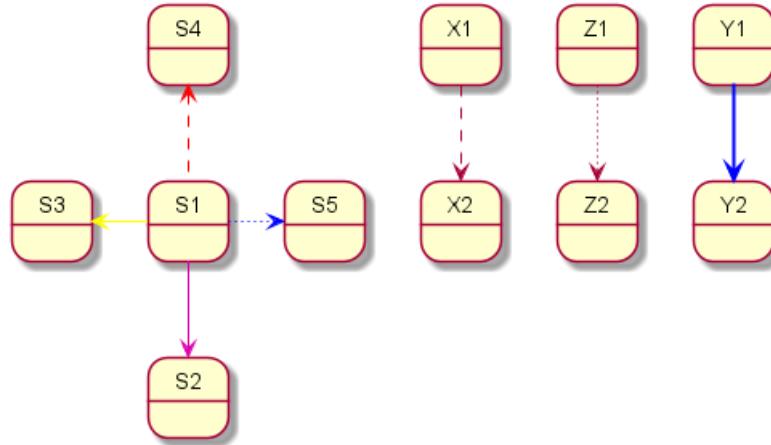


```

@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

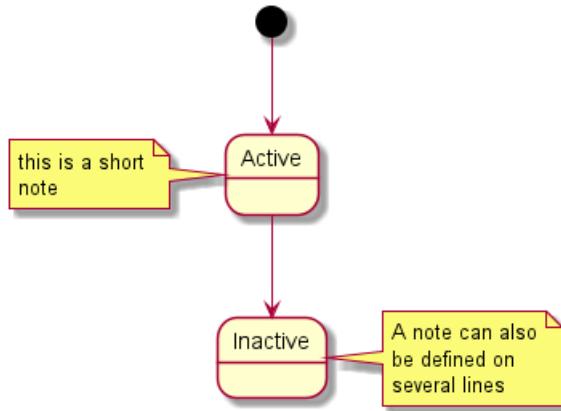
note left of Active : this is a short\nnote

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
```

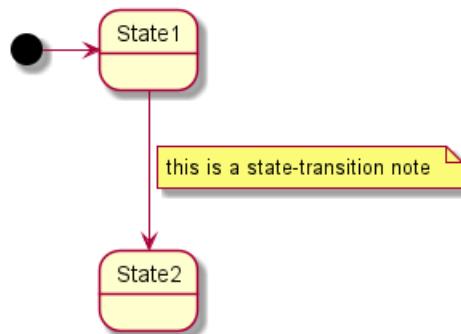
@enduml



## 9.16 Note on link

You can put notes on state-transition or link, with `note on link` keyword.

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



## 9.17 更多注释

可以在合成状态中放置注释。

@startuml

```
[*] --> NotShooting
state "Not Shooting State" as NotShooting {
```



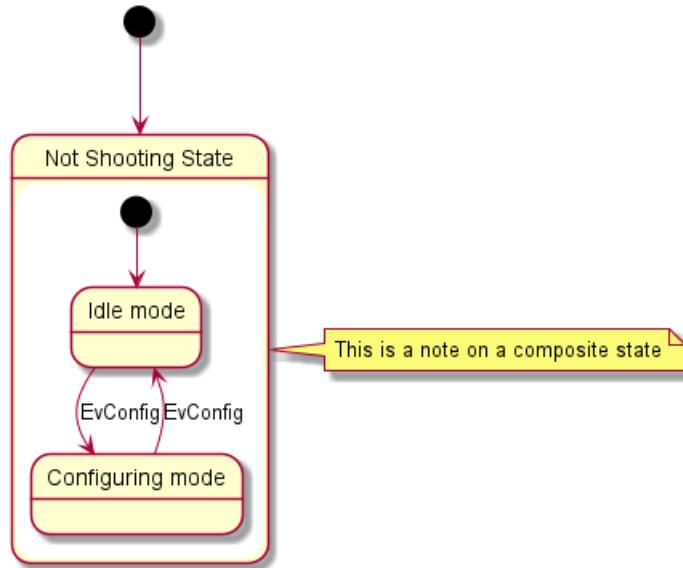
```

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

```



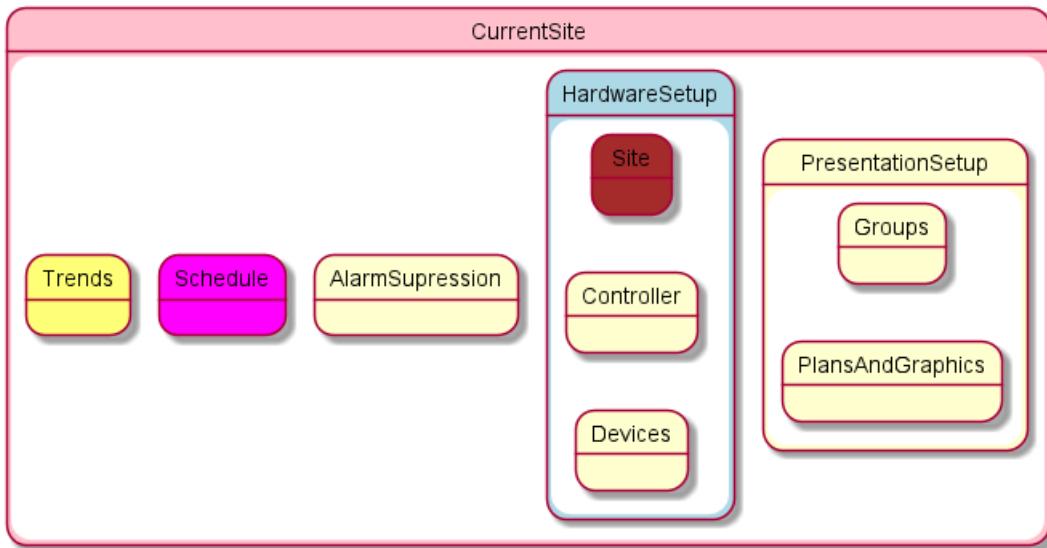
## 9.18 Inline color

```

@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

```





[Ref. 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，
- 在引入的文件中，
- 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
}

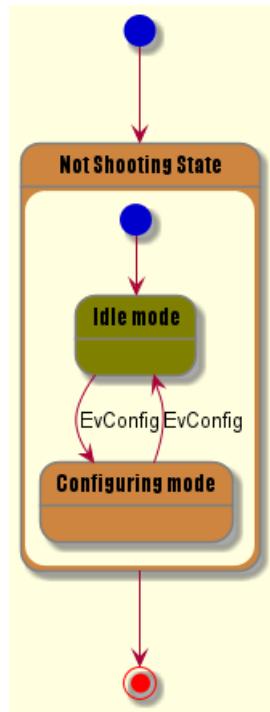
[*] --> NotShooting

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

```





## 9.20 Changing style

You can change style.

@startuml

```

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

```

[\*] --> NotShooting

```

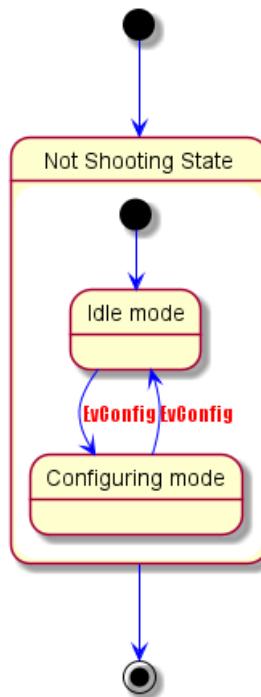
state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle <<Warning>>
    state "Configuring mode" as Configuring
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

```

NotShooting --> [\*]

@enduml





## 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`

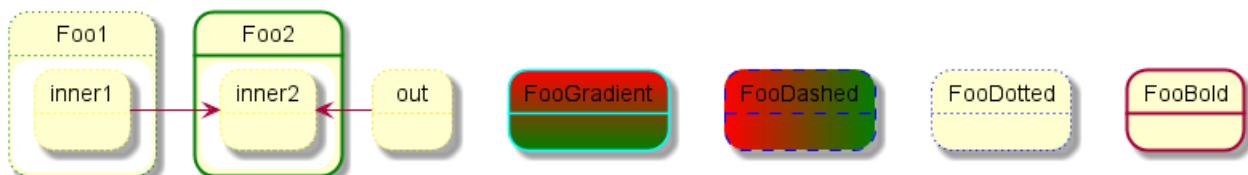
With background color first (`#color`), then line style and line color (`##[style]color`).

```

@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]

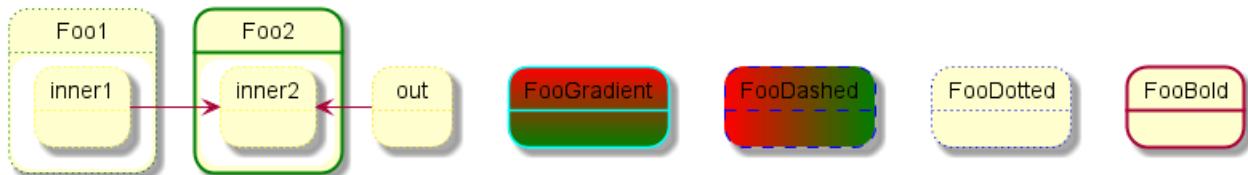
- #color;line:color;line.[bold|dashed|dotted];text:color

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

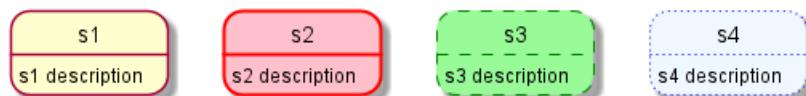
```
@startuml
@startuml
state FooGradient #red-green;line:00FFFF
state FooDashed #red|green;line.dashed;line:blue {
}
state FooDotted #line.dotted;line:blue {
}
state FooBold #line.bold {
}
state Foo1 #line.dotted;line:green {
state inner1 #line.dotted;line:yellow
}

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]



## 10 定时图

这只是个提案，主题和内容可能改变。

非常欢迎您参与这个新特性的讨论。您的反馈、创意和建议可以帮助我们找寻适合的解决方案。

### 10.1 声明参与者

使用 `concise` or `robust` 关键字声明参与者, 选择哪个取决于所需的显示样式。

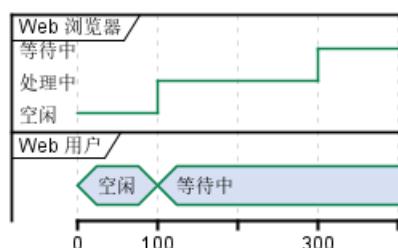
通过 `@` 标注, 和 `is` 动词定义状态.

```
@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
```



### 10.2 Binary and Clock

It's also possible to have binary and clock signal, using the following keywords:

- `binary`
- `clock`

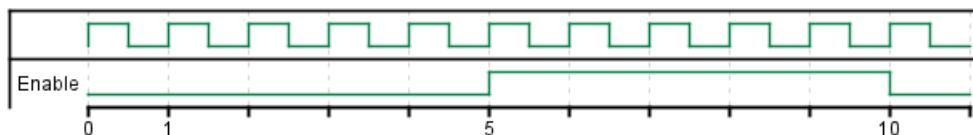
```
@startuml
clock clk with period 1
binary "Enable" as EN

@0
EN is low

@5
EN is high

@10
EN is low
@enduml
```





### 10.3 增加消息

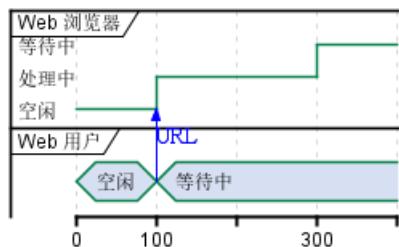
使用下述的语法增加对消息的描述。

```
@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

It is possible to use relative time with @.

```
@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

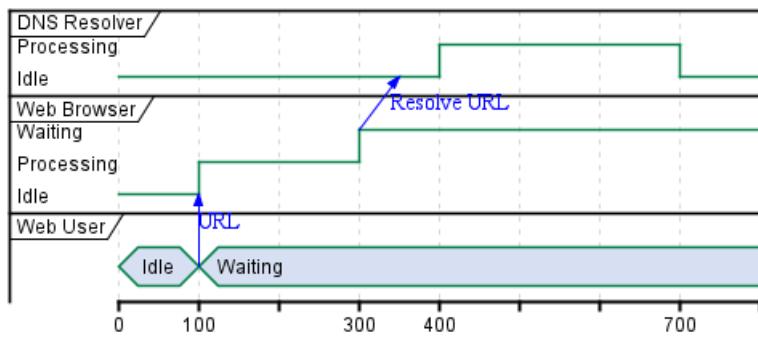
@+100
WU -> WB : URL
WU is Waiting
WB is Processing

@+200
WB is Waiting
WB -> DNS@+50 : Resolve URL
```



```
@+100
DNS is Processing
```

```
@+300
DNS is Idle
@enduml
```



## 10.5 Anchor Points

Instead of using absolute or relative time on an absolute time you can define a time as an anchor point by using the `as` keyword and starting the name with a `:`.

```
@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
```

```
@:start
EN is low
db is "0x0000"
```

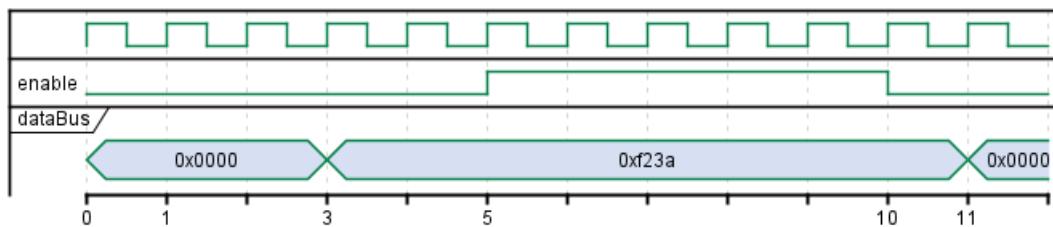
```
@:en_high
EN is high
```

```
@:en_low
EN is low
```

```
@:en_high-2
db is "0xf23a"
```

```
@:en_high+6
db is "0x0000"
@enduml
```





## 10.6 Participant oriented

Rather than declare the diagram in chronological order, you can define it by participant.

```
@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
```

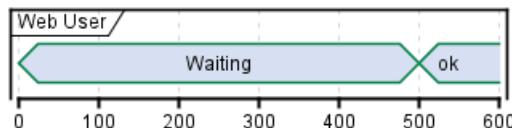


## 10.7 Setting scale

You can also set a specific scale.

```
@startuml
concise "Web User" as WU
scale 100 as 50 pixels
```

```
@WU
0 is Waiting
+500 is ok
@enduml
```



## 10.8 Initial state

You can also define an initial state.

```
@startuml
robust "Web Browser" as WB
```

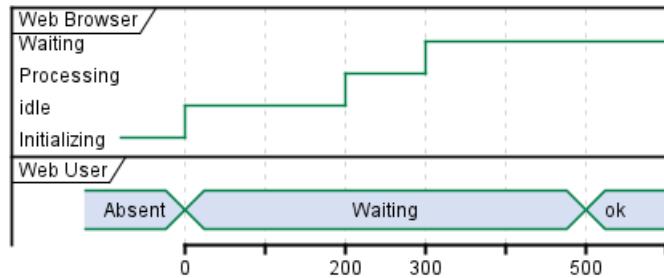


concise "Web User" as WU

WB is Initializing  
WU is Absent

@WB  
0 is idle  
+200 is Processing  
+100 is Waiting

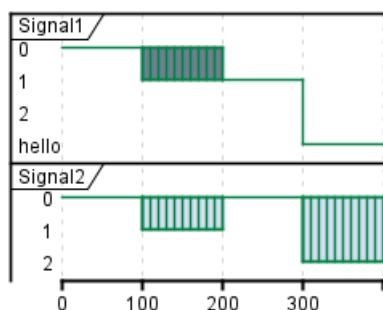
@WU  
0 is Waiting  
+500 is ok  
@enduml



## 10.9 Intricated state

A signal could be in some undefined state.

```
@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
```



## 10.10 Hidden state

It is also possible to hide some state.

```
@startuml
concise "Web User" as WU

@0
WU is {-}

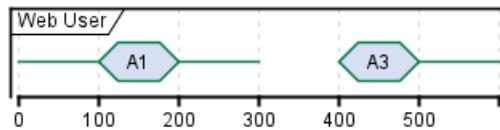
@100
WU is A1

@200
WU is {-}

@300
WU is {hidden}

@400
WU is A3

@500
WU is {-}
@enduml
```



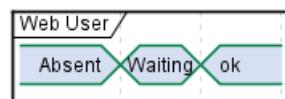
## 10.11 Hide time axis

It is possible to hide time axis.

```
@startuml
hide time-axis
concise "Web User" as WU

WU is Absent

@WU
0 is Waiting
+500 is ok
@enduml
```



## 10.12 Using Time and Date

It is possible to use time or date.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@2019/07/02
WU is Idle
```



WB is Idle

@2019/07/04

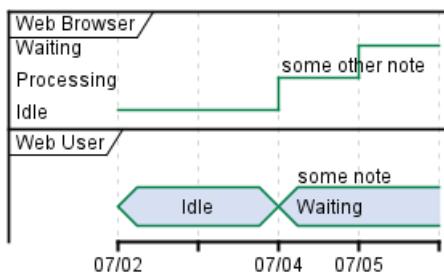
WU is Waiting : some note

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

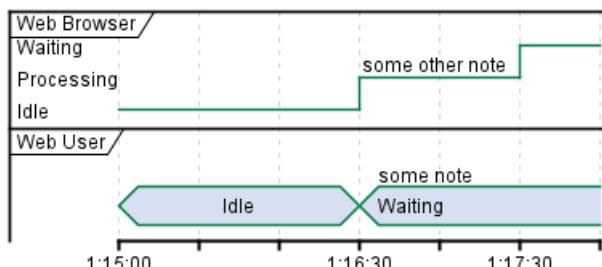
WU is Waiting : some note

WB is Processing : some other note

@1:17:30

WB is Waiting

@enduml



## 10.13 Adding constraint

It is possible to display time constraints on the diagrams.

@startuml

robust "Web Browser" as WB

concise "Web User" as WU

WB is Initializing

WU is Absent

@WB

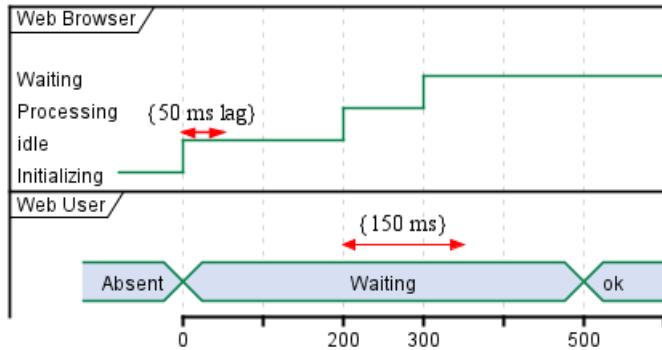
0 is idle

+200 is Processing



```
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}
```

```
@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



## 10.14 Highlighted period

You can highlight a part of diagram.

```
@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.

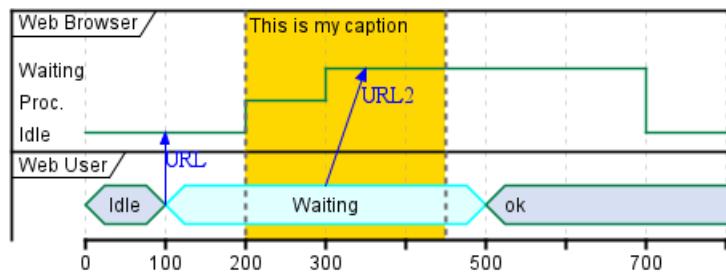
@300
WU -> WB@350 : URL2
WB is Waiting

@+200
WU is ok

@+200
WB is Idle

highlight 200 to 450 #Gold;line:DimGrey : This is my caption
@enduml
```





## 10.15 Adding texts

You can optionally add a title, a header, a footer, a legend and a caption:

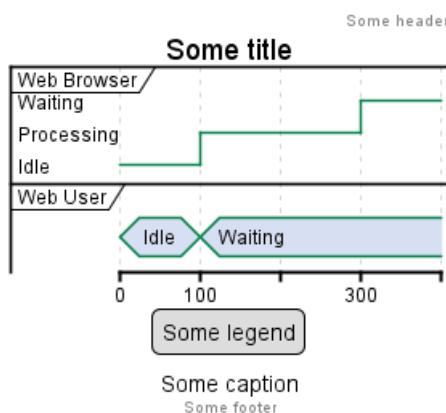
```
@startuml
Title Some title
header: Some header
footer: Some footer
legend
Some legend
end legend
caption Some caption

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 Complete example

Thanks to Adam Rosien for this example.

```
@startuml
concise "Client" as Client
concise "Server" as Server
```



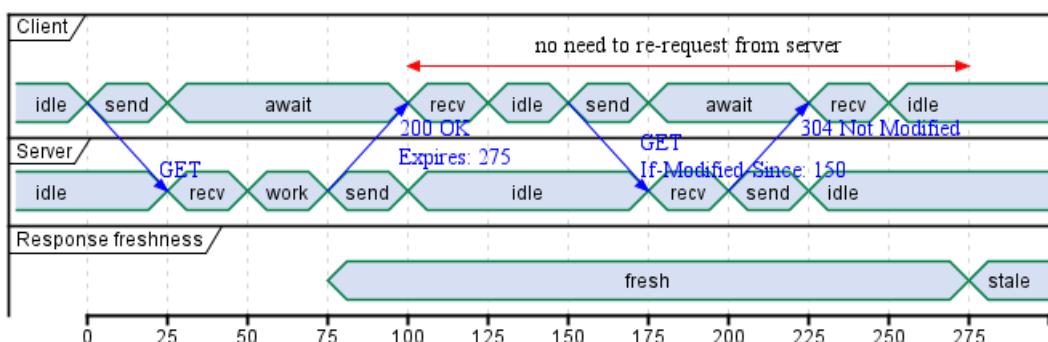
concise "Response freshness" as Cache

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 Digital Example

```
@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"

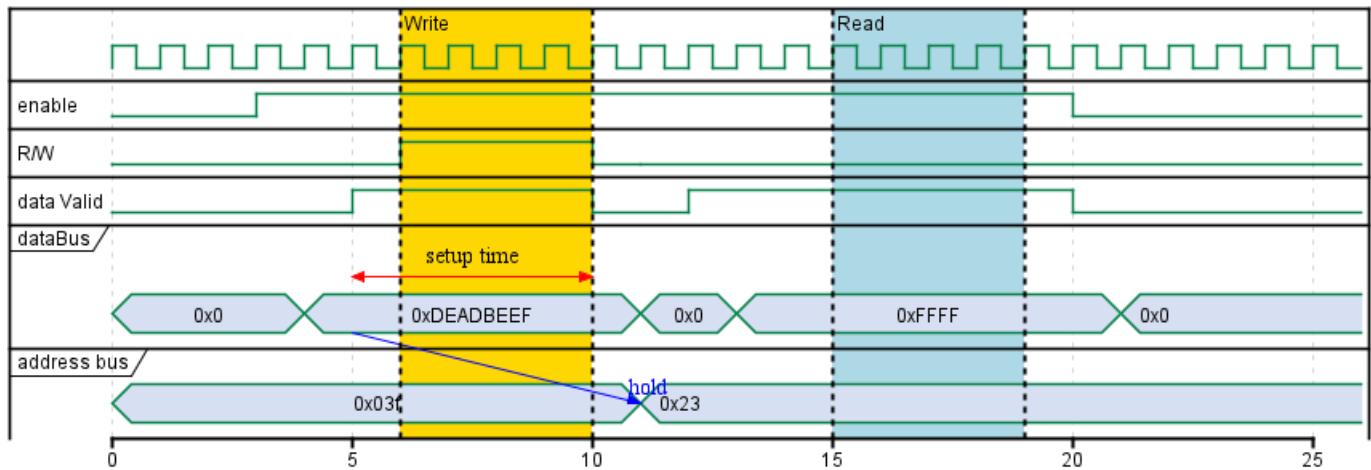
@12
dv is high
@13
db is "0xFFFF"

@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 Adding color

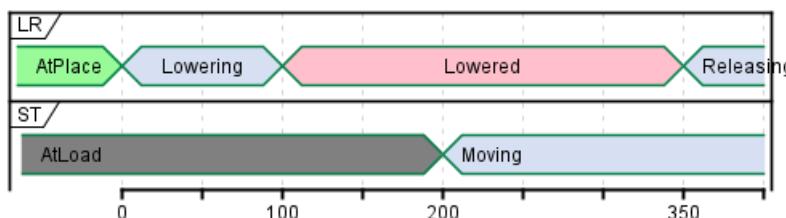
You can add color.

```
@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]



## 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"
}
@endjson
```

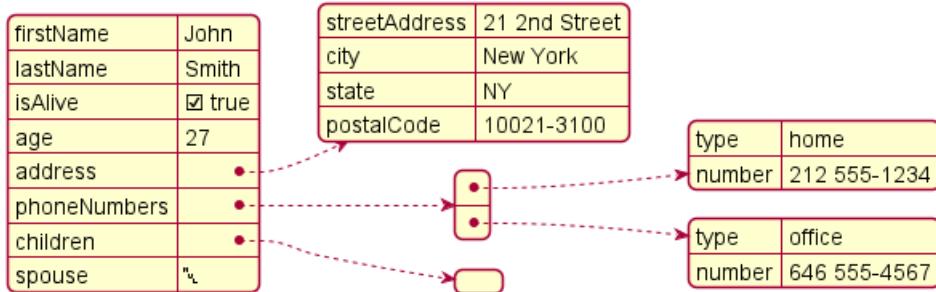
fruit	Apple
size	Large
color	Red

### 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
```

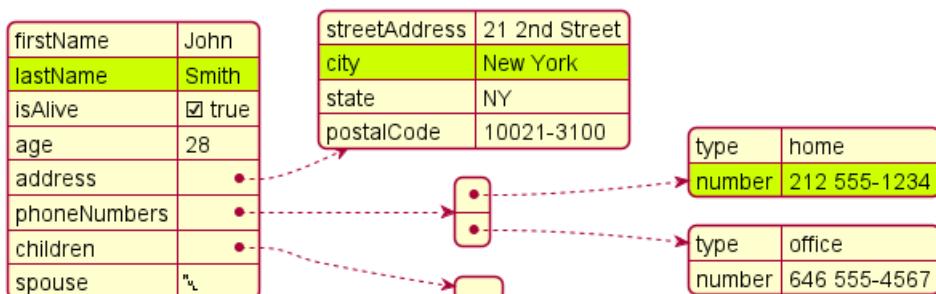




## 11.2 Highlight parts

```

@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
  
```



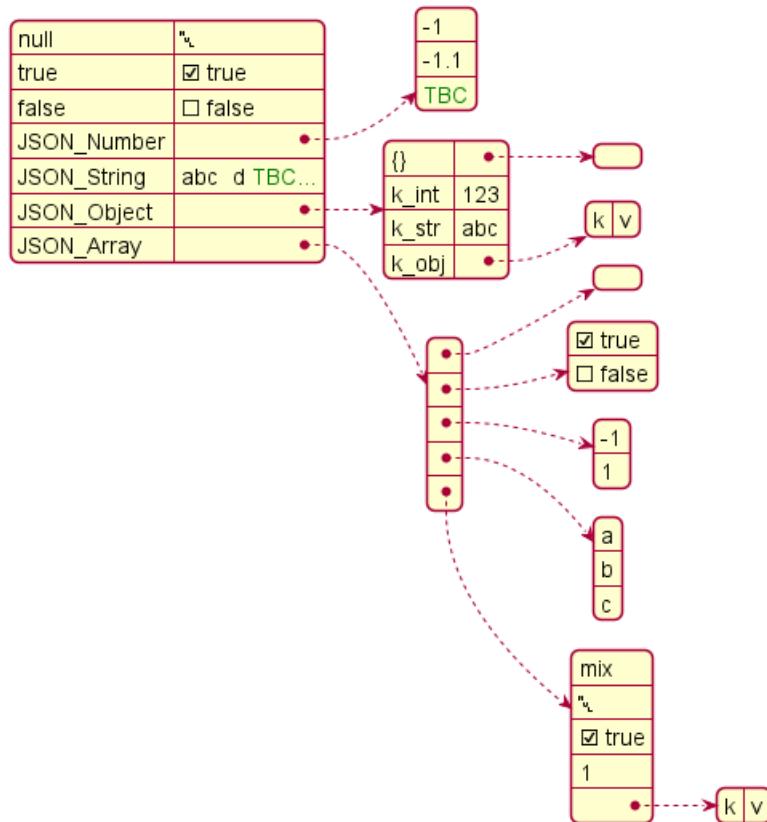
## 11.3 JSON basic element

### 11.3.1 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
```



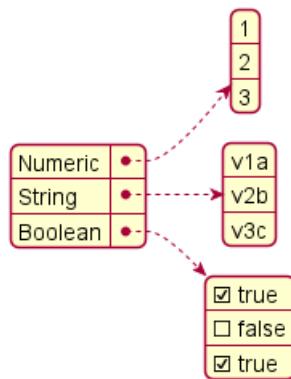
## 11.4 JSON array or table

### 11.4.1 Array type

```
@startjson
{
  "Numeric": [1, 2, 3],
  "String": ["v1a", "v2b", "v3c"],
  "Boolean": [true, false, true]
}
```



```
@endjson
```



#### 11.4.2 Minimal array or table

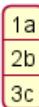
#### 11.4.3 Number array

```
@startjson
[1, 2, 3]
@endjson
```



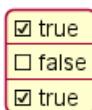
#### 11.4.4 String array

```
@startjson
["1a", "2b", "3c"]
@endjson
```



#### 11.4.5 Boolean array

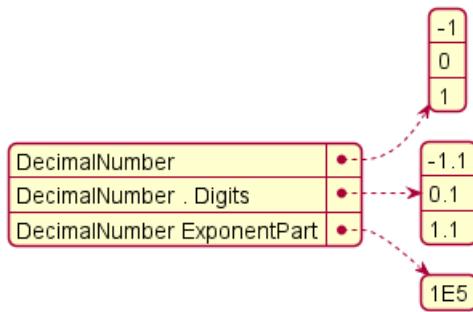
```
@startjson
[true, false, true]
@endjson
```



## 11.5 JSON numbers

```
@startjson
{
  "DecimalNumber": [-1, 0, 1],
  "DecimalNumber . Digits": [-1.1, 0.1, 1.1],
  "DecimalNumber ExponentPart": [1E5]
}
@endjson
```





## 11.6 JSON strings

### 11.6.1 JSON Unicode

On JSON you can use Unicode directly or by using escaped form like .

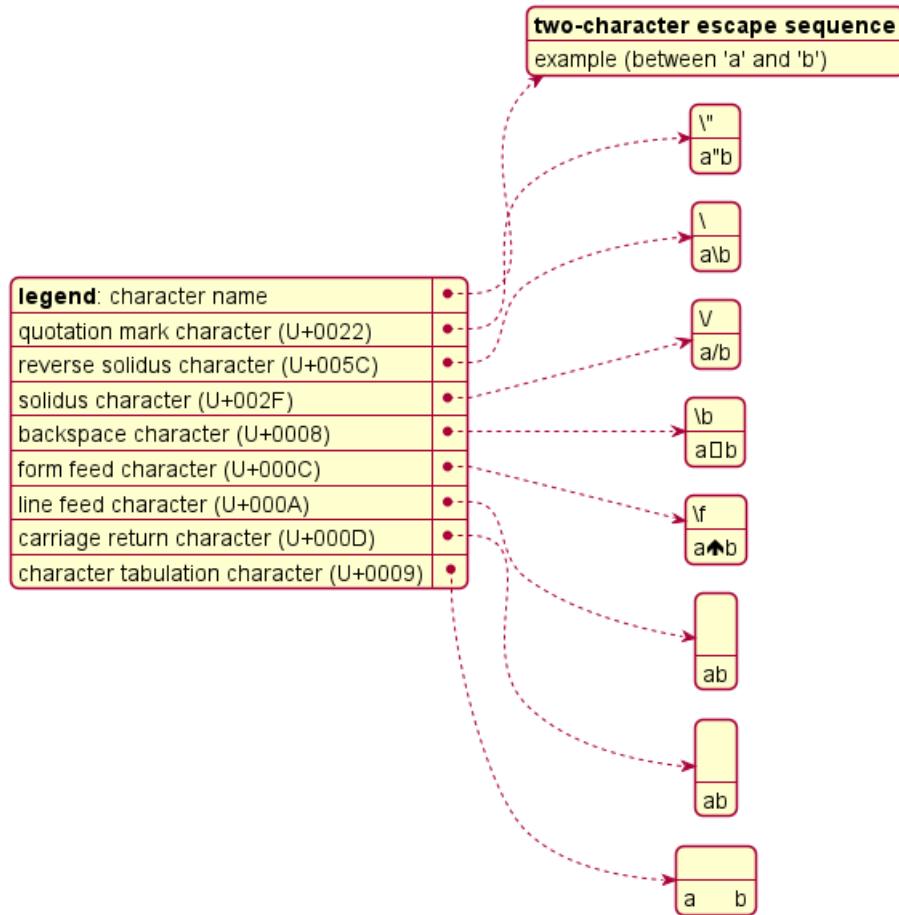
```
@startjson
{
    "<color:blue><b>code": "<color:blue><b>value",
    "a\\u005Cb": "a\u005Cb",
    "\\uD83D\\uDE10": "\uD83D\uDE10",
    " ":
}
@endjson
```

code	value
a\u005Cb	a\b
\uD83D\uDE10	😊
😊	😊

### 11.6.2 JSON two-character escape sequence

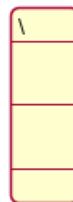
```
@startjson
{
    "**legend**: character name": "**two-character escape sequence**", "example (between"
    "quotation mark character (U+0022)": ["\\\"", "a\"b"],
    "reverse solidus character (U+005C)": ["\\\\\\\", "a\\\"b"],
    "solidus character (U+002F)": ["\\\\\\/", "a\\/b"],
    "backspace 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
```





**TODO:** FIXME FIXME or not , on the same item as management in PlantUML **TODO:** FIXME

```
@startjson
[
  "\\\\",
  "\\n",
  "\\r",
  "\\t"
]
@endjson
```



## 11.7 Minimal JSON examples

```
@startjson
"Hello world!"
@endjson
```

Hello world!

```
@startjson
```

```
42
@endjson
```

42

```
@startjson
true
@endjson
```

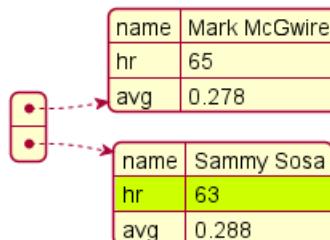
true

(Examples come from STD 90 - Examples)

## 11.8 Using (global) style

### 11.8.1 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
```



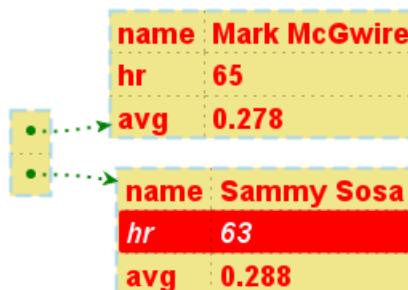
### 11.8.2 With style

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]



## 12 Display YAML Data

YAML format is widely used in software.

You can use PlantUML to visualize your data.

To activate this feature, the diagram must:

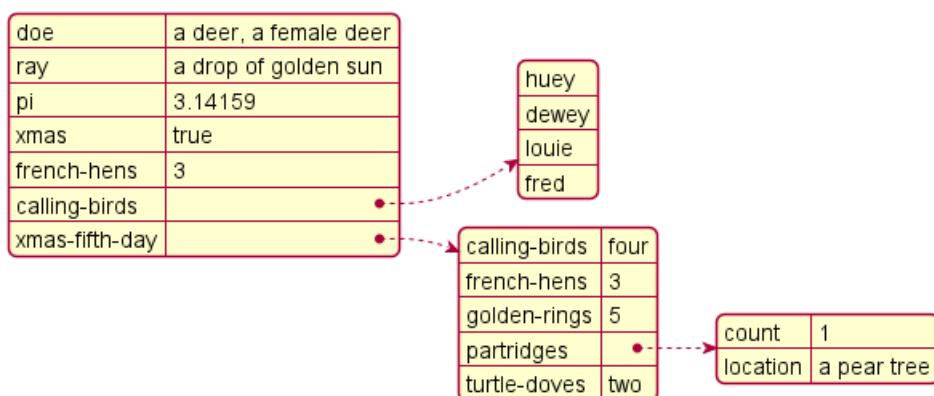
- begin with `@startyaml` keyword
- end with `@endyaml` keyword.

```
@startyaml
fruit: Apple
size: Large
color: Red
@endyaml
```

fruit	Apple
size	Large
color	Red

### 12.1 Complex example

```
@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 Specific key (with symbols or unicode)

```
@startyaml
@fruit: Apple
$size: Large
&color: Red
: Heart
%: Per mille
@endyaml
```

@fruit	Apple
\$size	Large
&color	Red
♥	Heart
%	Per mille

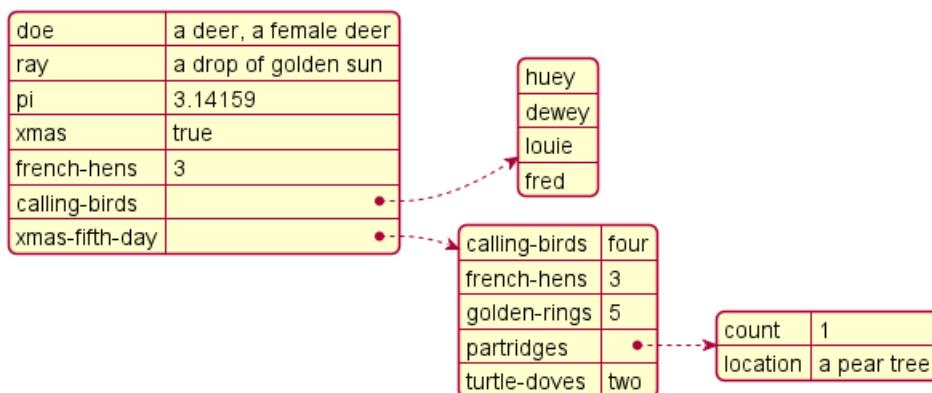
[Ref. QA-13376]

## 12.3 Highlight parts

### 12.3.1 Normal style

```
@startyaml
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

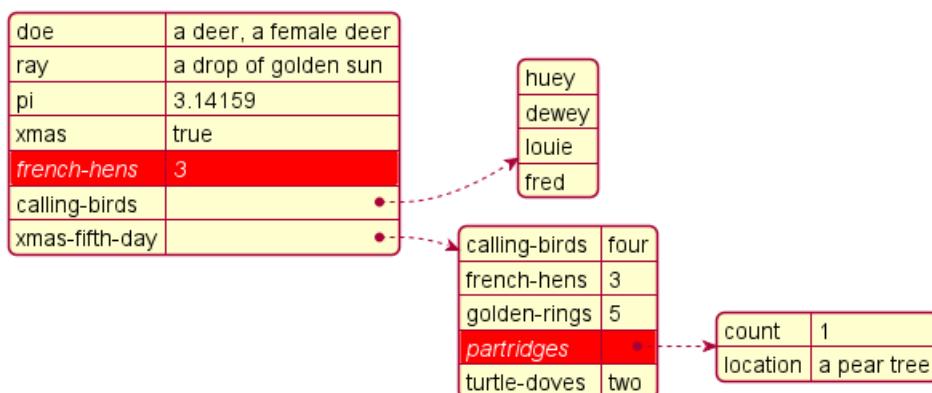
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



### 12.3.2 Customised style

```
@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 (global) style

### 12.4.1 Without style (*by default*)

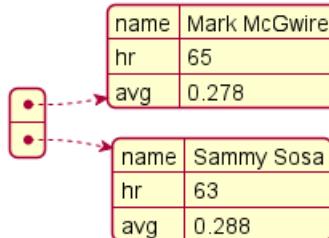
```
@startyaml
-
  name: Mark McGwire
  hr: 65
  avg: 0.278
```



```

-
  name: Sammy Sosa
  hr: 63
  avg: 0.288
@endyaml

```



### 12.4.2 With style

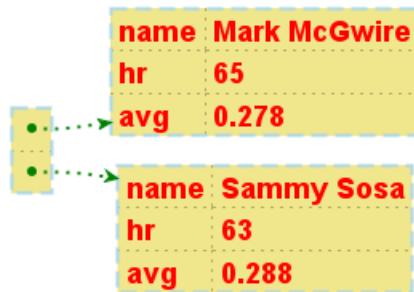
You can use style to change rendering of elements.

```

@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 Network diagram (nwdiag)

nwdiag has been created by Takeshi Komiya and allows to quickly draw network diagrams. So we thank him for his creation!

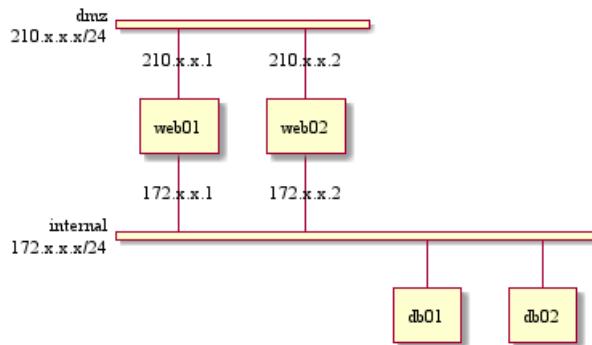
Since the syntax is clear and simple, this has been integrated within PlantUML. We reuse here the examples that Takeshi has documented.

### 13.1 Simple diagram

```
@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 Define multiple addresses

```
@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";

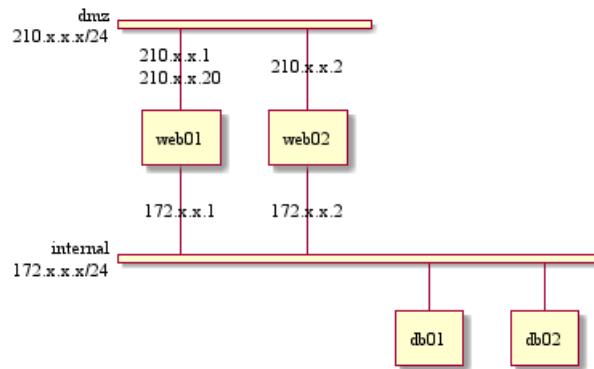
        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01;
    }
}
```



```

        db02;
    }
}
@enduml

```



## 13.3 Grouping nodes

### 13.3.1 Define group inside network definitions

```

@startuml
nwdiag {
    network Sample_front {
        address = "192.168.10.0/24";

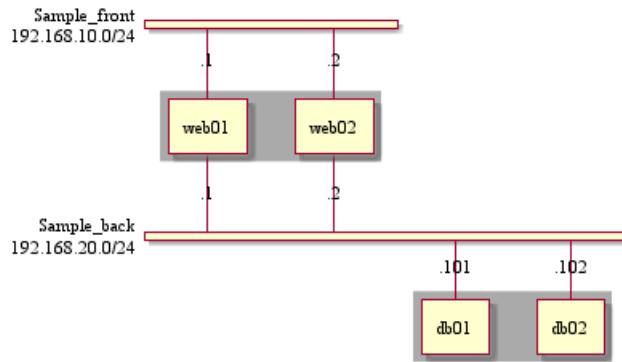
        // define group
        group web {
            web01 [address = ".1"];
            web02 [address = ".2"];
        }
    }

    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

```

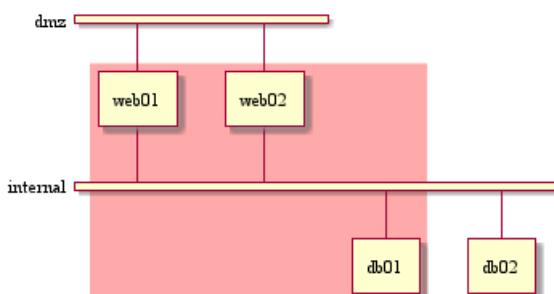




### 13.3.2 Define group outside of network definitions

```
@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 Define several groups on same network

#### 13.3.4 Example with 2 group

```
@startuml
nwdiag {
    group {
```

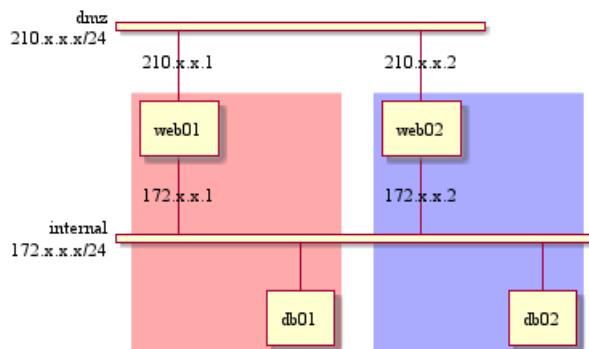
```

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 Example with 3 groups

```

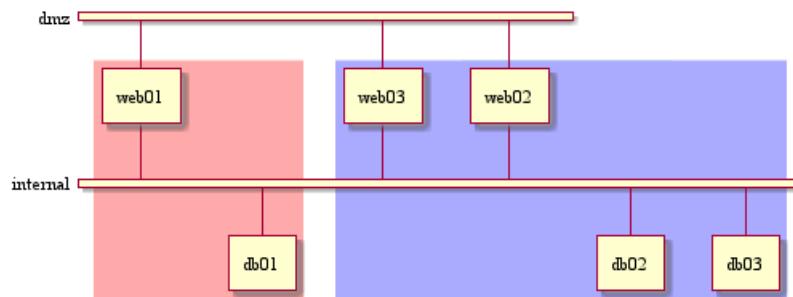
@startuml
nwdiag {
group {
color = "#FFaaaa";
web01;
db01;
}
group {
color = "#aaFFaa";
web02;
db02;
}
group {
color = "#aaaaFF";
web03;
db03;
}
}
```



```

}
network dmz {
    web01;
    web02;
    web03;
}
network internal {
    web01;
    db01 ;
    web02;
    db02 ;
    web03;
    db03;
}
}
@enduml

```



[Ref. QA-13138]

## 13.4 Extended Syntax (for network or group)

### 13.4.1 Network

For network or network's component, you can add or change:

- addresses (*separated by comma ,*);
- color;
- description;
- shape.

```

@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"]
    }
}

```



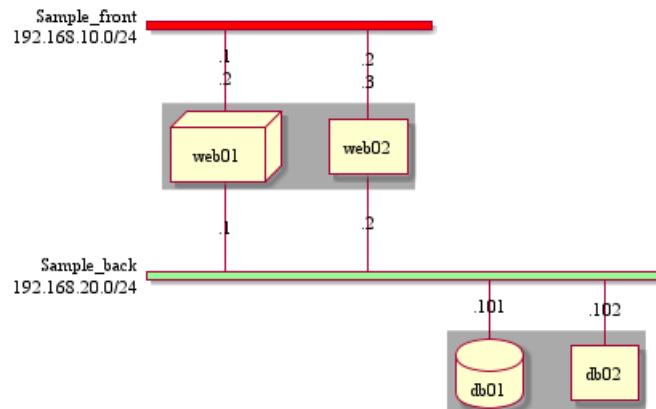
```

web02 [address = ".2"]
db01 [address = ".101", shape = database ]
db02 [address = ".102"]

// define network using defined nodes
group db {
    db01;
    db02;
}
}

@enduml

```



### 13.4.2 Group

For a group, you can add or change:

- color;
- description.

```

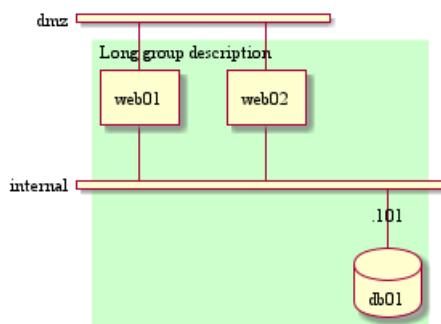
@startuml
nwdiag {
    group {
        color = "#CCFFCC";
        description = "Long group description";

        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.

```

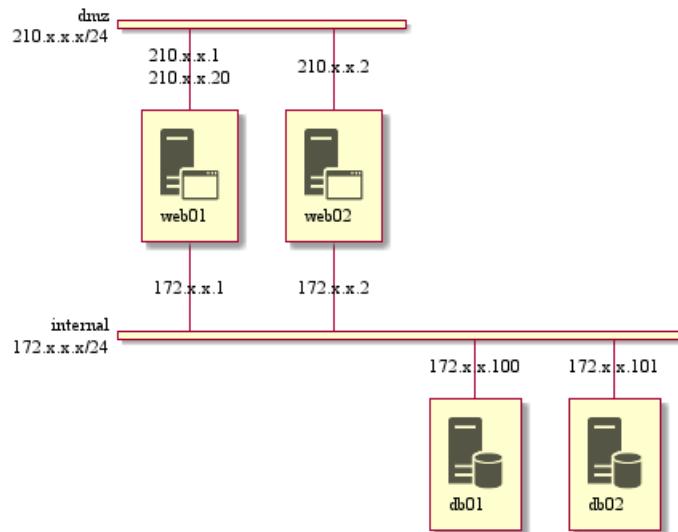
@startuml
!include <office/Servers/application_server>
!include <office/Servers/database_server>

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 `\n` to make a newline:

```
@startuml

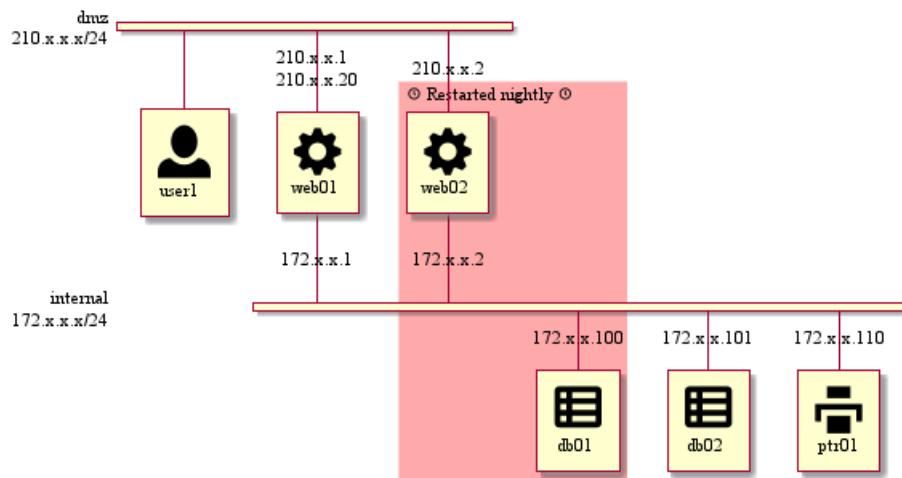
nwdiag {
    group nightly {
        color = "#FFAAAA";
        description = "<&clock> Restarted nightly <&clock>";
        web02;
        db01;
    }
    network dmz {
        address = "210.x.x.x/24"

        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;
    }

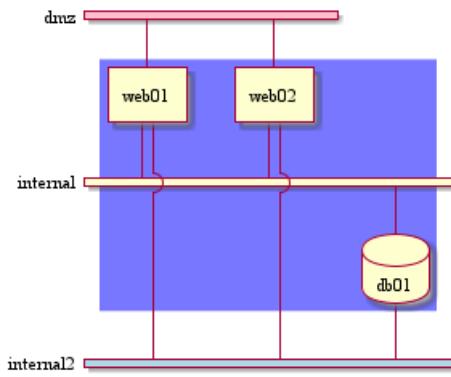
    network internal {
        web01;
        web02;
        db01 [shape = database ];
    }

    network internal2 {
        color = "LightBlue";

        web01;
        web02;
        db01;
    }
}
@enduml

```





## 13.8 Peer networks

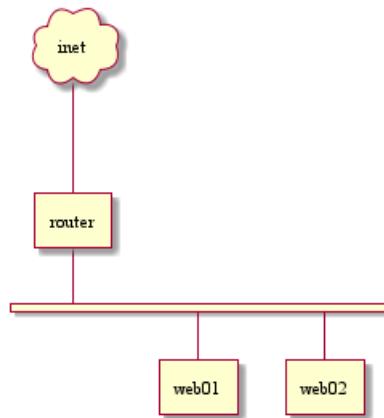
Peer networks are simple connections between two nodes, for which we don't use a horizontal "busbar" network

```

@startuml
nwdiag {
    inet [shape = cloud];
    inet -- router;

    network {
        router;
        web01;
        web02;
    }
}
@enduml

```



## 13.9 Peer networks and group

### 13.9.1 Without group

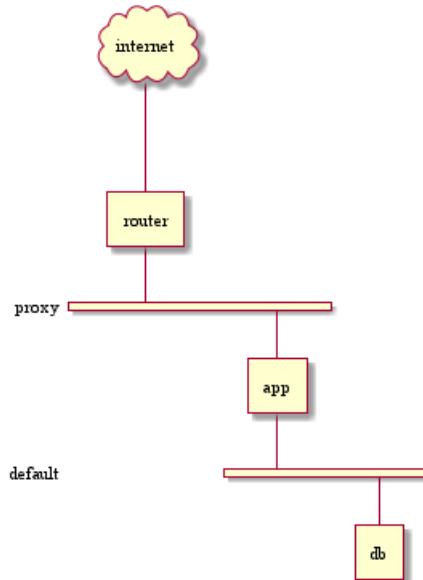
```

@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

```



```
network proxy {
    router;
    app;
}
network default {
    app;
    db;
}
}
@enduml
```



### 13.9.2 Group on first

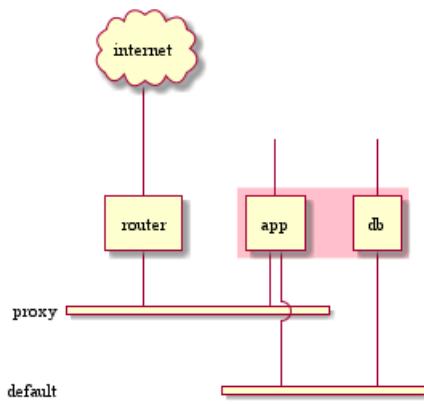
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    group {
        color = "pink";
        app;
        db;
    }

    network proxy {
        router;
        app;
    }

    network default {
        app;
        db;
    }
}
@enduml
```

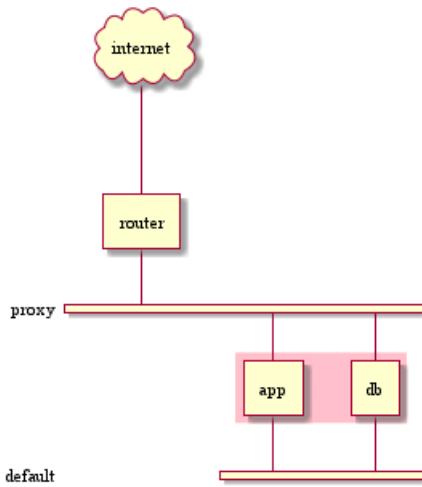




### 13.9.3 Group on second

```
@startuml  
nwdiag {  
    internet [ shape = cloud];  
    internet -- router;  
  
    network proxy {  
        router;  
        app;  
    }  
  
    group {  
        color = "pink";  
        app;  
        db;  
    }  
  
    network default {  
        app;  
        db;  
    }  
}  
@enduml
```





**TODO: FIXME** Why the line on proxy for 'db'? ('db' must be only on 'default network') [See example without group]

#### 13.9.4 Group on third

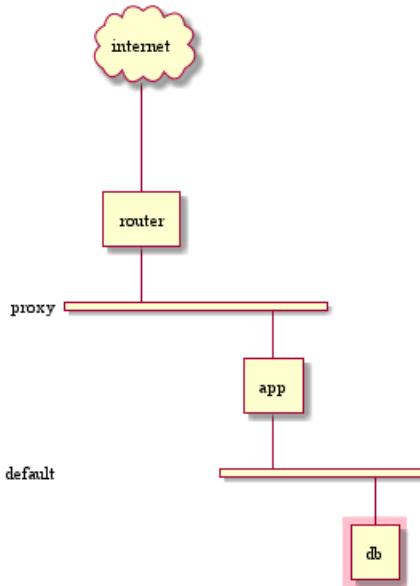
```

@startuml
nwdiag {
    internet [ shape = cloud ];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
        app;
        db;
    }
    group {
        color = "pink";
        app;
        db;
    }
}
@enduml

```





**TODO:** FIXME [Ref. Issue#408 and QA-12655] **TODO:** Not totally fixed

### 13.10 Add title, caption, header, footer or legend on network diagram

```
@startuml
```

```
header some header
```

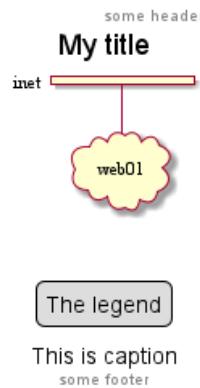
```
footer some footer
```

```
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]

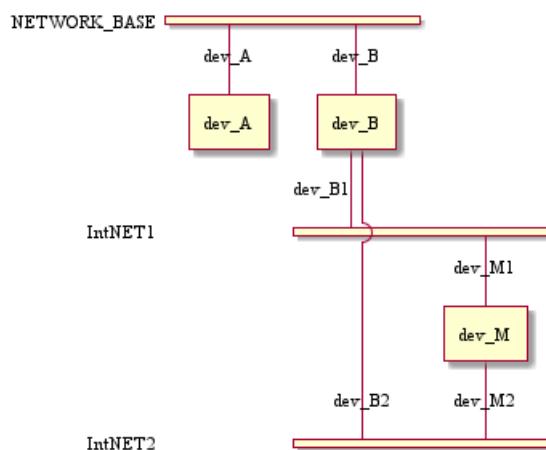
### 13.11 Change width of the networks

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
```

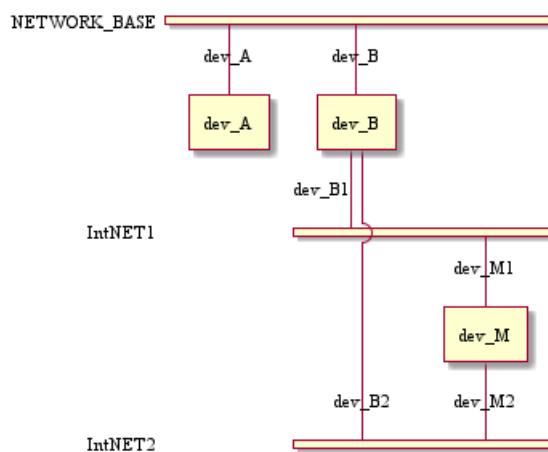


- only the first

```
@startuml
```



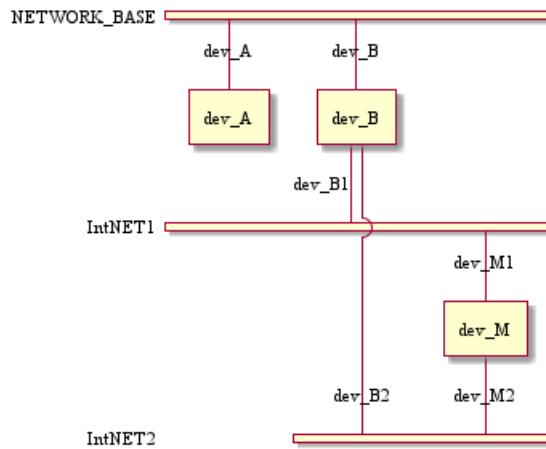
```
nwdiag {
    network NETWORK_BASE {
        width = full
        dev_A [address = "dev_A" ]
        dev_B [address = "dev_B" ]
    }
    network IntNET1 {
        dev_B [address = "dev_B1" ]
        dev_M [address = "dev_M1" ]
    }
    network IntNET2 {
        dev_B [address = "dev_B2" ]
        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
```



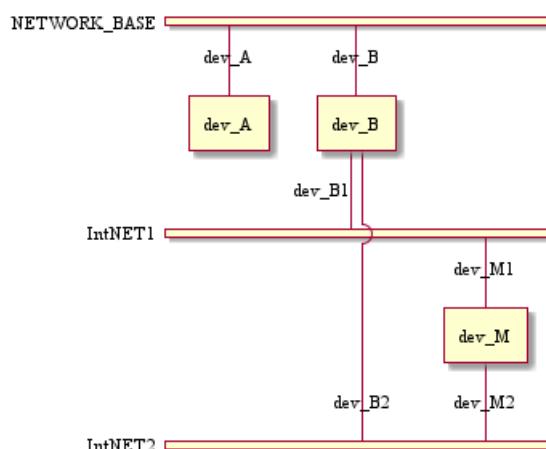


- all the network (with same full width)

```

@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

```



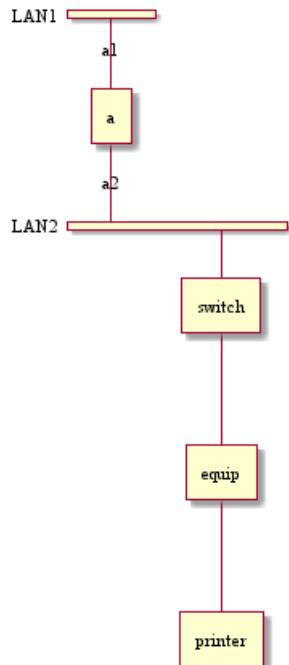
## 13.12 Other internal networks

You can define other internal networks (TCP/IP, USB, SERIAL,...).



- Without address or type

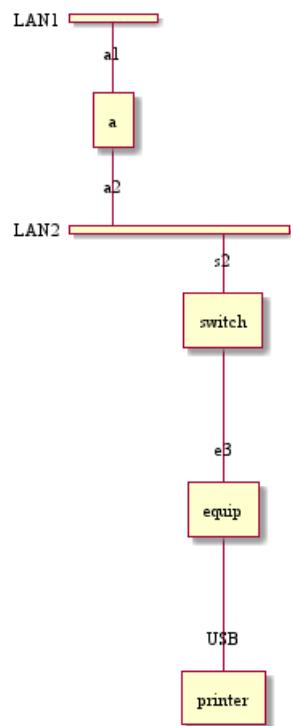
```
@startuml
nwdiag {
    network LAN1 {
        a [address = "a1"];
    }
    network LAN2 {
        a [address = "a2"];
        switch;
    }
    switch -- equip;
    equip -- printer;
}
@enduml
```



- 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]



## 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 dropdown^
}
@endsalt
```



这个工具是用来讨论简单的示例窗口。

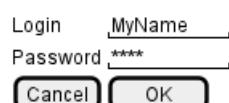
### 14.2 使用表格

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
```



### 14.3 分组框

```
@startsalt
{^"我的分组框"
    用户名 | "MyName"
    密码 | "*****"
    [取消] | [确认]
}
@endsalt
```



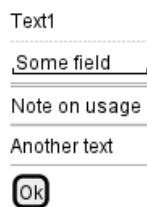
[参见 QA-5840]

### 14.4 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
```





## 14.5 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

```



## 14.6 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]

## 14.7 Enclosing brackets [ , ]

You can define subelements by opening a new opening bracket.

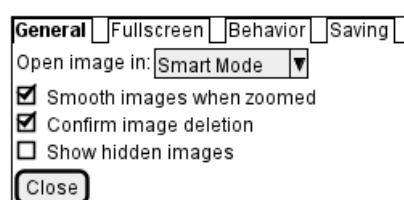
```
@startsalt
{
Name      | "
Modifiers: | { (X) public | () default | () private | () protected
           | [] abstract | [] final   | [] static }
Superclass: | { "java.lang.Object" | [Browse...]
}
@endsalt
```

Name	<input type="text"/>
Modifiers:	<input checked="" type="radio"/> public <input type="radio"/> default <input type="radio"/> private <input type="radio"/> protected <input type="checkbox"/> abstract <input type="checkbox"/> final <input type="checkbox"/> static
Superclass:	<input type="text"/> <input type="button" value="Browse..."/>

## 14.8 添加选项卡

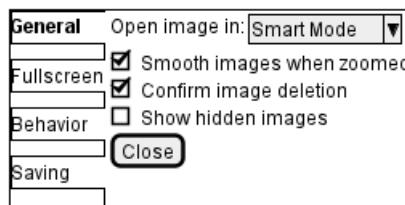
你可以通过{/ 标记增加对应的选项卡。注意：可以使用 HTML 代码来增加粗体效果。

```
@startsalt
{+
{/ <b>General | Fullscreen | Behavior | Saving </b>}
{
{ 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：

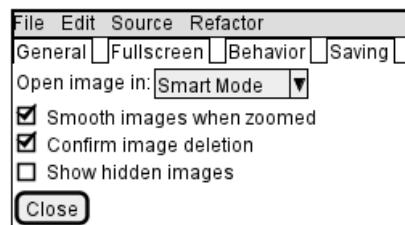
```
@startsalt
{+
{/ <b>General
Fullscreen
Behavior
Saving } |
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
[Close]
}
}
@endsalt
```



## 14.9 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
```

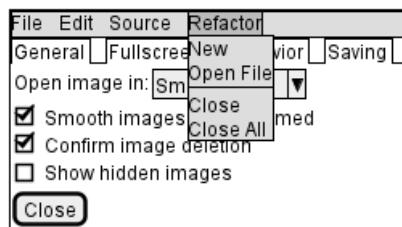


你也可以打开一个菜单：

```
@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
```



## 14.10 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	*

## 14.11 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
```

```
.
```

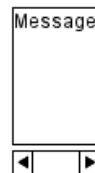
```
.
```

```
.
```

```
.
```

```
}
```

```
@endsalt
```



## 14.12 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
```





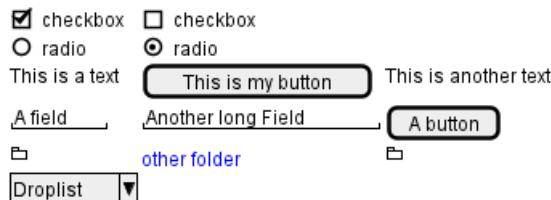
[Ref. QA-12177]

### 14.13 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
.....
.XXXX.....
.X...X.....
.XXXXXXXXXXX.
.X.....X.
.X.....X.
.X.....X.
.X.....X.
.XXXXXXXXXXX.

.....
>>|<color:blue>other folder|<<folder>>
^Dropelist^
}
@endsalt
```



[Ref. QA-5849]

### 14.14 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
```

<b>List Open Iconic</b>	▲ bell	▲ cloud	≡ excerpt	≡ justify-right	♪ musical-note	★ star
Credit to <a href="https://useiconic.com/open">https://useiconic.com/open</a>	✿ bluetooth	▲ cloudy	≡ expand-down	♫ key	☛ paperclip	☀ sun
B bold	▷ code	▷ bolt	▷ expand-left	▷ laptop	☛ pencil	▢ tablet
+ bolt	✿ cog	▷ collapse-down	▷ expand-right	▷ layers	☛ people	>tag
- account-login	■ book	▷ collapse-left	▷ expand-up	✿ lightbulb	☛ person	tags
- account-logout	■ bookmark	■ box	▷ collapse-right	○ eye	☛ phone	◎ target
↷ action-redo	■ briefcase	■ british-pound	▷ collapse-up	☛ eyedropper	☛ pie-chart	☒ task
↶ action-undo	■ command	■ comment-square	▷ command	■ file	☛ pin	terminal
≡ align-center	■ browser	■ compass	▷ fire	▷ list	● play-circle	text
≡ align-left	✓ brush	● contrast	▷ flag	▷ location	✚ plus	thumb-down
≡ align-right	✿ bug	■ copywriting	✿ flash	▷ lock-locked	○ power-standby	thumb-up
◐ aperture	✿ bullhorn	■ credit-card	■ folder	▷ lock-unlocked	☛ print	timer
↓ arrow-bottom	■ calculator	■ crop	☛ fork	▷ loop-circular	▷ project	transfer
◐ arrow-circle-bottom	■ calendar	○ dashboard	▷ fullscreen-enter	▷ loop-square	✚ pulse	trash
◐ arrow-circle-left	◐ camera-slr	▼ caret-bottom	▷ fullscreen-exit	▷ loop	☛ puzzle-piece	underline
◐ arrow-circle-right	◐ caret-bottom	▲ caret-left	▷ data-transfer-download	▷ magnifying-glass	?] question-mark	vertical-align-bottom
◐ arrow-circle-top	▼ caret-left	▼ caret-right	▷ data-transfer-upload	○ globe	✿ rain	vertical-align-center
↑ arrow-left	▼ caret-right	● delete	▷ delete	○ map-marker	✖ random	vertical-align-top
→ arrow-right	▲ caret-top	● dial	■ grid-four-up	○ map	● reload	video
↓ arrow-thick-bottom	■ caret-top	● document	■ grid-three-up	● media-pause	☛ media-both	volume-high
↑ arrow-thick-left	■ chat	✿ dollar	■ grid-two-up	▶ media-play	✚ resize-height	volume-low
→ arrow-thick-right	✓ check	“ double-quote-sans-left	■ hard-drive	● media-record	↑ resize-width	volume-off
↑ arrow-thick-top	▼ chevron-bottom	“ double-quote-sans-right	■ header	◀ media-skip-backward	▷ rss-alt	warning
↑ arrow-top	◀ chevron-left	“ double-quote-serif-left	○ headphones	▶ media-skip-forward	☛ rss	wifi
◐ audio-spectrum	▶ chevron-right	“ double-quote-serif-right	● heart	◀ media-step-backward	☛ script	wrench
◐ audio	▲ chevron-top	▲ droplet	● home	▶ media-step-forward	☛ share-boxed	x
✿ badge	● circle-check	▲ eject	■ image	● medical-cross	☛ share	yen
◐ ban	● circle-x	▲ elevator	■ inbox	≡ menu	● shield	zoom-in
■ bar-chart	■ clipboard	” ellipses	○ infinity	♀ microphone	■ signal	zoom-out
✿ basket	○ clock	■ envelope-closed	● italic	— minus	↑ signpost	
□ battery-empty	▲ cloud-download	■ envelope-open	▷ italic	☛ monitor	☛ sort-ascending	
■ battery-full	✿ cloud-upload	€ euro	≡ justify-center	● moon	◀ sort-descending	
✿ beaker			≡ justify-left	● move	■ spreadsheet	

## 14.15 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
{+
```

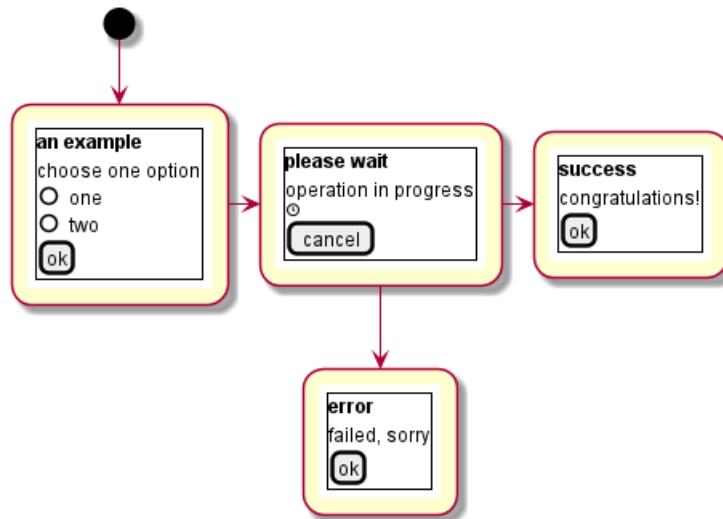


```

<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

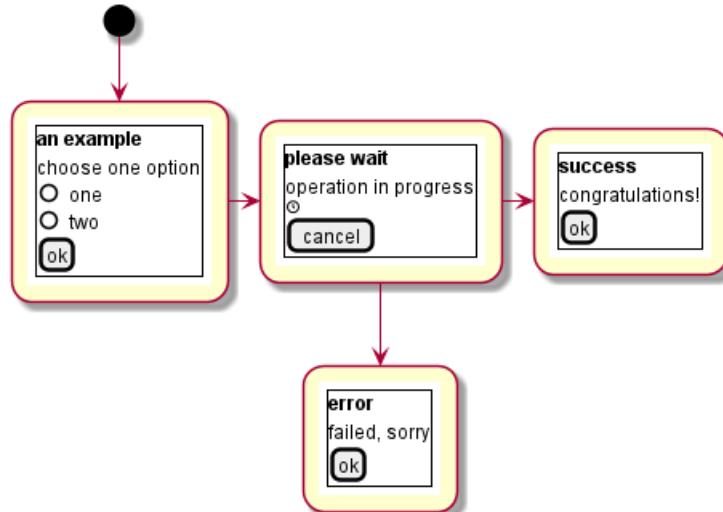
!procedure _wait()
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!endprocedure

!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

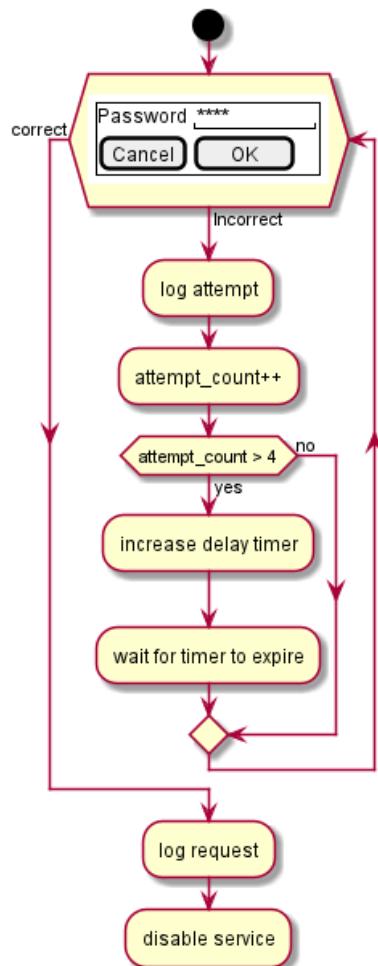
```



## 14.16 Include salt "on while condition of activity diagram"

You can include salt on while condition of activity diagram.

```
@startuml
start
while (\n{\nnsalt\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]



## 15 Archimate Diagram

This is only a proposal and subject to change.

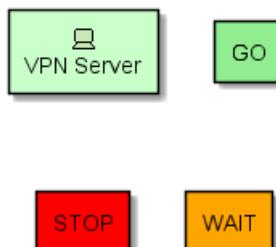
You are very welcome to create a new discussion on this future syntax. Your feedbacks, ideas and suggestions help us to find the right solution.

### 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 Defining Junctions

Using the `circle` keyword and the preprocessor, you can also create junctions.

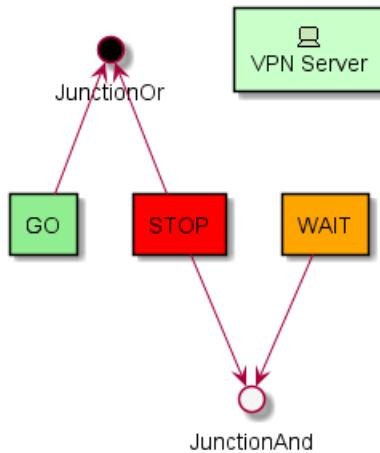
```
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

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 Example 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
  
```

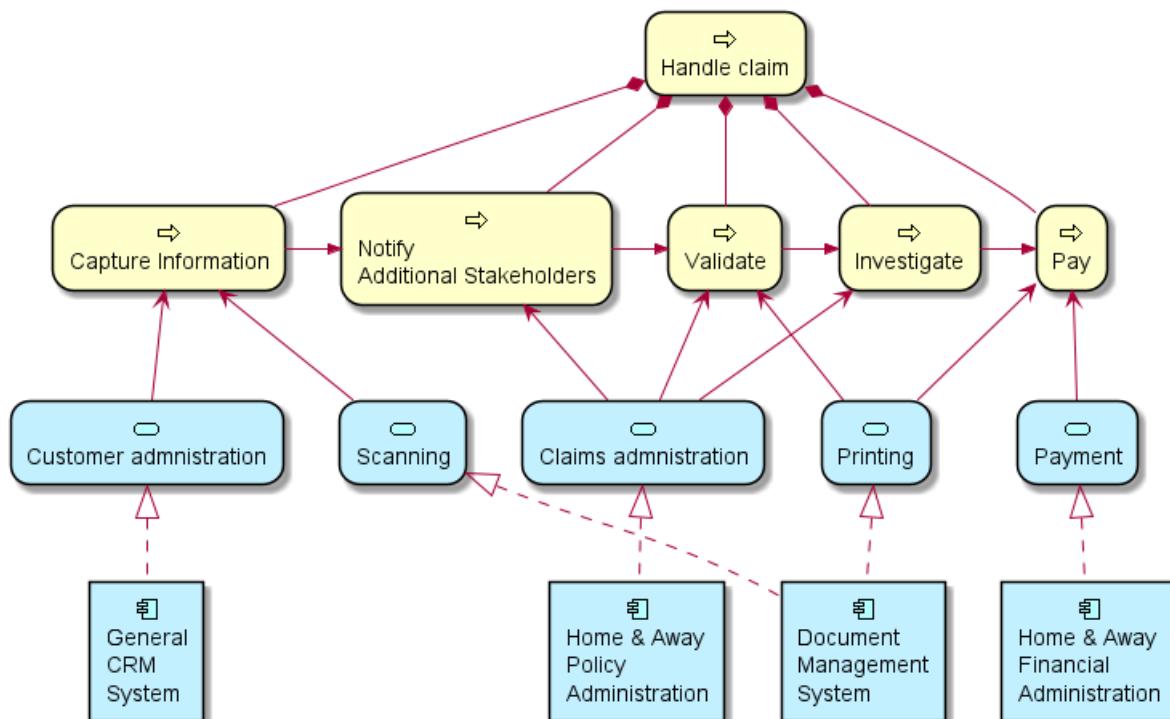
```

rectangle "Document\Management\System" as DMS <<$aComponent>> #Application
rectangle "General\CRM\System" as CRM <<$aComponent>> #Application
rectangle "Home & Away\Policy\Administration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\Financial\Administration" as HFPA <<$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

```



Example from the "Archisurance case study" (OpenGroup).
See
⇒ :business process
□ : application service
■ : application component

## 15.4 Example 2

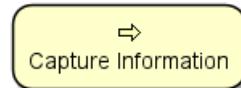
@startuml



```

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

```



## 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("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")
```

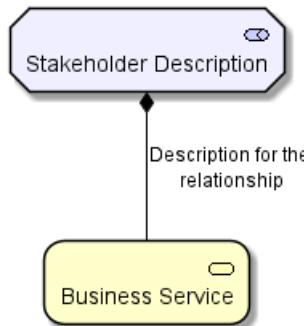
```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
```



```

Business_Service(BService, "Business Service")
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml

```

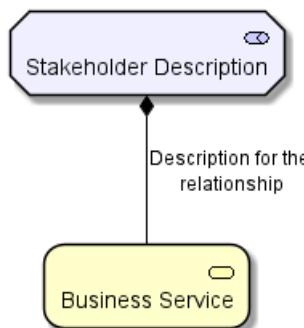


- Unordered List Item To orient the two elements in top - down position, the syntax will be

```

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

```

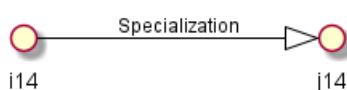
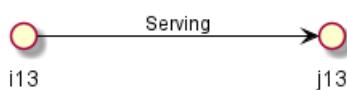
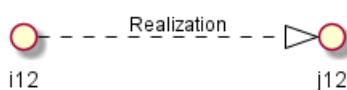
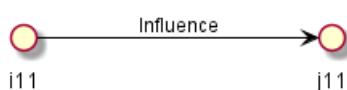
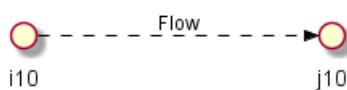
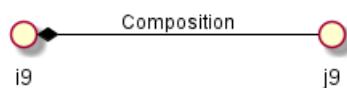
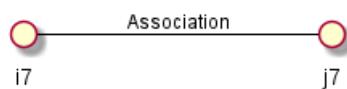
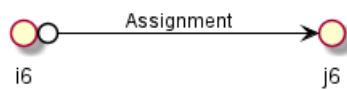
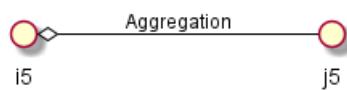
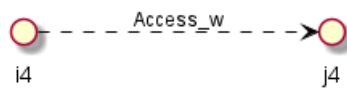
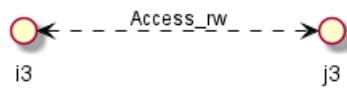
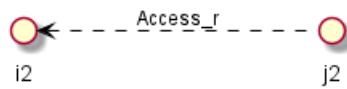
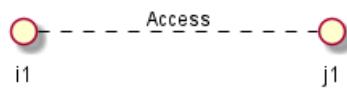


#### 15.6.4 Appendix: Examples of all Archimate RelationTypes

```

@startuml
left to right direction
!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

```



# 16 Gantt Diagram

The Gantt is described in *natural* language, using very simple sentences (subject-verb-complement).

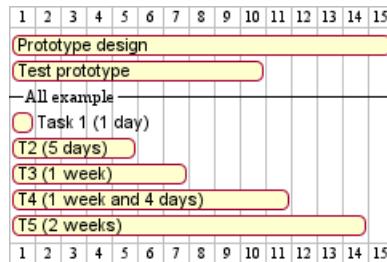
## 16.1 Declaring tasks

Tasks defined using square bracket.

### 16.1.1 Duration

Their durations are defined using the `lasts` verb:

```
@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
```

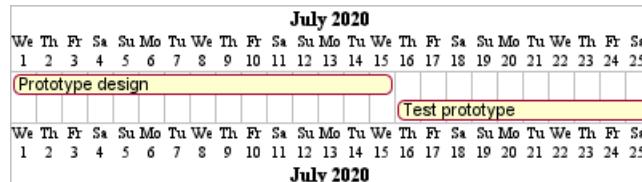


### 16.1.2 Start

Their beginning are defined using the `start` verb:

```
@startuml
[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
@enduml
```



### 16.1.3 End

Their ending are defined using the `end` verb:

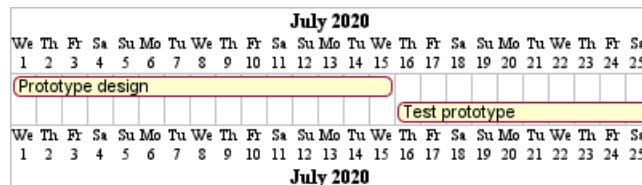
```
@startuml
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
```

```
Project starts 2020-07-01
[Prototype design] ends 2020-07-15
```



[Test prototype] ends 2020-07-25

@enduml



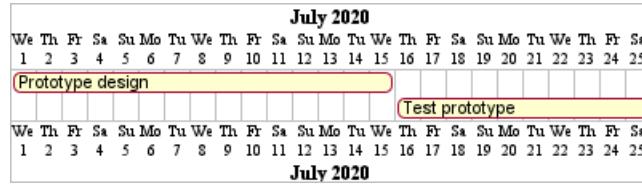
#### 16.1.4 Start/End

It is possible to define both absolutely, by specifying dates:

@startuml

```
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
```

@enduml

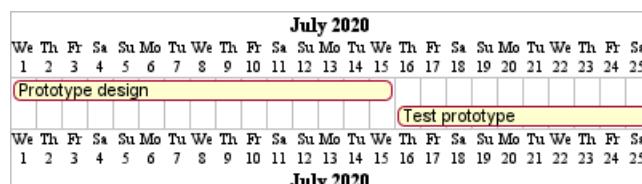


## 16.2 One-line declaration (with the and conjunction)

It is possible to combine declaration on one line with the `and` conjunction.

@startuml

```
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
@enduml
```



## 16.3 Adding constraints

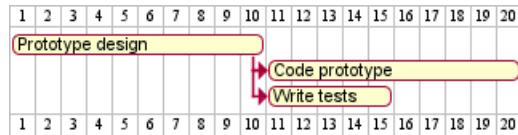
It is possible to add constraints between tasks.

@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 Short names

It is possible to define short name for tasks with the `as` keyword.

```
@startgantt
[Prototype design] as [D] lasts 15 days
[Test prototype] as [T] lasts 10 days
[T] starts at [D]'s end
@endgantt
```



## 16.5 Customize colors

It is also possible to customize colors with `is colored in`.

```
@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 Completion status

You can set the completion status of a task.

```
@startgantt
[foo] lasts 21 days
[foo] is 40% completed
[bar] lasts 30 days and is 10% complete
@endgantt
```



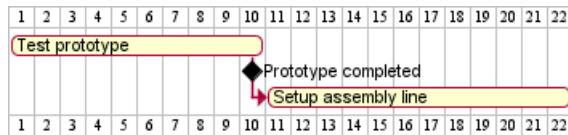
## 16.7 Milestone

You can define Milestones using the `happen` verb.



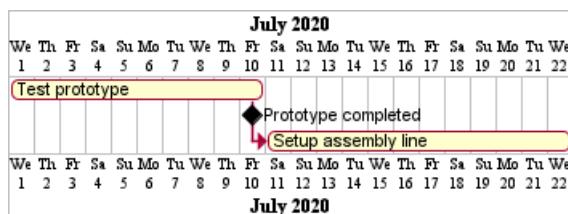
### 16.7.1 Relative milestone (use of constraints)

```
@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 Absolute milestone (use of fixed date)

```
@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
```



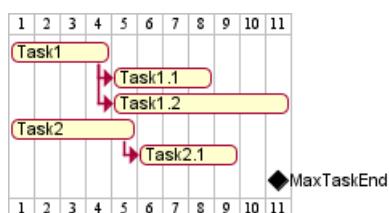
### 16.7.3 Milestone of maximum end of tasks

```
@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 Hyperlinks

You can add hyperlinks to tasks.



```
@startgantt
[task1] lasts 10 days
[task1] links to [[http://plantuml.com]]
@endgantt
```



## 16.9 Calendar

You can specify a starting date for the whole project. By default, the first task starts at this date.

```
@startgantt
Project starts the 20th of september 2017
[Prototype design] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue
@endgantt
```



## 16.10 Coloring days

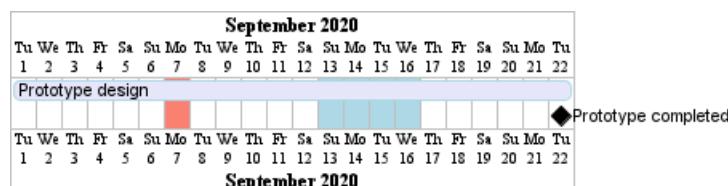
It is possible to add colors to some days.

```
@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 Changing scale

You can change scale for very long project, with one of those parameters:

- printscale
- ganttscale
- projectscale

and one of the values:

- daily (*by default*)
- weekly
- monthly

(See QA-11272, QA-9041 and QA-10948)

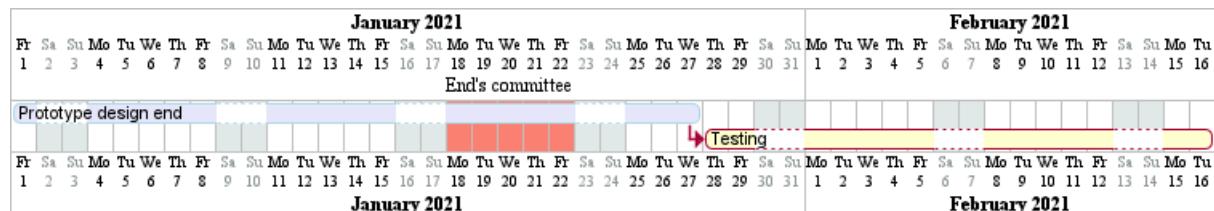


### 16.11.1 Daily (*by default*)

```
@startuml  
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  
@endum1



### 16.11.2 Weekly

```
@startuml  
printscale weekly  
saturday are closed  
sunday are closed
```

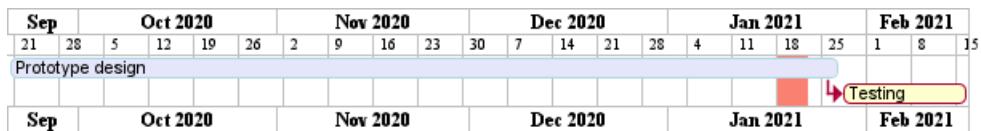
Project starts the 1st of january 2021  
[Prototype design end] as [TASK1] lasts 19 days  
[TASK1] is colored in Lavender/LightBlue  
[Testing] lasts 14 days  
[TASK1]->[Testing]

```
2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@enduml
```



```
@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 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]
```

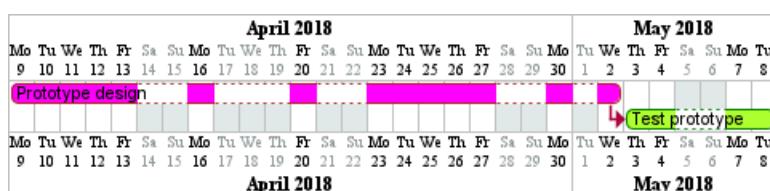
2021-01-18 to 2021-01-22 are named [End's committee]  
 2021-01-18 to 2021-01-22 are colored in salmon  
 @endgantt



## 16.12 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
```

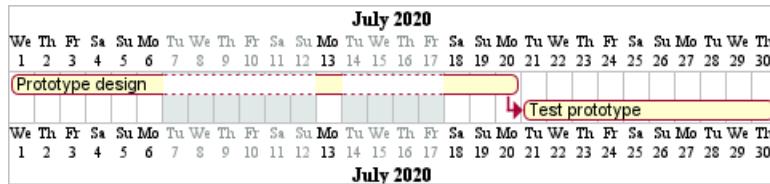


Then it is possible to open some closed day.

```
@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

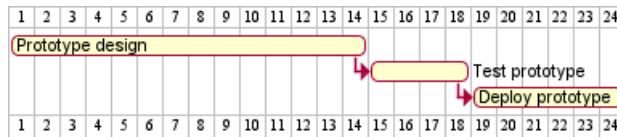




### 16.13 Simplified task succession

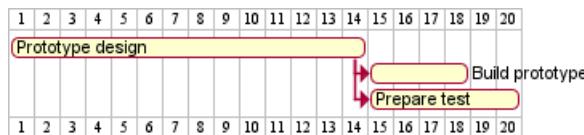
It's possible to use the `then` keyword to denote consecutive tasks.

```
@startgantt
[Prototype design] lasts 14 days
then [Test prototype] lasts 4 days
then [Deploy prototype] lasts 6 days
@endgantt
```



You can also use arrow `->`

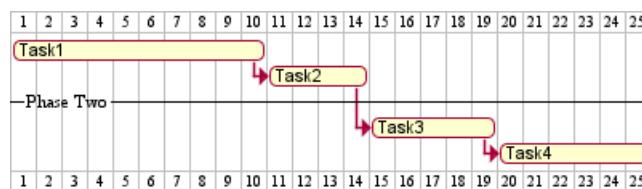
```
@startgantt
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -> [Build prototype]
[Prototype design] -> [Prepare test]
@endgantt
```



### 16.14 Separator

You can use `--` to separate sets of tasks.

```
@startgantt
[Task1] lasts 10 days
then [Task2] lasts 4 days
-- Phase Two --
then [Task3] lasts 5 days
then [Task4] lasts 6 days
@endgantt
```



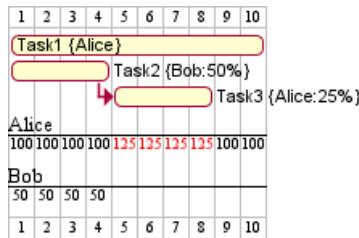
### 16.15 Working with resources

You can affect tasks on resources using the `on` keyword and brackets for resource name.

```
@startgantt
[Task1] on {Alice} lasts 10 days
```

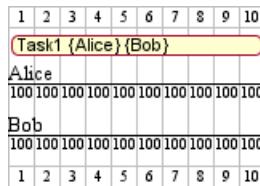


```
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
@endgantt
```



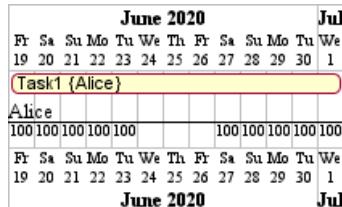
Multiple resources can be assigned to a task:

```
@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
```

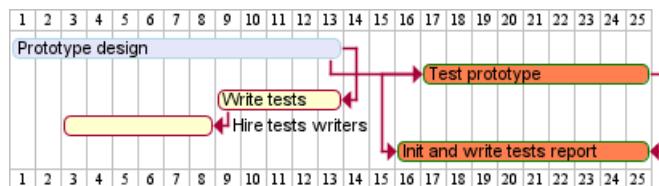


## 16.16 Complex example

It is 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 design]
[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 prototype]
@endgantt
```



## 16.17 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 '*)

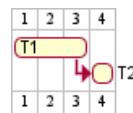
```
@startgantt
' This is a comment
```

```
[T1] lasts 3 days
```

```
/' this comment
is on several lines '/
```

```
[T2] starts at [T1]'s end and lasts 1 day
```

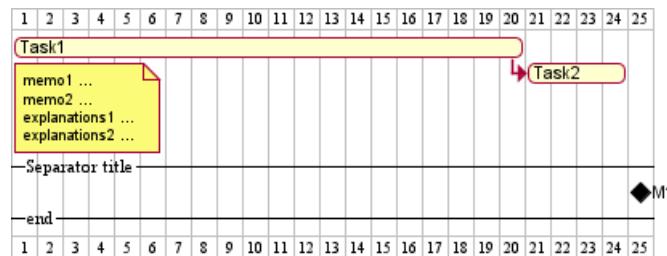
```
@endgantt
```



## 16.18 Using style

### 16.18.1 Without style (by default)

```
@startuml
[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 --
@enduml
```



### 16.18.2 With style

You can use style to change rendering of elements.

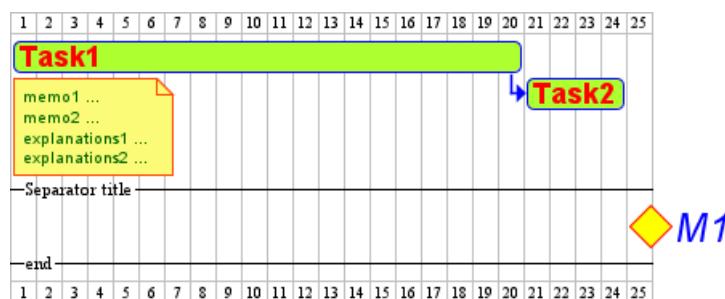
```
@startuml
<style>
ganttDiagram {
task {
FontName Helvetica
FontColor red
FontSize 18
}}
```



```

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 --
@enduml

```



[Ref. QA-10835, QA-12045, QA-11877 and PR-438]

**TODO:** TODO Awaiting style for Separator and all style for Arrow (thickness...)

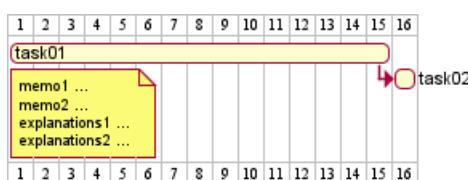


### 16.19 Add notes

```
@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note
```

[task01] -> [task02]

@endgantt

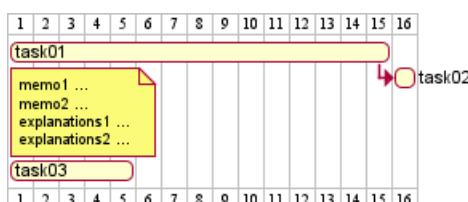


Example with overlap.

```
@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note
```

[task01] -> [task02]
[task03] lasts 5 days

@endgantt



@startgantt

-- test01 --

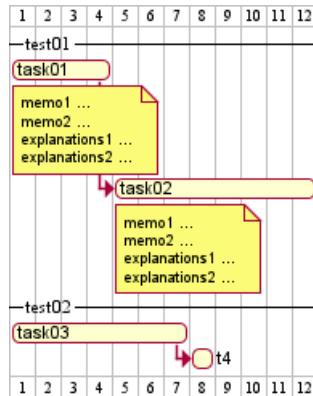
```
[task01] lasts 4 days
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note
```

```
[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:** DONE *Thanks for correction (of #386 on v1.2020.18) when overlapping*

@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
```

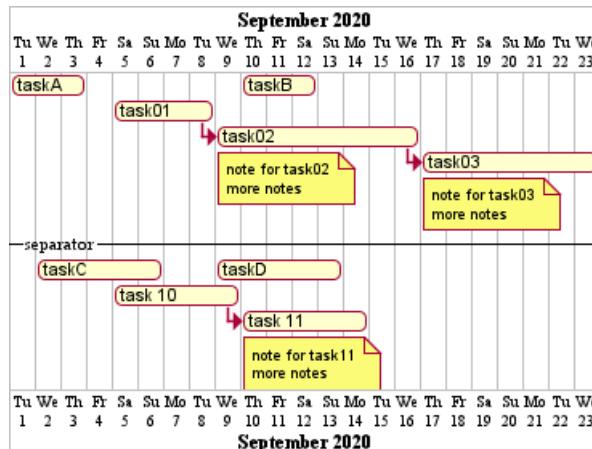
```
then [task03] lasts 7 days
note bottom
    note for task03
    more notes
end note
```

-- 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]
```

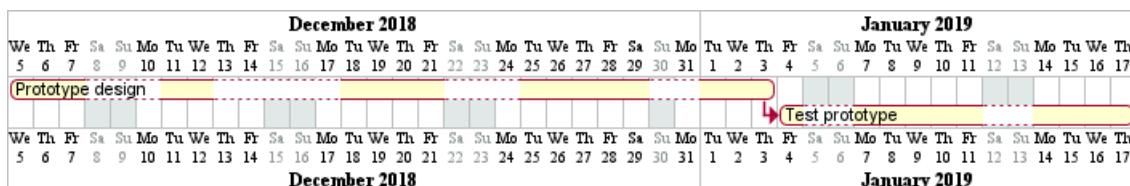
```
[task 10] starts 2020-09-05 and lasts 5 days  
then [task 11] lasts 5 days  
note bottom  
    note for task11
```

```
more notes
end note
@endgantt
```



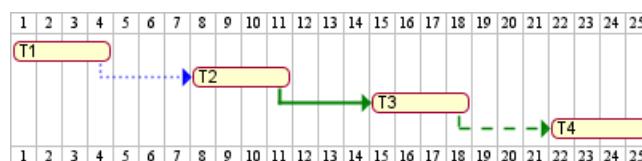
## 16.20 Pause tasks

```
@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
```



## 16.21 Change link colors

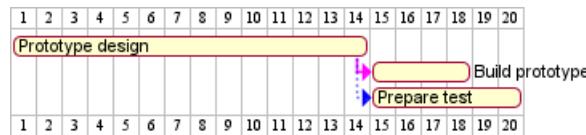
```
@startgantt
[T1] lasts 4 days
[T2] lasts 4 days and starts 3 days after [T1]'s end with blue dotted link
[T3] lasts 4 days and starts 3 days after [T2]'s end with green bold link
[T4] lasts 4 days and starts 3 days after [T3]'s end with green dashed link
@endgantt
```



```
@startuml
Links are colored in blue
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
```

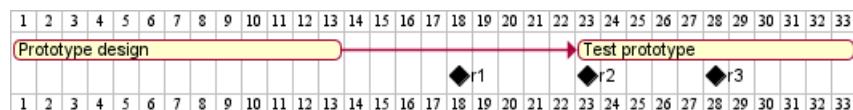


```
[Prototype design] -[#FF00FF]-> [Build prototype]
[Prototype design] -[dotted]-> [Prepare test]
@enduml
```



## 16.22 Tasks or Milestones on the same line

```
@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
```

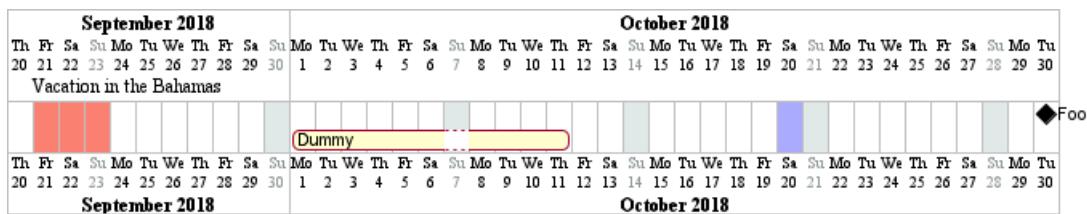


## 16.23 Highlight today

```
@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]
```

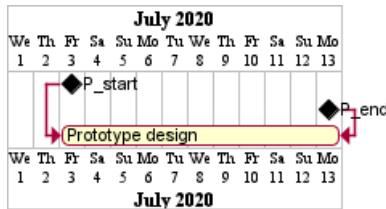
today is 30 days after start and is colored in #AAFF  
[Foo] happens 40 days after start  
[Dummy] lasts 10 days and starts 10 days after start

@endgantt



## 16.24 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
```



## 16.25 Grammar and verbal form

Verbal form	Example
[T] starts	
[M] happens	

## 16.26 Add title, header, footer, caption or legend on gantt diagram

```
@startuml
```

```
header some header
```

```
footer some footer
```

```
title My title
```

```
[Prototype design] lasts 13 days
```

```
legend
```

```
The legend
```

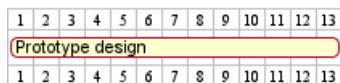
```
end legend
```

```
caption This is caption
```

```
@enduml
```

some header

**My title**



This is caption

some footer

(See also: Common commands)

## 16.27 Removing Foot Boxes

You can use the `hide footbox` keywords to remove the foot boxes of the gantt diagram (as for sequence diagram).

Examples on:

- daily scale (*without project start*)

```
@startgantt
```

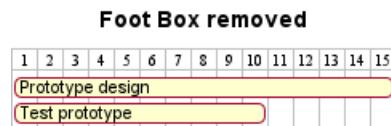
```
hide footbox
```

```
title Foot Box removed
```

```
[Prototype design] lasts 15 days  
[Test prototype] lasts 10 days
```



```
@endgantt
```



- daily scale

```
@startgantt
```

Project starts the 20th of september 2017  
 [Prototype design] as [TASK1] lasts 13 days  
 [TASK1] is colored in Lavender/LightBlue

```
hide footbox
```

```
@endgantt
```



- weekly scale

```
@startgantt
```

```
hide footbox
```

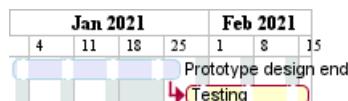
printscale weekly  
 saturday are closed  
 sunday are closed

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
```



- monthly scale

```
@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
```



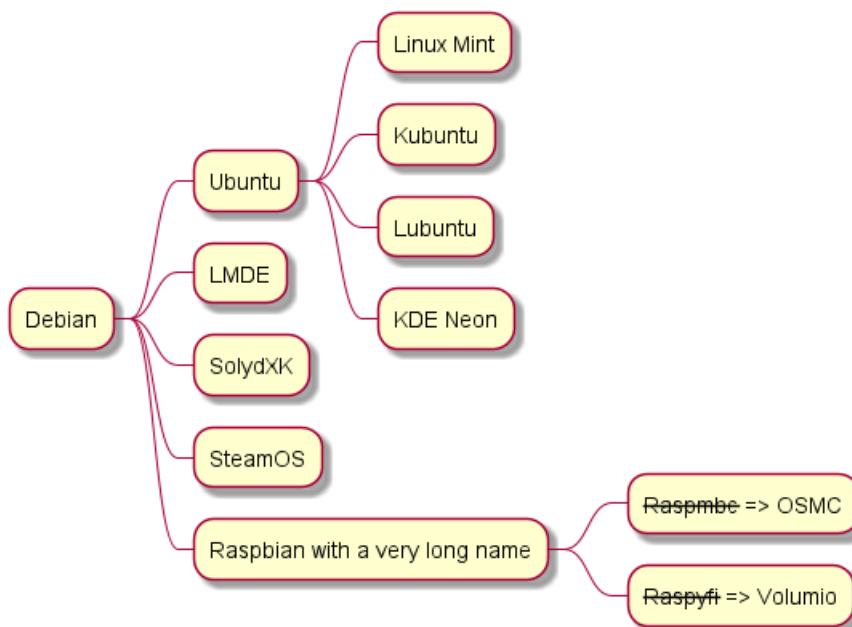
## 17 思维导图

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
```

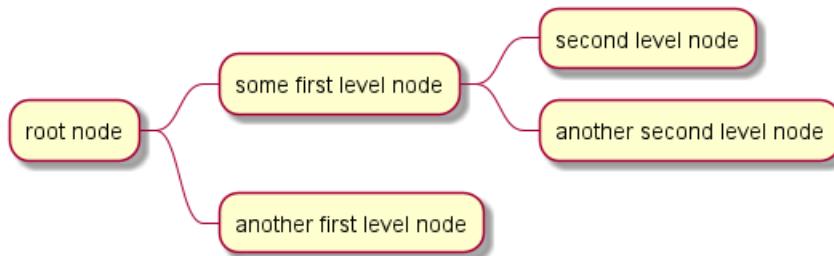


### 17.2 Markdown 语法

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
```



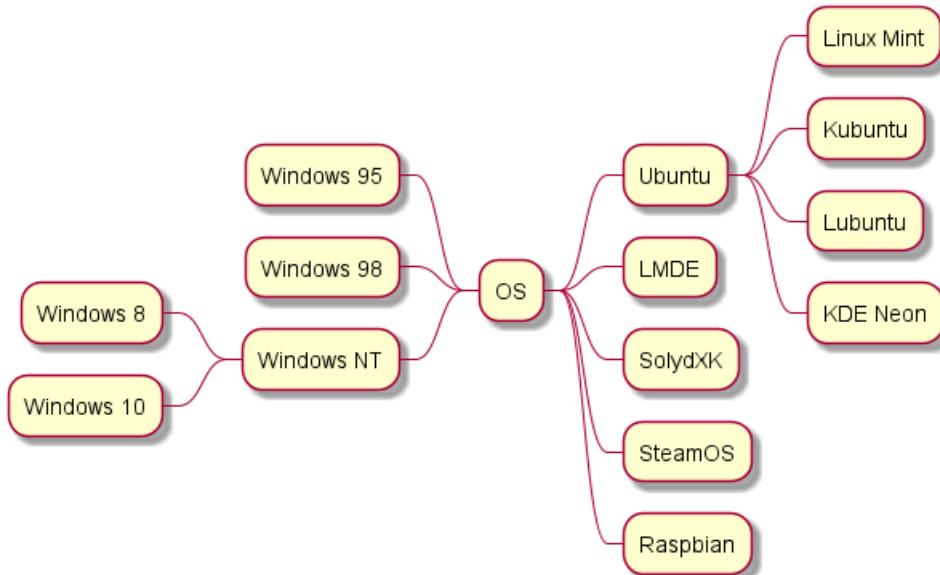


### 17.3 运算符

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 Multilines

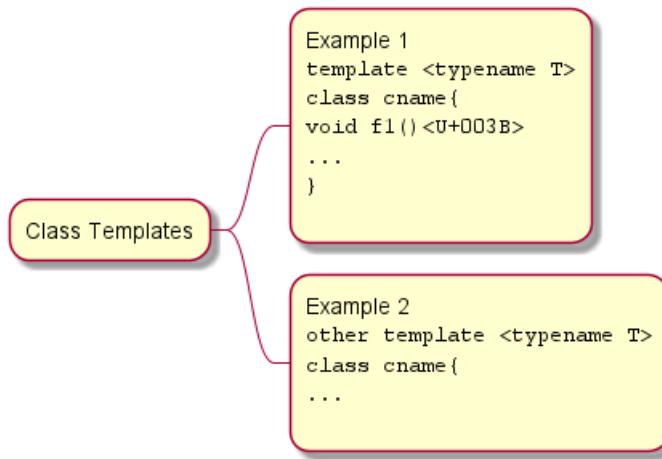
You can use : and ; to have multilines box.

```

@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
```



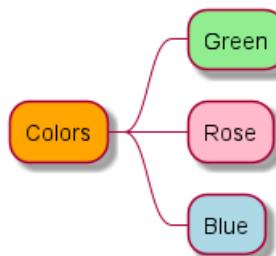
## 17.5 Colors

It is possible to change node color.

### 17.5.1 With inline color

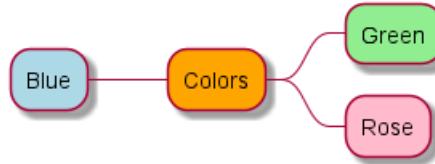
- OrgMode syntax mindmap

```
@startmindmap
*[#Orange] Colors
**[#lightgreen] Green
**[#FFBBCC] Rose
**[#lightblue] Blue
@endmindmap
```



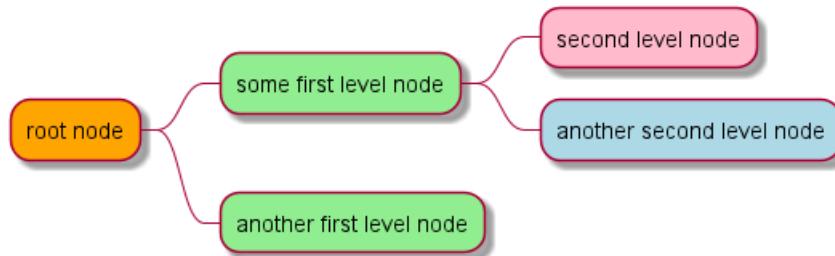
- Arithmetic notation syntax mindmap

```
@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
```

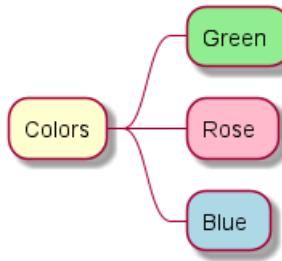


### 17.5.2 With style color

- 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
```

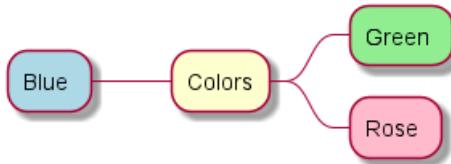




- Arithmetic notation syntax mindmap

```

@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
  
```



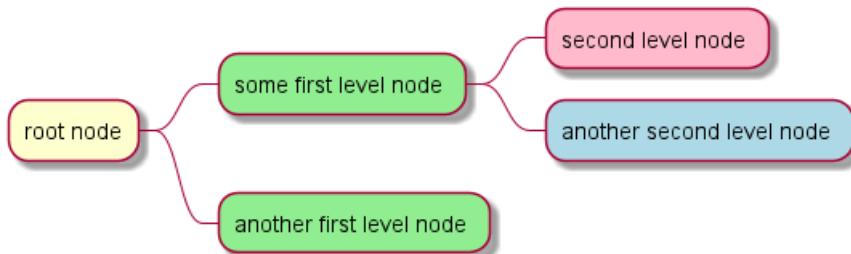
- Markdown syntax mindmap

```

@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>>
  
```



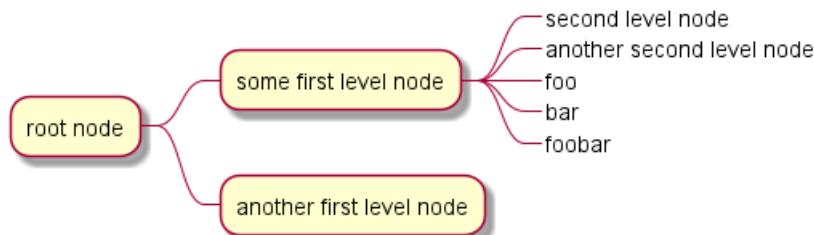
```
* another first level node <<green>>
@endmindmap
```



## 17.6 去除外边框

你可以用下划线去除外边框。

```
@startmindmap
* root node
** some first level node
*** _ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
** another first level node
@endmindmap
```



## 17.7 改变图形方向

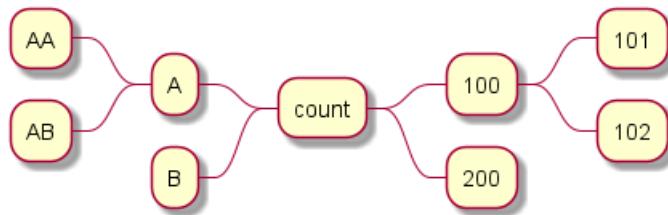
你可以同时使用图形的左右两侧。

```
@startmindmap
* count
** 100
*** 101
*** 102
** 200

left side

** A
*** AA
*** AB
** B
@endmindmap
```





## 17.8 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
```

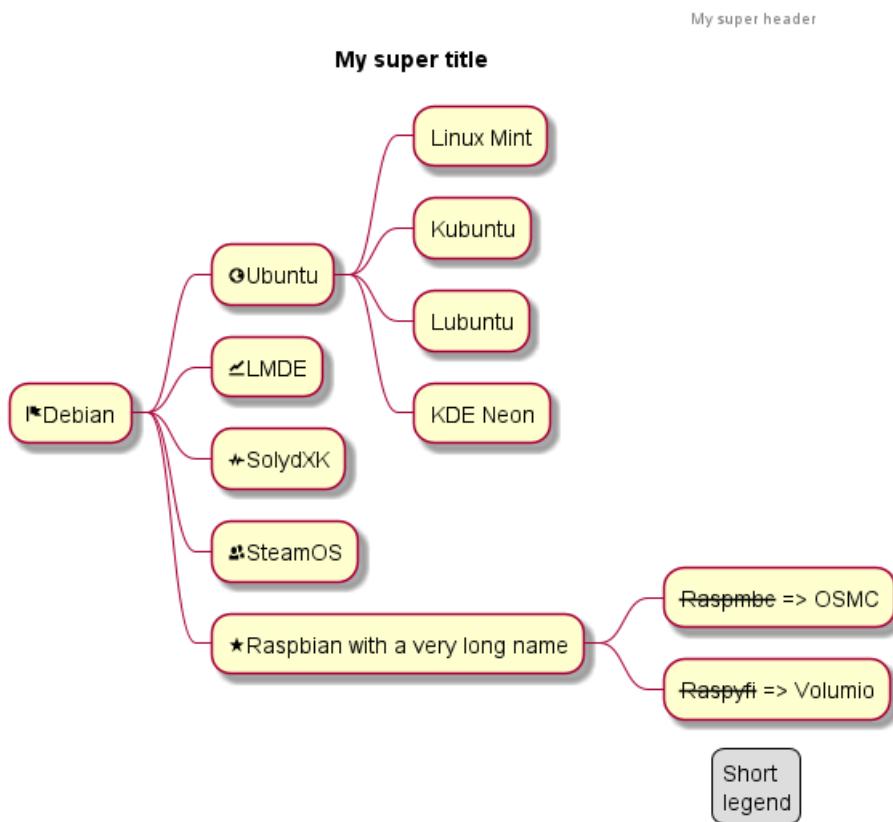


figure 1  
My super footer

## 17.9 Changing style

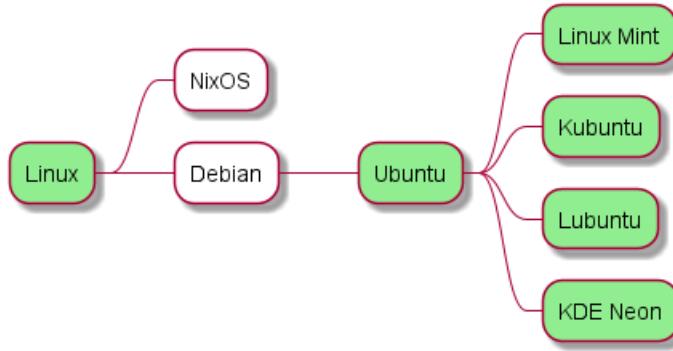
### 17.9.1 node, depth

```

@startmindmap
<style>
mindmapDiagram {
    node {
        BackgroundColor lightGreen
    }
    :depth(1) {
        BackGroundColor white
    }
}
</style>
* Linux
** NixOS
** Debian
*** Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap

```

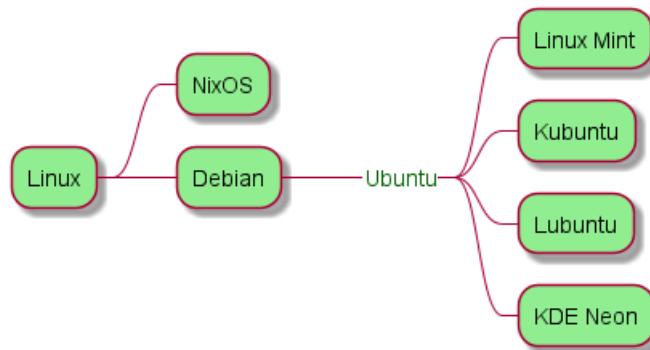




### 17.9.2 boxless

```

@startmindmap
<style>
mindmapDiagram {
    node {
        BackgroundColor lightGreen
    }
    boxless {
        FontColor darkgreen
    }
}
</style>
* Linux
** NixOS
** Debian
*** _ Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap
  
```



## 17.10 Word Wrap

Using `MaximumWidth` setting you can control automatic word wrap. Unit used is pixel.

```
@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
}

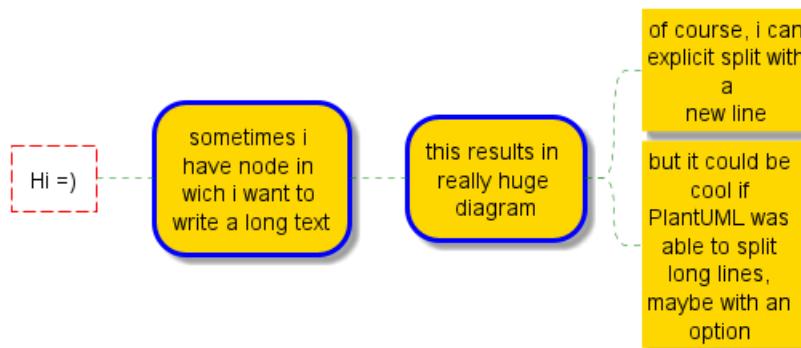
leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in which 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

```



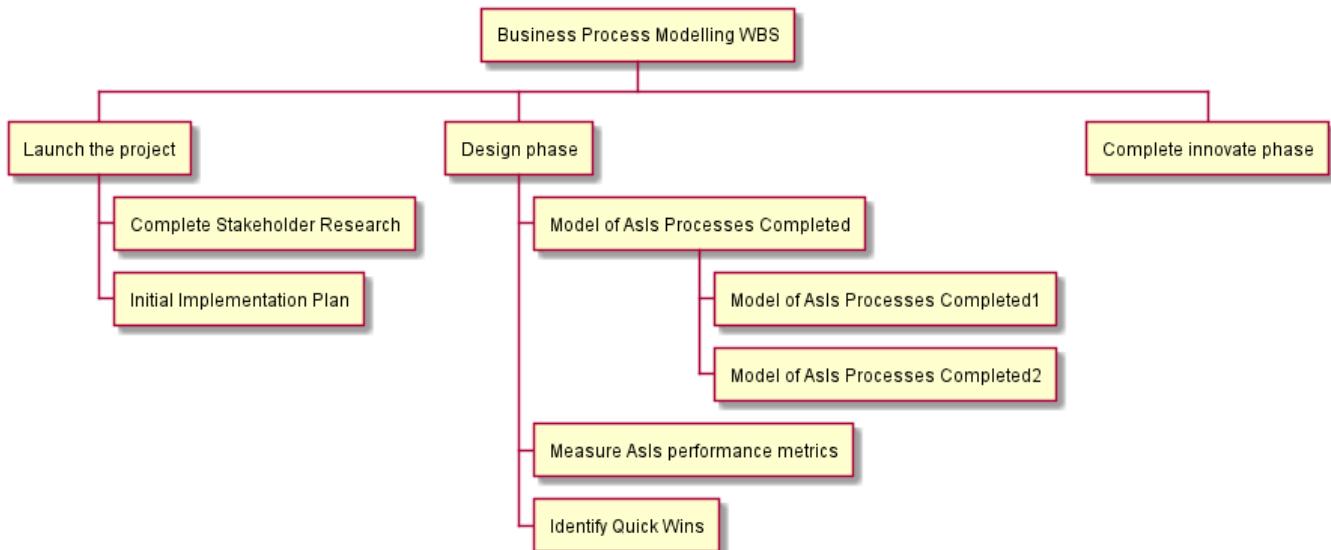
## 18 Work Breakdown Structure (WBS)

WBS diagram are still in beta: the syntax may change without notice.

### 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
```

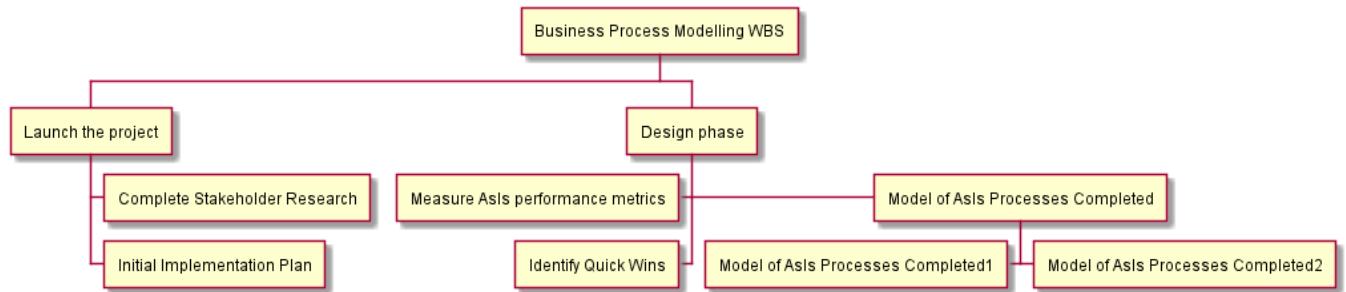


### 18.2 Change direction

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
```



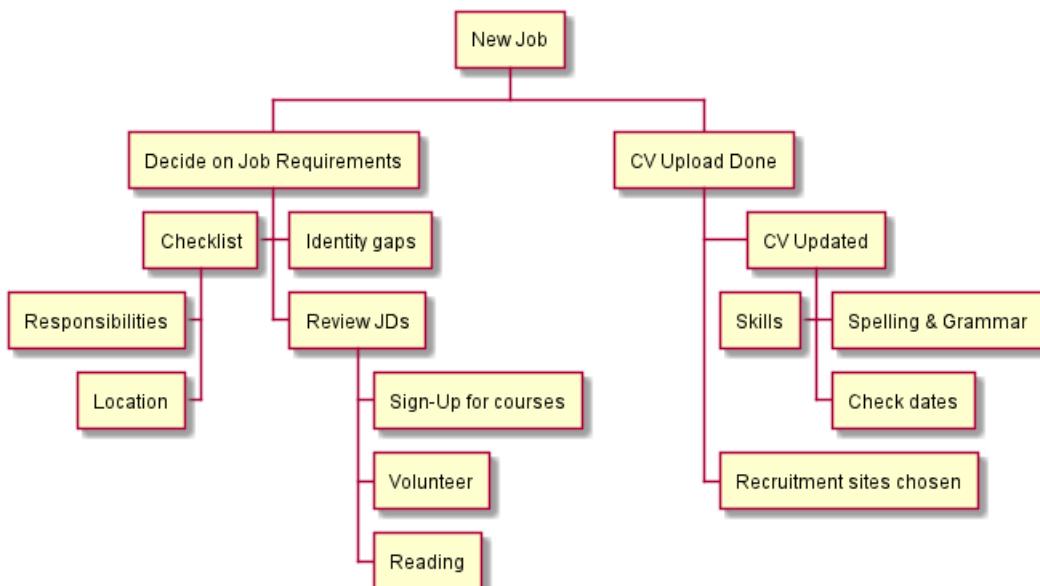


### 18.3 Arithmetic notation

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 Removing box

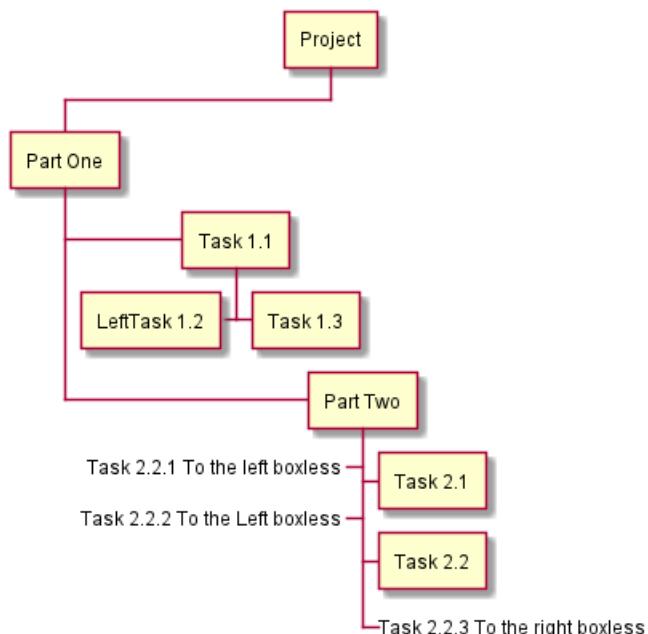
You can use underscore \_ to remove box drawing.



### 18.4.1 Boxless on Arithmetic notation

#### 18.4.2 Several boxless node

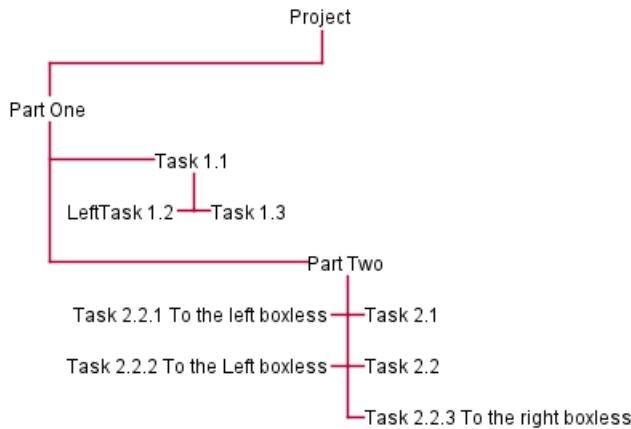
```
@startwbs
+ Project
+ Part One
+ Task 1.1
- LeftTask 1.2
+ Task 1.3
+ Part Two
+ Task 2.1
+ Task 2.2
-_ Task 2.2.1 To the left boxless
-_ Task 2.2.2 To the Left boxless
+_ Task 2.2.3 To the right boxless
@endwbs
```



#### 18.4.3 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
```

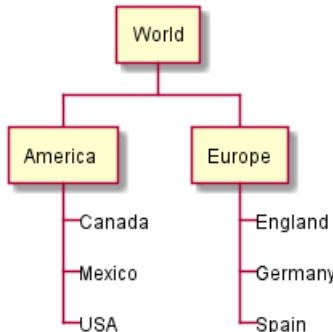




#### 18.4.4 Boxless on OrgMode syntax

#### 18.4.5 Several boxless node

```
@startwbs
* World
** America
*** _ Canada
*** _ Mexico
*** _ USA
** Europe
*** _ England
*** _ Germany
*** _ Spain
@endwbs
```

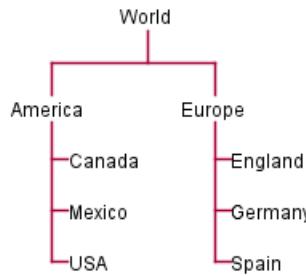


[Ref. QA-13297]

#### 18.4.6 All boxless node

```
@startwbs
*_ World
**_ America
***_ Canada
***_ Mexico
***_ USA
**_ Europe
***_ England
***_ Germany
***_ Spain
@endwbs
```





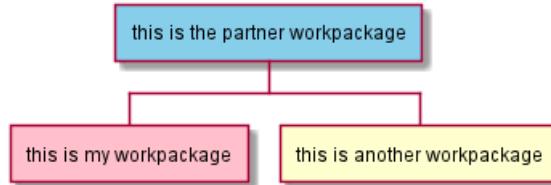
[Ref. QA-13355]

## 18.5 Colors (with inline or style color)

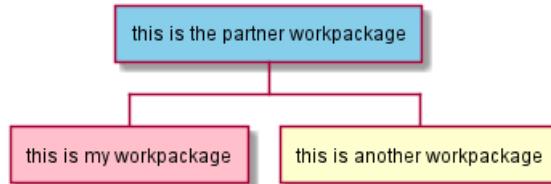
It is possible to change node color:

- 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
```



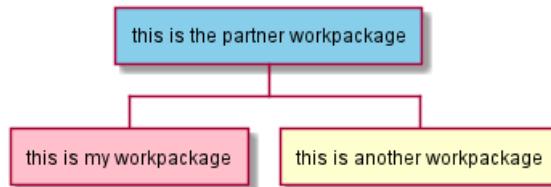
[Ref. QA-12374, only from v1.2020.20]

- with style color

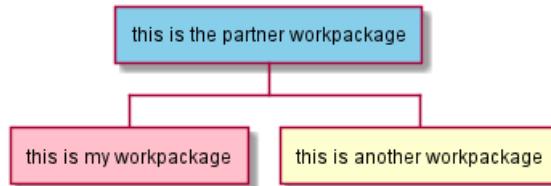
```
@startwbs
<style>
wbsDiagram {
    .pink {
        BackgroundColor pink
    }
    .your_style_name {
        BackgroundColor SkyBlue
    }
}
</style>
* this is the partner workpackage <<your_style_name>>
```



```
** 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
```



## 18.6 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) {

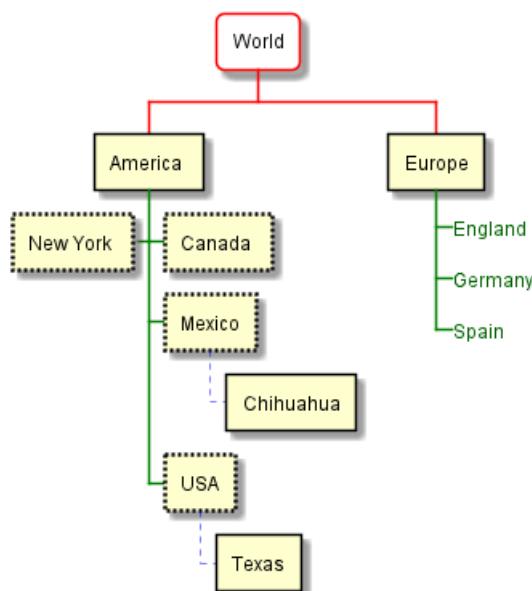
```



```

// 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

```



## 18.7 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
}

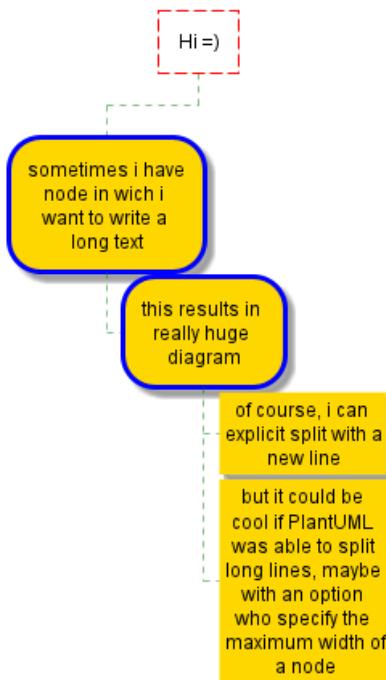
leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in which i want to write a long text
*** this results in really huge diagram
**** of course, i can explicitly split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option who specifies

@endwbs
```

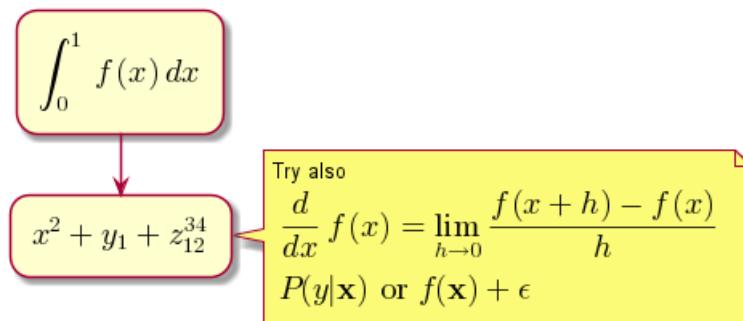




## 19 简介

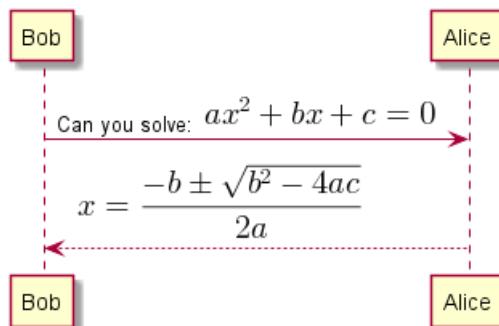
您可以在 PlantUML 中用 AsciiMath 或 JLaTeXMath 符号:

```
@startuml
:<math>\int_0^1 f(x) dx</math>;
:<math>x^2 + y_1 + z_{12}^{34}</math>;
note right
Try also
<math>d/dx f(x) = \lim_{h \rightarrow 0} (f(x+h) - f(x))/h</math>
<math>P(y|\mathbf{x}) \text{ or } f(\mathbf{x}) + \epsilon</math>
end note
@enduml
```



或:

```
@startuml
Bob --> Alice : Can you solve: <math>ax^2+bx+c=0</math>
Alice --> Bob: <math>x = (-b \pm \sqrt{b^2-4ac})/(2a)</math>
@enduml
```



### 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)^oo a_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?

要绘制此公式, 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 Entity Relationship Diagram

Based on the Information Engineering notation.

This is an extension to the existing Class Diagram. This extension adds:

- Additional relations for the Information Engineering notation.
- An `entity` alias that maps to the class diagram `class`.
- An additional visibility modifier `*` to identify mandatory attributes.

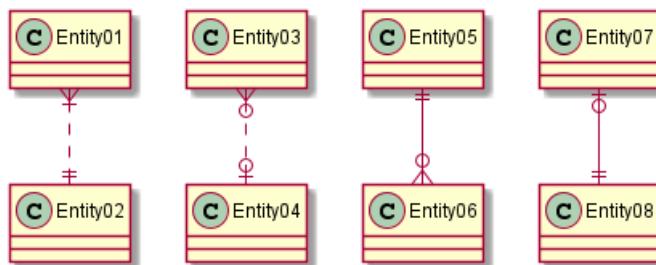
Otherwise, the syntax for drawing diagrams is the same as for class diagrams. All other features of class diagrams are also supported.

### 20.1 Information Engineering Relations

Type	Symbol
Zero or One	o--
Exactly One	--
Zero or Many	}o--
One or Many	} --

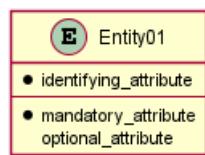
Examples:

```
@startuml
Entity01 }|...|| Entity02
Entity03 }o..o| Entity04
Entity05 ||--o{ Entity06
Entity07 |o--|| Entity08
@enduml
```



### 20.2 Entities

```
@startuml
entity Entity01 {
    * identifying_attribute
    --
    * mandatory_attribute
    optional_attribute
}
@enduml
```



Again, this is the normal class diagram syntax (aside from use of `entity` instead of `class`). Anything that you can do in a class diagram can be done here.

The `*` visibility modifier can be used to identify mandatory attributes. A space can be used after the modifier character to avoid conflicts with the creole bold:



```
@startuml
entity Entity01 {
    optional attribute
    **optional bold attribute**
    * **mandatory bold attribute**
}
@enduml
```



### 20.3 Complete Example

```
@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
}

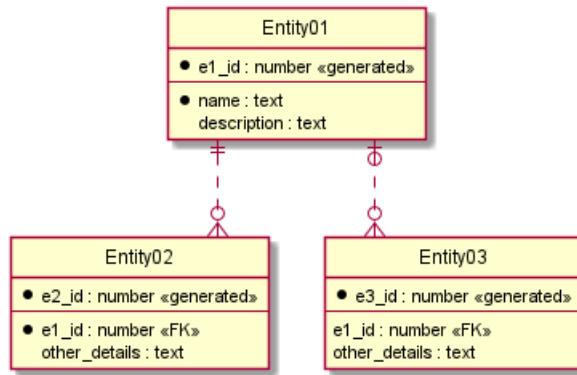
entity "Entity02" as e02 {
    *e2_id : number <<generated>>
    --
    *e1_id : number <<FK>>
    other_details : text
}

entity "Entity03" as e03 {
    *e3_id : number <<generated>>
    --
    e1_id : number <<FK>>
    other_details : text
}

e01 ||..o{ e02
e01 |o..o{ e03

@enduml
```





Currently the crows feet do not look very good when the relationship is drawn at an angle to the entity. This can be avoided by using the `linetype ortho` skinparam.



## 21 通用命令

### 21.1 注释

所有以单引号'开头的语句，被认为是一个注释。

多行注释，以'/'和'/'作为注释的开始和结束

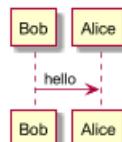
### 21.2 缩放

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 \n 添加新行。

Some `skinparam` settings are available to put borders on the title.

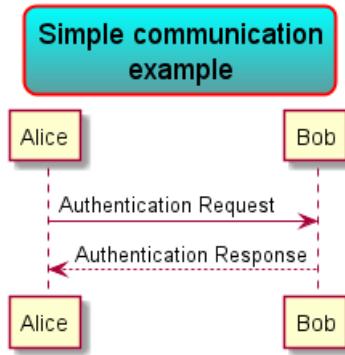
```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue

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
```

```

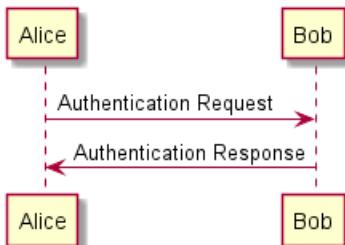
title
<u>Simple</u> communication example
on <i>several</i> lines and using <back:cadetblue>creole tags</back>
end title
  
```

```

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

```
@enduml
```

### Simple communication example on *several* lines and using creole tags



## 21.4 图片标题

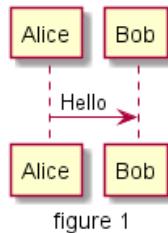
使用 `caption` 关键字在图像下放置一个标题。

```
@startuml
```

```

caption figure 1
Alice -> Bob: Hello
  
```

```
@enduml
```



## 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。

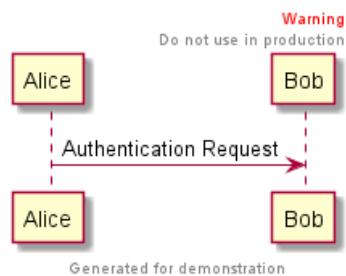
和标题一样，页眉或页脚内容可以在多行中定义，而且同样可以在页眉或页脚中输入一些 HTML 代码。

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

header
<font color=red>Warning:</font>
Do not use in production.
endheader

center footer Generated for demonstration

@enduml
```

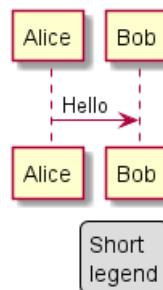


## 21.6 图例说明

`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
```



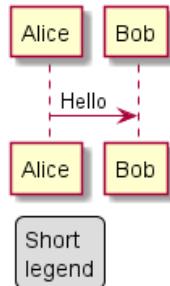
```
@startuml
```

```
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
```

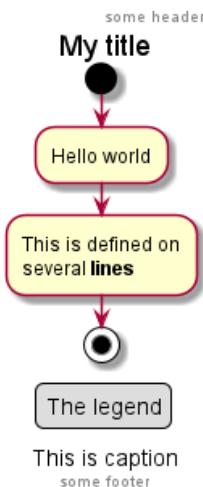
```
title My title
```

```
caption This is caption
```

```
legend
The legend
end legend
```

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

```
@enduml
```



This is caption  
some footer

### 21.7.2 Archimate

```
@startuml
```



```

header some header

footer some footer

title My title

caption This is caption

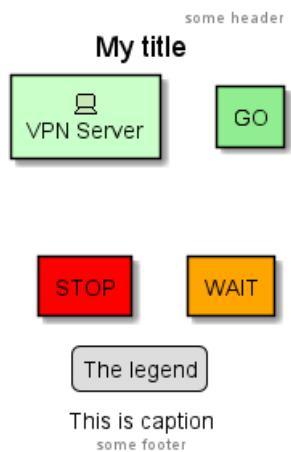
legend
The legend
end legend

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

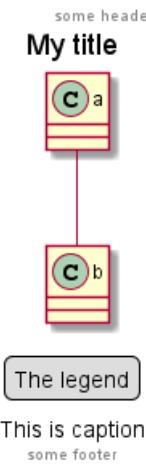
legend
The legend
end legend

a -- b

@enduml

```





#### 21.7.4 Component, Deployment, Use-Case

```
@startuml
header some header

footer some footer

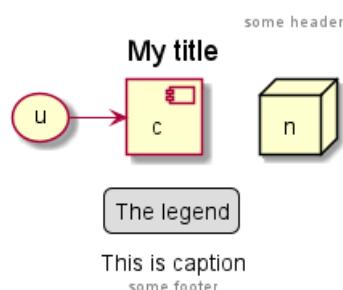
title My title

caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml
```



#### 21.7.5 Gantt project planning

```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
```



```
end legend
```

```
[t] lasts 5 days
```

```
@enduml
```

some header

### My title

1	2	3	4	5
t				
1	2	3	4	5

The legend

This is caption

some footer

**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
```

some header

### My title

user
name = "Dummy"

The legend

This is caption

some footer

#### 21.7.7 MindMap

```
@startmindmap
header some header
```

```
footer some footer
```

```
title My title
```

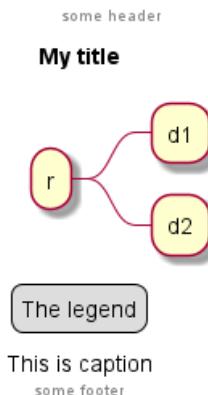


```
caption This is caption
```

```
legend
The legend
end legend
```

```
* r
** d1
** d2
```

```
@endmindmap
```



### 21.7.8 Network (nwdiag)

```
@startuml
header some header
```

```
footer some footer
```

```
title My title
```

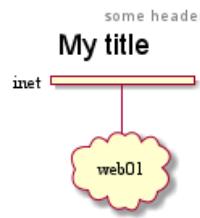
```
caption This is caption
```

```
legend
The legend
end legend
```

```
nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
```

```
@enduml
```





The legend

This is caption  
some footer

### 21.7.9 Sequence

```
@startuml
```

```
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
```

```
The legend
```

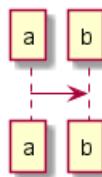
```
end legend
```

```
a->b
```

```
@enduml
```

some header

My title



The legend

This is caption

some footer

### 21.7.10 State

```
@startuml
```

```
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
```

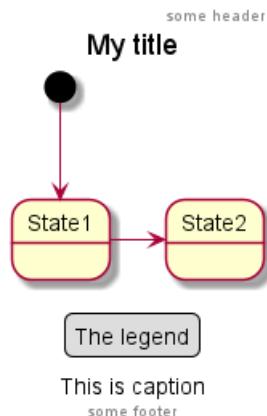
```
The legend
```

```
end legend
```



```
[*] --> State1  
State1 -> State2
```

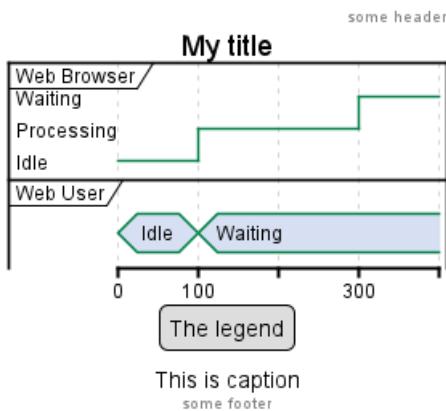
```
@enduml
```



### 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

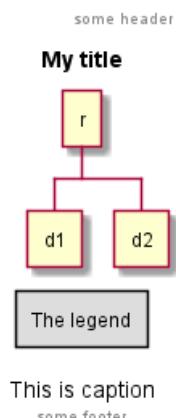
title My title

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
```

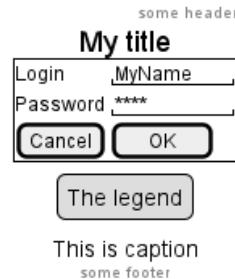


```

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

```

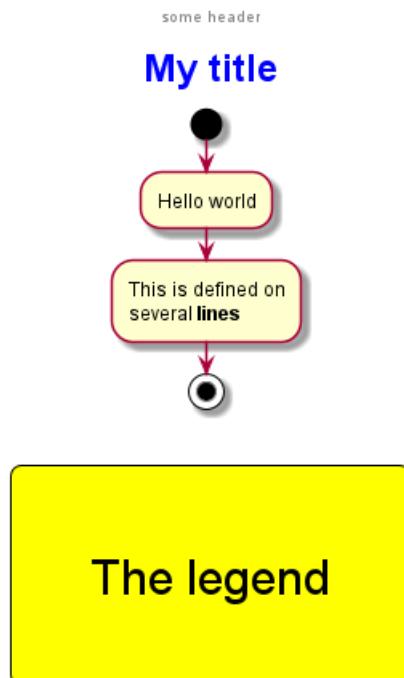


```

start
:Hello world;
:This is defined on
several **lines**;
stop

```

```
@enduml
```



## This is caption

some footer

### 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

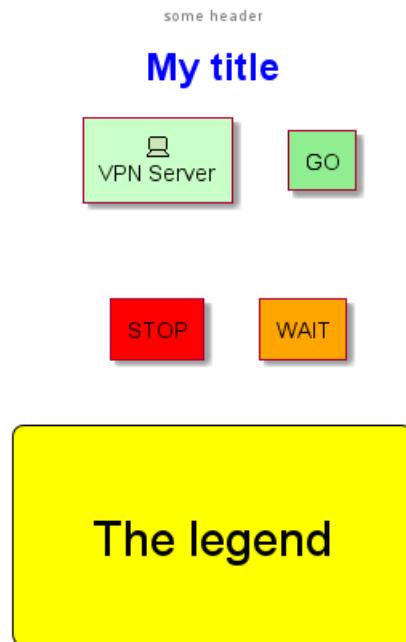
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml

```



This is caption

some footer



### 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

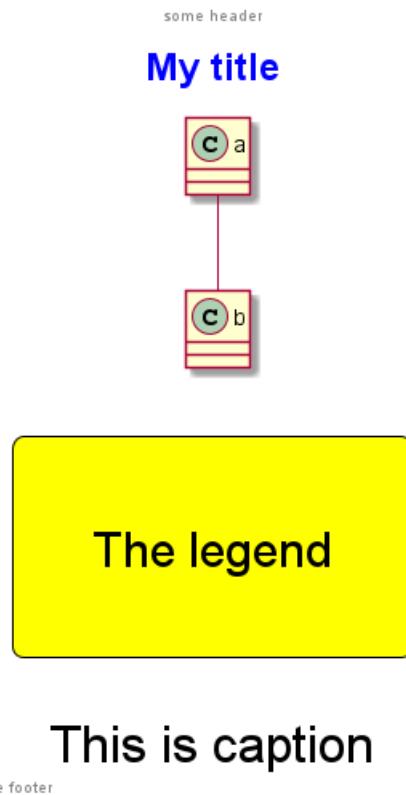
caption This is caption

legend
The legend
end legend

a -- b

@enduml
```





## This is caption

### 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

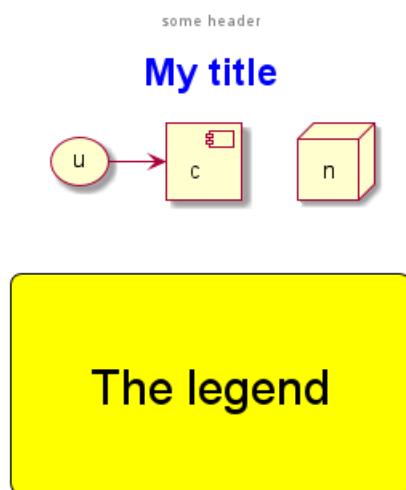
caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml

```



**This is caption**

some footer

#### 21.8.5 Gantt project planning

```

@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

```

[t] lasts 5 days

@enduml

some header

## My title

1	2	3	4	5
t				
1	2	3	4	5

The legend

This is caption

some footer

### 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

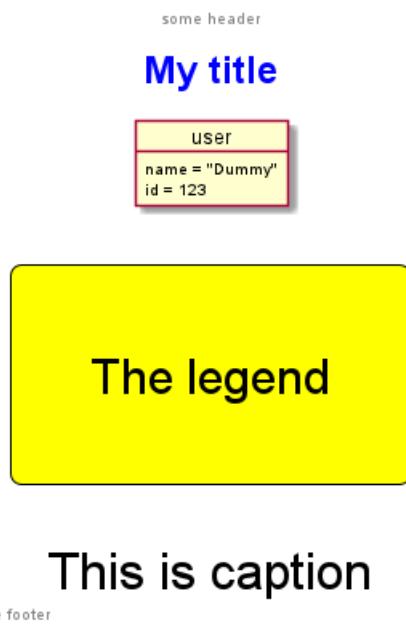
caption This is caption

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
```

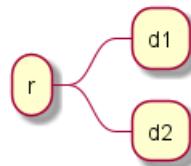
```
legend
The legend
end legend
```

```
* r
** d1
** d2
```

```
@endmindmap
```

some header

## My title



The legend

This is caption

some footer

### 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
```

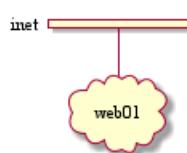
```
legend
The legend
end legend
```

```
nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
```

```
@enduml
```

some header

## My title



The legend

This is caption

some footer

### 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

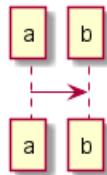
legend
The legend
end legend

a->b
@enduml
```



some header

## My title



The legend

This is caption  
some footer

### 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

```

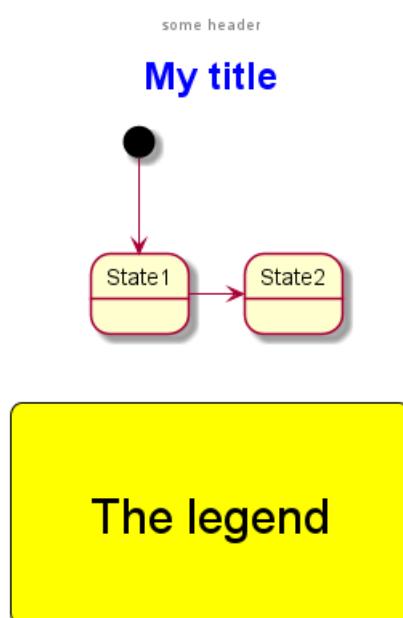
```

legend
The legend
end legend

[*] --> State1
State1 -> State2

```

```
@enduml
```



This is caption

some footer

### 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
```

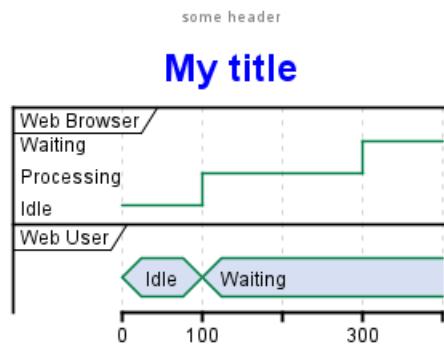
```
@0
WU is Idle
WB is Idle
```

```
@100
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
```

```
@enduml
```





This is caption

some footer

#### 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

```

```

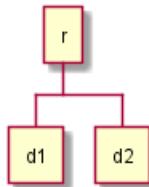
* r
** d1
** d2

```

```
@endwbs
```

some header

## My title



The legend

This is caption

some footer

### 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
```



```
The legend
```

```
This is caption
```

```
some footer
```



## 22 Creole

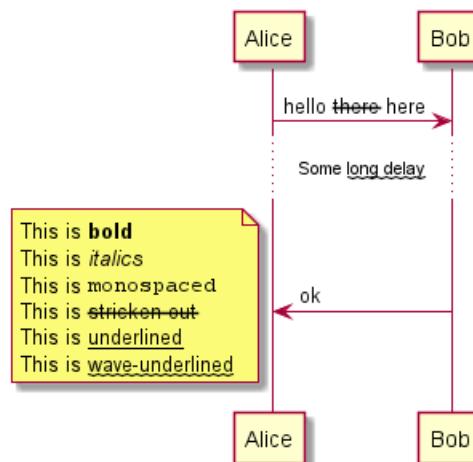
Creole is a lightweight common markup language for various wikis. A light-weight Creole engine is integrated in PlantUML to have a standardized way to emit styled text.

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

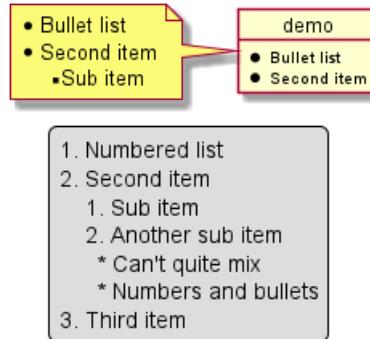
legend
    # Numbered list
    # Second item
```



```

## Sub item
## Another sub item
  * Can't quite mix
  * Numbers and bullets
# Third item
end legend
@enduml

```



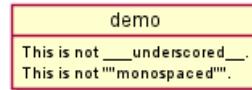
## 22.3 Escape character

You can use the tilde ~ to escape special creole characters.

```

@startuml
object demo {
  This is not ~__underscored__.
  This is not ~""monospaced"".
}
@enduml

```



## 22.4 Horizontal lines

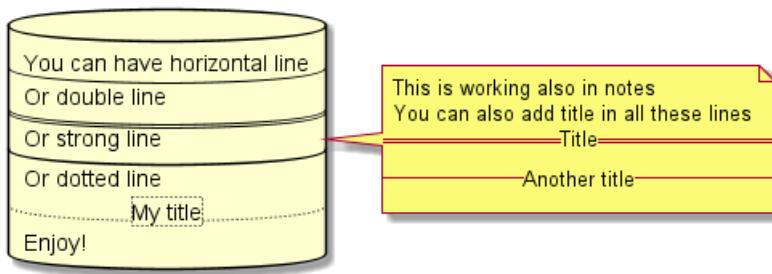
```

@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

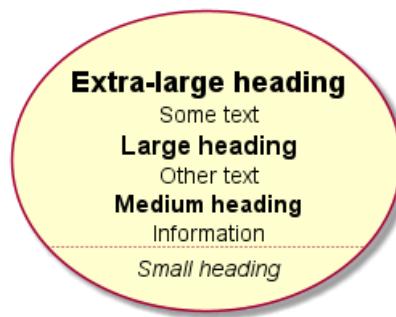
```





## 22.5 Headings

```
@startuml
usecase UC1 as "
= Extra-large heading
Some text
== Large heading
Other text
==== Medium heading
Information
....
===== Small heading"
@enduml
```



## 22.6 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
- <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>
```



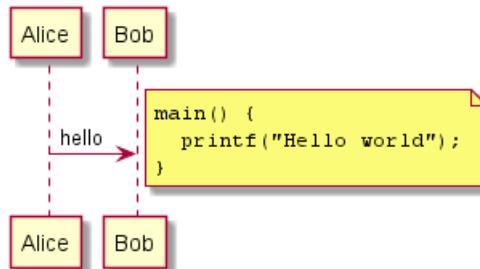
```
* You use <u:red>color</u> <s:green>in HTML</s> <w:#0000FF>tag</w>
-----
* Use image : <img:http://plantuml.com/logo3.png>
;
@enduml
```



## 22.7 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
```

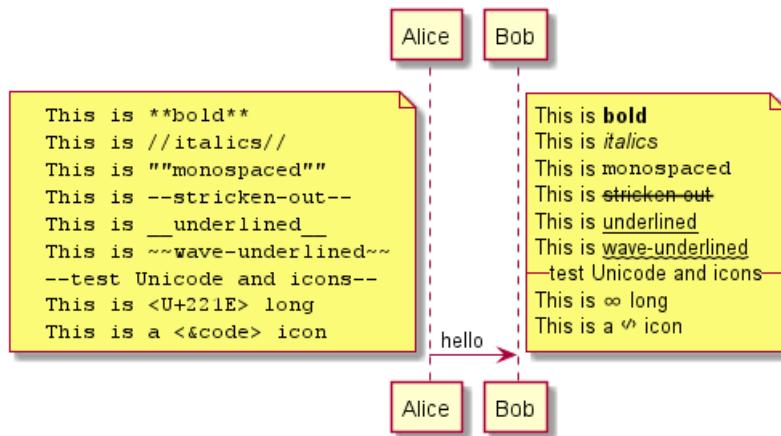


This is especially useful to illustrate some PlantUML code and the resulting rendering:

```
@startuml
Alice -> Bob : hello
note left
<code>
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
</code>
```



```
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
</code>
end note
note right
    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
```



## 22.8 Table

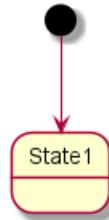
### 22.8.1 Create a table

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
```

Example of simple table

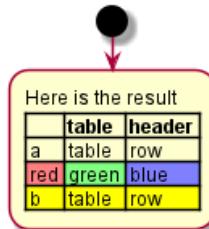
	table	header
a	table	row
b	table	row



### 22.8.2 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
```



### 22.8.3 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]

### 22.8.4 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
```



[Ref. QA-12448]

### 22.8.5 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]

## 22.9 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
```

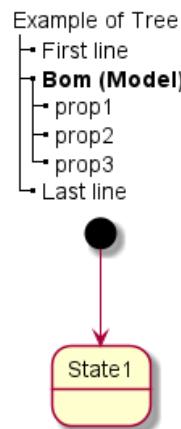
```
|_ prop2
```

```
|_ prop3
```

```
|_ Last line
```



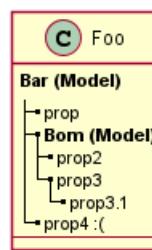
```
end title
[*] --> State1
@enduml
```



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:

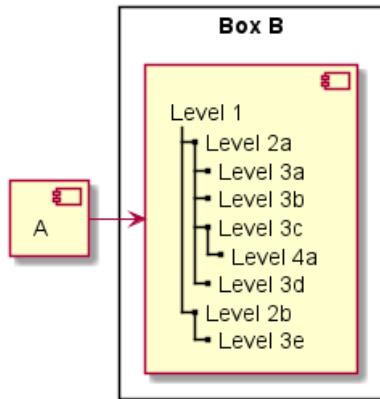
```
@startuml
[A] as A
rectangle "Box B" {
  component B [
    Level 1
    |_ Level 2a
    |_ Level 3a
    |_ Level 3b
    |_ Level 3c
    |_ Level 4a
    |_ Level 3d
    |_ Level 2b
```



```

|_ Level 3e
]
}
A -> B
@enduml

```



[Ref. QA-11365]

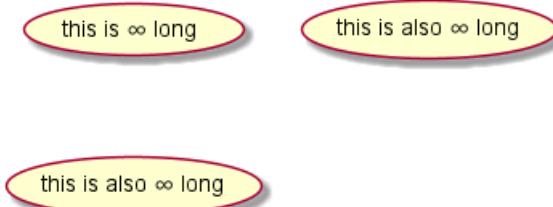
## 22.10 Special characters

It's possible to use any unicode character, either directly or with syntax &#XXX or <U+XXXX>:

```

@startuml
usecase direct as "this is ☺ long"
usecase ampHash as "this is also &#8734; long"
usecase angleBrackets as "this is also <U+221E> long"
@enduml

```



## 22.11 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.

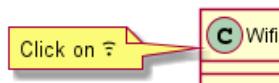
Use the following syntax: <&ICON\_NAME>.

```

@startuml
title: <size:20><&heart>Use of OpenIconic<&heart></size>
class Wifi
note left
    Click on <&wifi>
end note
@enduml

```

♥Use of OpenIconic♥



The complete list is available at the OpenIconic Website, or you can use the following special command to list them:

```
@startuml
listopeniconic
@enduml
```

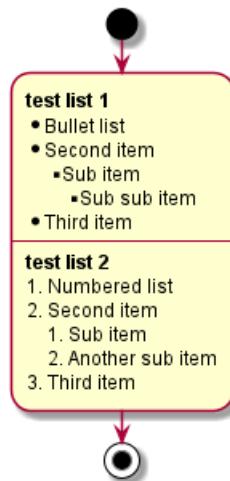
<b>List Open Iconic</b>	<b>bell</b>	<b>cloud</b>	<b>excerpt</b>	<b>justify-right</b>	<b>musical-note</b>	<b>star</b>
<i>Credit to</i>						
<a href="https://useiconic.com/open">https://useiconic.com/open</a>						
<b>B bold</b>						

## 22.12 Appendix: Examples of "Creole List" on all diagrams

### 22.12.1 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
```





### 22.12.2 Class

**TODO:** FIXME

- *Sub item*
- *Sub sub item*

**TODO:** FIXME

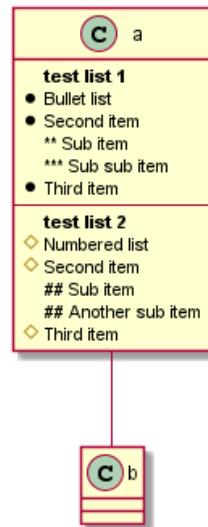
@startuml

```
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



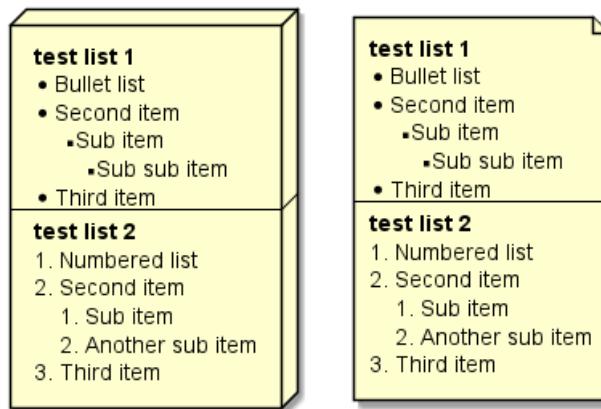


### 22.12.3 Component, Deployment, Use-Case

```
@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]

#### 22.12.4 Gantt project planning

N/A

#### 22.12.5 Object

**TODO:** FIXME

- *Sub item*
- *Sub sub item*

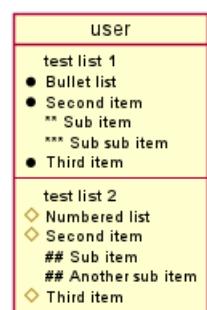
**TODO:** FIXME

```

@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

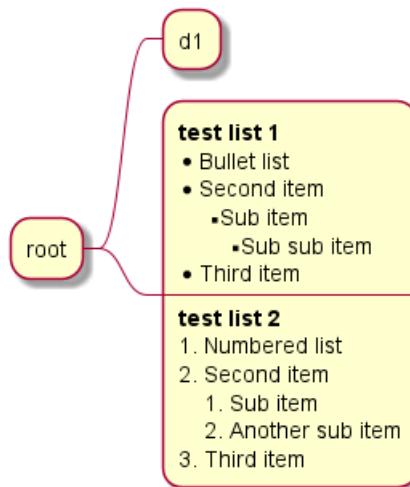


### 22.12.6 MindMap

```
@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
```



### 22.12.7 Network (nwdiag)

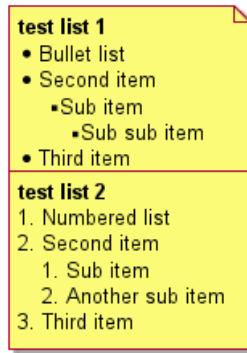
N/A

### 22.12.8 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
```



### 22.12.9 Sequence

N/A (*or on note or common commands*)

### 22.12.10 State

N/A (*or on note or common commands*)

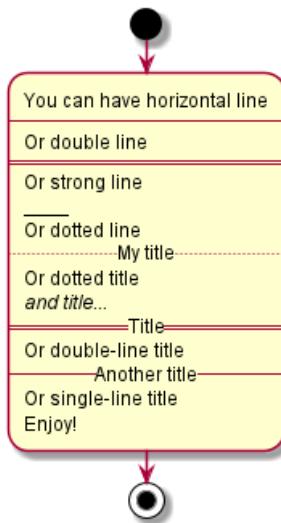
## 22.13 Appendix: Examples of "Creole horizontal lines" on all diagrams

### 22.13.1 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
```





### 22.13.2 Class

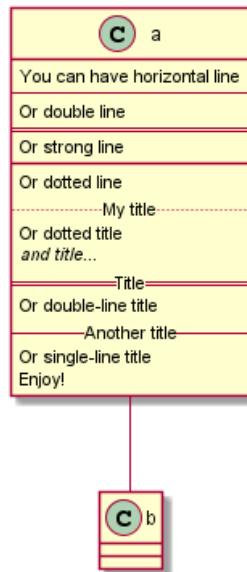
```
@startuml
```

```
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
```



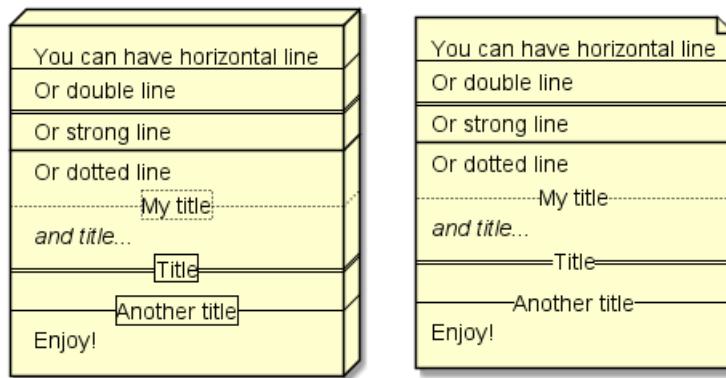


### 22.13.3 Component, Deployment, Use-Case

```
@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!
"
@enduml
```





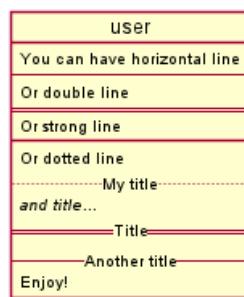
#### 22.13.4 Gantt project planning

N/A

#### 22.13.5 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]

#### 22.13.6 MindMap

**TODO:** FIXME strong line ---- **TODO:** FIXME

@startmindmap

```
* root
** d1
**:You can have horizontal line
----
```



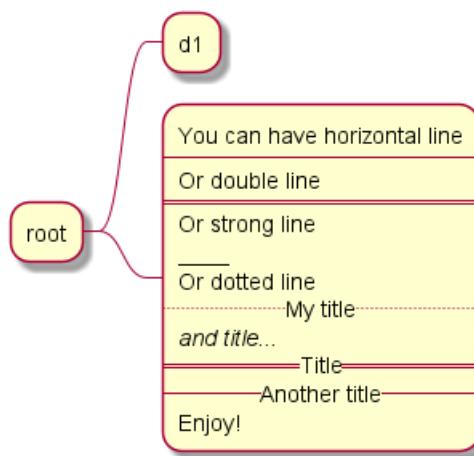
```
Or double line
=====
```

```
Or strong line
=====
```

```
Or dotted line
```

```
..My title...
//and title... //
==Title==
--Another title--
Enjoy!;
```

```
@endmindmap
```



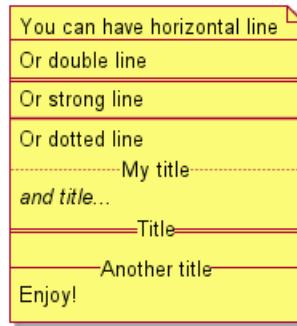
### 22.13.7 Network (nwdiag)

N/A

### 22.13.8 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
```





### 22.13.9 Sequence

N/A (or on note or common commands)

### 22.13.10 State

N/A (or on note or common commands)

## 22.14 Style equivalent (between Creole and HTML)

Style	Creole	Legacy HTML like
<b>bold</b>	This is **bold**	This is <b>bold</b>
<i>italics</i>	This is //italics//	This is <i>italics</i>
<u>monospaced</u>	This is ""monospaced""	This is <font:monospaced>monospaced</font>
stroked	This is --stroked--	This is <s>stroked</s>
<u>underlined</u>	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</font>
| \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</s> |
| \n This is ""~<u:color:red>""red""</color>"">underlined</u>""\n | \n This is <u:red>underlined</u> |
| \n This is ""~<w:#0000FF>""#0000FF""</color>"">waved</w>""\n | \n This is <w:#0000FF>waved</w>
@endmindmap
  
```



Style equivalent (between Creole and HTML)

Creole	
code	output
This is **bold**	This is <b>bold</b>
This is //italics//	This is <i>italics</i>
This is ""monospaced""	This is monospaced
This is --stroked--	This is <del>stroked</del>
This is __underlined__	This is <u>underlined</u>
This is ~~waved~~	This is <u>waved</u>

Legacy HTML like	
code	output
This is <b>bold</b>	This is <b>bold</b>
This is <i>italics</i>	This is <i>italics</i>
This is <font:monospaced>monospaced</font>	This is monospaced
This is <s>stroked</s>	This is <del>stroked</del>
This is <u>underlined</u>	This is <u>underlined</u>
This is <w>waved</w>	This is <u>waved</u>

And color as a bonus...

code		output
This is <s:green>stroked</s>	This is <del>stroked</del>	
This is <u:red>underlined</u>	This is <u>underlined</u>	
This is <w:#0000FF>waved</w>	This is <u>waved</u>	



## 23 Defining and using sprites

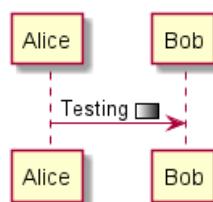
A *Sprite* is a small graphic element that can be used in diagrams.

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.

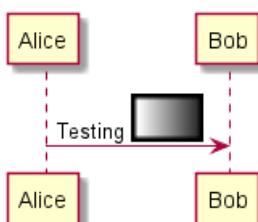
Then you can use the sprite using <\$XXX> where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    FFFFFFFFFFFFFF
}
Alice -> Bob : Te
@enduml
```



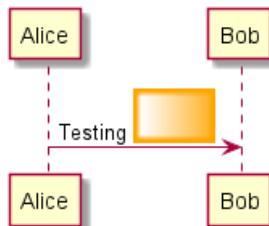
You can scale the sprite.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABCF
    FFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```



## 23.1 Changing colors

Although sprites are monochrome, it's possible to change their color.



## 23.2 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.

### 23.3 Importing Sprite

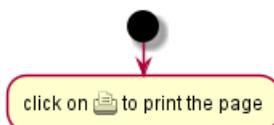
You can also launch the GUI to generate a sprite from an existing image.

Click in the menubar then on File/Open Sprite Window.

After copying an image into your clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

## 23.4 Examples

```
@startuml
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
start
:click on <$printer> to print the page;
@enduml
```



```

@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEqB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_12MHM-BLRyywPhrr
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj71HWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXw
sprite $disk {
    444445566677881
    4360000000009991
    436000000000ACA1
    53700000001A7A1
    53700000012B8A1
    53800000123B8A1
    63800001233C9A1
    634999AABC99B1
    744566778899AB1
    7456AAAAA99AAB1
    8566AFC228AABB1
    8567AC8118BBBB1
    867BD4433BBBB1
    39AAAAABBBBBBC1
}

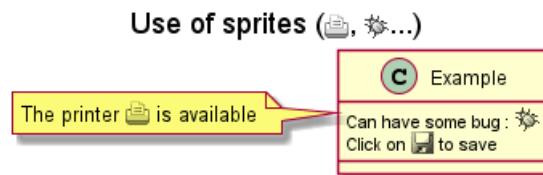
title Use of sprites (<$printer>, <$bug>...)

class Example {
    Can have some bug : <$bug>
    Click on <$disk> to save
}

note left : The printer <$printer> is available

@enduml

```



## 23.5 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

## 23.6 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/master/sprites.puml
!include osaPuml/Common.puml
!include osaPuml/User/all.puml
```

```
listsprites
@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
```



```

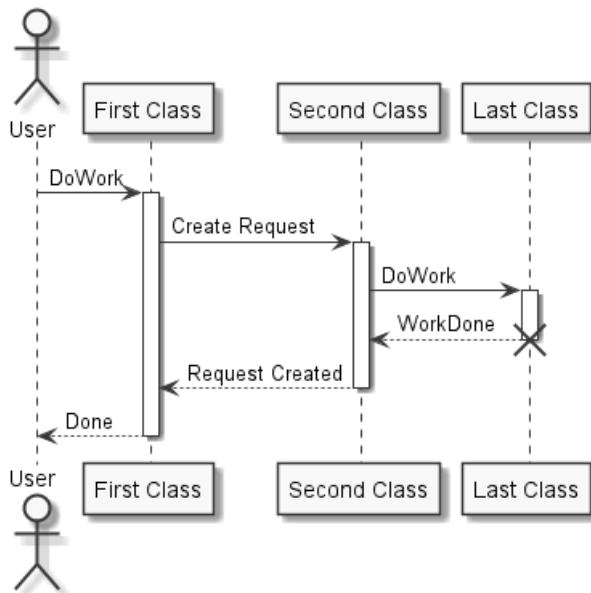
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

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

```

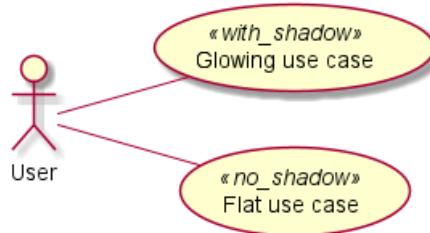
```

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

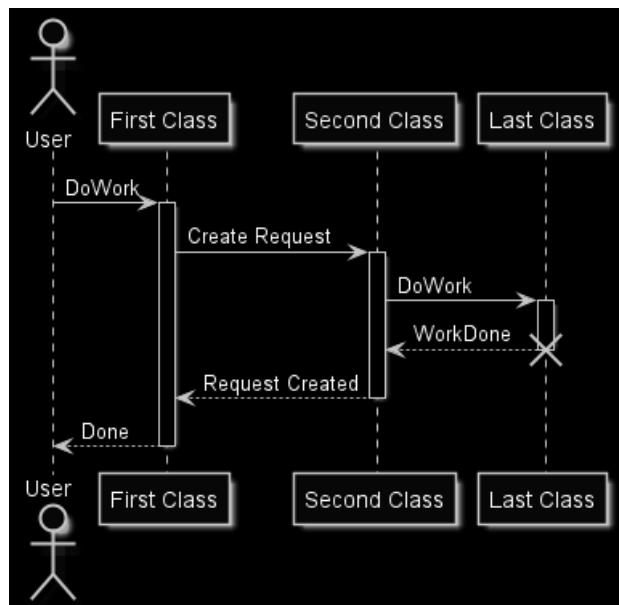
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

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	MediumStateBlue	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)

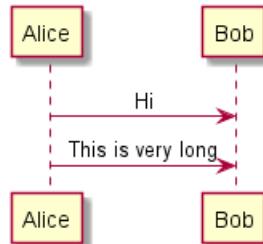
通过 `left`, `right` or `center`, 可以设置文本对齐.

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
<code>sequenceMessageAlign</code>	<code>left</code>	用于时序图中的消息 (message)
<code>sequenceReferenceAlign</code>	<code>center</code>	在时序图中用于 <code>ref over</code>

```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Alice -> Bob : This is very long
@enduml
```



## 24.9 Examples

```
@startuml
skinparam backgroundColor #EEEBCD
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

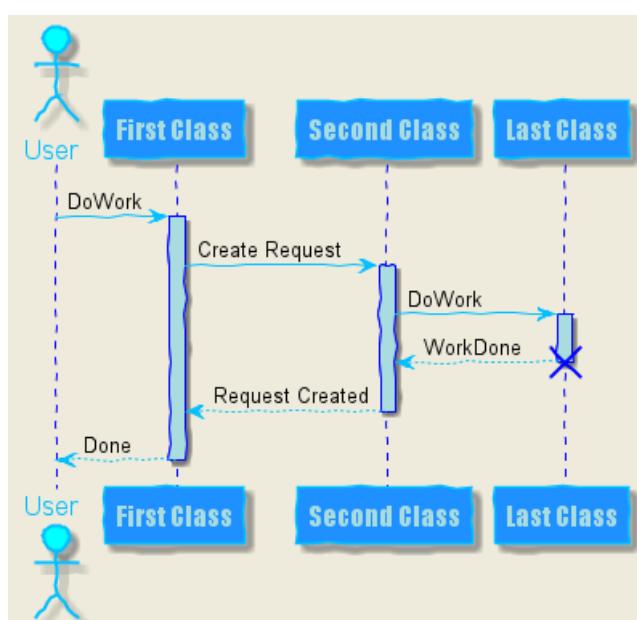
A -> B: Create Request
activate B

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

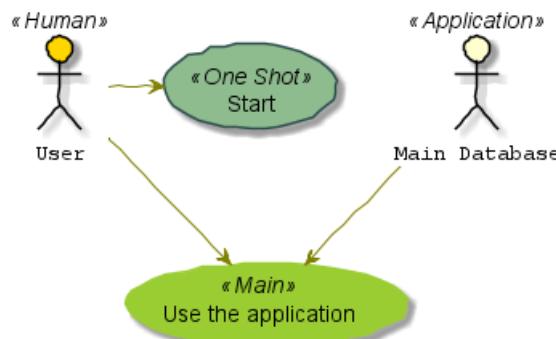
BackgroundColor<< Main >> YellowGreen
BorderColor<< Main >> YellowGreen

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
```

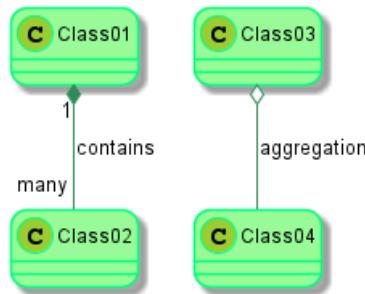


```
@startuml
skinparam roundcorner 20
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation
@enduml
```





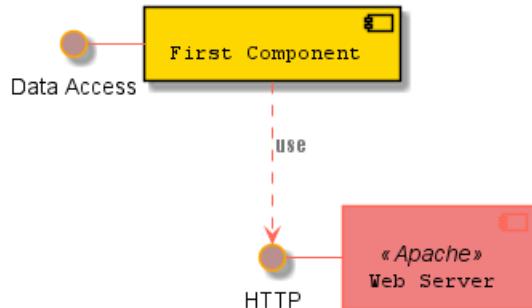
```

@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

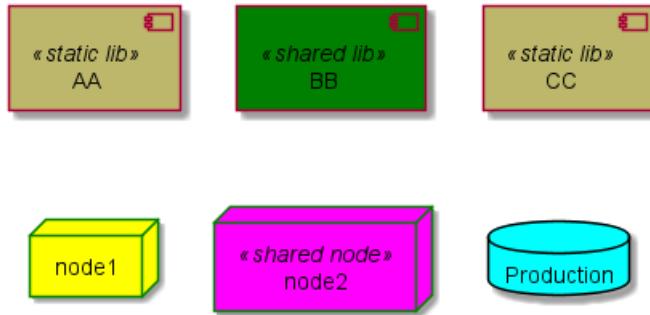
skinparam component {
  backgroundColor<<static lib>> DarkKhaki
}
  
```

```

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 Since the documentation is not always up to date, you can have the complete list of parameters using this command:

`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:

结果如下：

blockquote That will give you the following result:

```

@startuml
help skinparams
@enduml

```



### Help on skinparam

The code of this command is located in `net.sourceforge.plantuml.help` package.

You may improve it on <https://github.com/plantuml/plantuml/tree/master/src/net/sourceforge/plantuml/help>

The possible skinparam are :

- ActivityBackgroundColor
- ActivityBarColor
- ActivityBorderColor
- ActivityBorderThickness
- ActivityDiamondBackgroundColor
- ActivityDiamondBorderColor
- ActivityDiamondFontColor
- ActivityDiamondFontName
- ActivityDiamondFontSize
- ActivityDiamondFontStyle
- ActivityEndColor
- ActivityFontColor
- ActivityFontName
- ActivityFontSize
- ActivityFontStyle
- ActivityStartColor
- ActorBackgroundColor
- ActorBorderColor
- ActorFontColor
- ActorFontName
- ActorFontSize
- ActorFontStyle
- ActorStereotypeFontColor
- ActorStereotypeFontName
- ActorStereotypeFontSize
- ActorStereotypeFontStyle
- AgentBackgroundColor
- AgentBorderColor
- AgentBorderThickness
- AgentFontColor
- AgentFontName
- AgentFontSize
- AgentFontStyle
- AgentStereotypeFontColor
- AgentStereotypeFontName
- AgentStereotypeFontSize
- AgentStereotypeFontStyle
- ArchimateBackgroundColor
- ArchimateBorderColor
- ArchimateBorderThickness
- ArchimateFontColor
- ArchimateFontName
- ArchimateFontSize
- ArchimateFontStyle
- ArchimateStereotypeFontColor
- ArchimateStereotypeFontName
- ArchimateStereotypeFontSize
- ArchimateStereotypeFontStyle
- ArrowColor
- ArrowFontColor
- ArrowFontName
- ArrowFontSize
- ArrowFontStyle
- ArrowLollipopColor
- ArrowMessageAlignment
- ArrowThickness



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>](<https://plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html>).



## 25 预处理

PlantUML 包含了一些辅助性的预处理功能, 并且适用于所有的图.

这些功能与 C language preprocessor 很相似, 除了标记由 # 替换为!.

### 25.1 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 现在可以接受 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.

### 25.2 定义变量

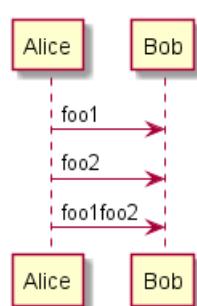
虽然这还是必须的, 我们强烈建议变量名以 \$ 开头. 有两类数据类型:

- 整型
- 字符串 - 必须被单引号或双引号包围.

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
!$ab = "foo1"
!$cd = "foo2"
!$ef = $ab + $cd

Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
@enduml
```



## 25.3 Boolean expression

### 25.3.1 Boolean representation [0 is false]

There is not real boolean type, but PlantUML use this integer convention:

- Integer 0 means **false**
- and any non-null number (as 1) or any string (as "1", or even "0") means **true**.

[Ref. QA-9702]

### 25.3.2 Boolean operation and operator [&&, ||, ()]

You can use boolean expression, in the test, with :

- *parenthesis ()*;
- *and operator &&*;
- *or operator ||*.

(See next example, within *if* test.)

### 25.3.3 Boolean builtin functions [%false(), %true(), %not(<exp>) ]

For convenience, you can use those boolean builtin functions:

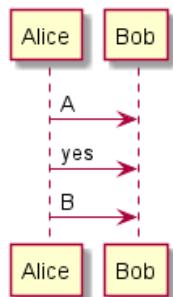
- `%false()`
- `%true()`
- `%not(<exp>)`

[See also *Builtin functions*]

## 25.4 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
```

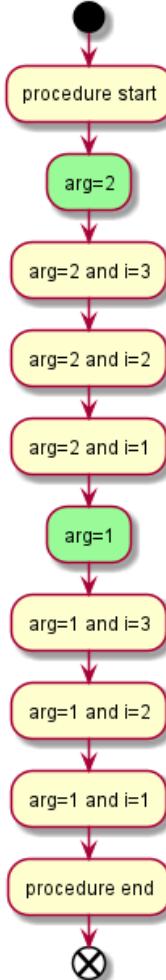


## 25.5 While loop [`!while`, `!endwhile`]

You can use `!while` and `!endwhile` keywords to have repeat loops.

```
@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
```



[Adapted from QA-10838]

```
@startmindmap
!procedure $foo($arg)
```

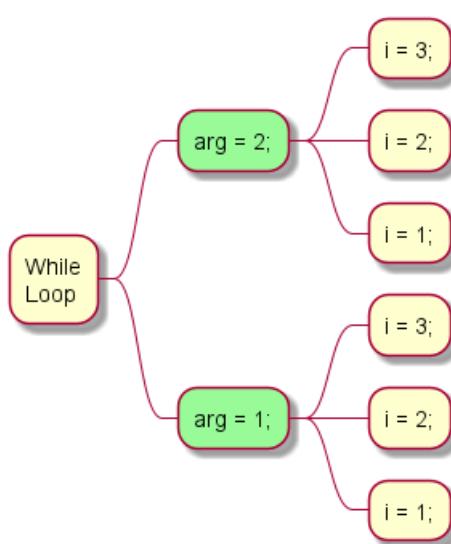


```

!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

```



## 25.6 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

```

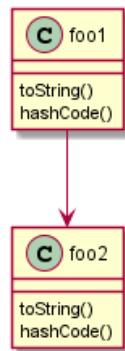
```

!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.

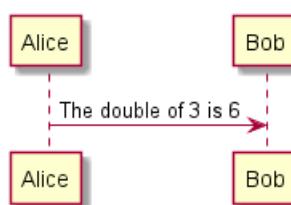
## 25.7 返回函数

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
```

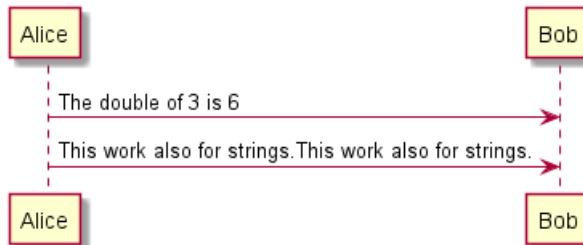


可以简化函数定义为一行:

```
@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.

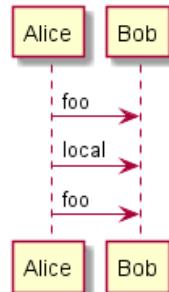
```

@startuml
!procedure $dummy()
!local $ijk = "local"
Alice -> Bob : $ijk
!endprocedure

!global $ijk = "foo"

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

```



## 25.8 参数默认值

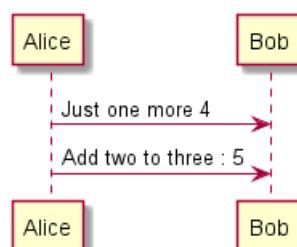
在返回和空函数中, 你可以定义参数默认值.

```

@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
```

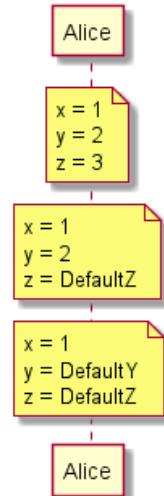


```

!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

```



## 25.9 非引号函数

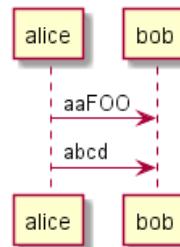
默认情况下, 调用一个函数需要引号. 可以使用 `unquoted` 关键字指明一个函数的参数不需要使用引号.

```

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

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

```



## 25.10 Keywords arguments

Like in Python, you can use keywords arguments :

```
@startuml
```

```

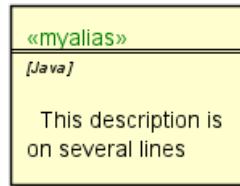
!unquoted procedure $element($alias, $description="", $label="", $technology="", $size=12, $colour=")
rectangle $alias as "
<color:$colour><$alias></color>
==$label==
```



```
//<size:$size>[$technology]</size>//

$description"
!endprocedure

$element(myalias, "This description is %newline()on several lines", $size=10, $technology="Java")
@enduml
```



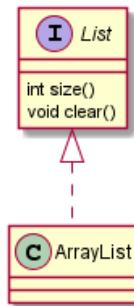
## 25.11 Including files or URL [`!include`, `!include_many`, `!include_once`]

Use the `!include` directive to include file in your diagram. Using URL, you can also include file from Internet/Intranet.

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.

## 25.12 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
```

## 25.13 Built-in functions [%]

Some functions are defined by default. Their name starts by %



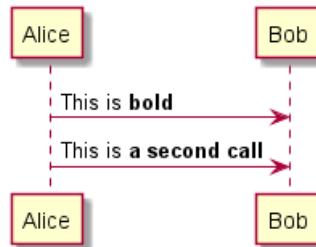
Name	Description	Example
%date	Retrieve current date. You can provide an optional format for the date	%date("yyyy.MM.dd")
%dirname	Retrieve current dirname	%dirname()
%false	Return always false	%false()
%file_exists	Check if a file exists on the local filesystem	%file_exists("c:/")
%filename	Retrieve current filename	%filename()
%function_exists	Check if a function exists	%function_exists()
%get_variable_value	Retrieve some variable value	%get_variable_value()
%getenv	Retrieve environment variable value	%getenv("OS")
%intval	Convert a String to Int	%intval("42")
%lower	Return a lowercase string	%lower("Hello")
%newline	Return a newline	%newline()
%not	Return the logical negation of an expression	%not(2+2==4)
%set_variable_value	Set a global variable	%set_variable_value()
%string	Convert an expression to String	%string(1 + 2)
%strlen	Calculate the length of a String	%strlen("foo")
%strpos	Search a substring in a string	%strpos("abcdef", "def")
%substr	Extract a substring. Takes 2 or 3 arguments	%substr("abcdef", 3, 3)
%true	Return always true	%true()
%upper	Return an uppercase string	%upper("Hello")
%variable_exists	Check if a variable exists	%variable_exists()
%version	Return PlantUML current version	%version()

## 25.14 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
```



## 25.15 Memory dump [!memory\_dump]

You can use `!memory_dump` to dump the full content of the memory when generating the diagram. An optional string can be put after `!memory_dump`. 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
!endfunction
```

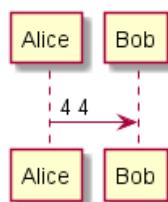


```

!dump_memory
!return $val+1
!endfunction

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

```



## 25.16 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>

(If you use this software, you accept its license)  
(Details by typing `license` keyword)

```

PlantUML 1.2021.3beta6
[From string (line 3) ]

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

```

## 25.17 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
```

...

## 25.18 Search path

You can specify the java property `plantuml.include.path` in the command line.

For example:

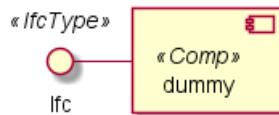
```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

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.

## 25.19 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
```



## 25.20 Dynamic invocation [%invoke\_procedure(), %call\_user\_func()]

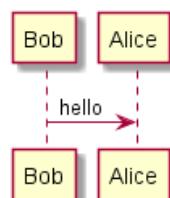
You can dynamically invoke a procedure using the special `%invoke_procedure()` procedure. This procedure takes as first argument the name of the actual procedure to be called. The optional following arguments are copied to the called procedure.

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
```

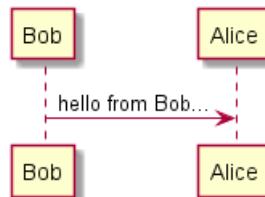


```

!endprocedure

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

```



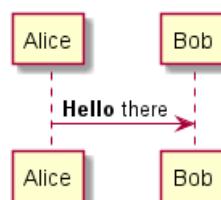
For return functions, you can use the corresponding special function `%call_user_func()` :

```

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

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

```



## 25.21 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

```



	<b>\$a</b>	<b>\$b</b>	<b>%string(\$a + \$b)</b>
<b>type</b>	str	str	str (concatenation)
<b>example</b>	"a"	"b"	ab
<b>type</b>	str	int	str (concatenation)
<b>ex.</b>	"a"	2	a2
<b>type</b>	str	int	str (concatenation)
<b>ex.</b>	1	"b"	1b
<b>type</b>	bool	str	str (concatenation)
<b>ex.</b>	%true()	"b"	1b
<b>type</b>	str	bool	str (concatenation)
<b>ex.</b>	"a"	%false()	a0
<b>type</b>	int	int	int (addition of int)
<b>ex.</b>	1	2	3
<b>type</b>	bool	int	int (addition)
<b>ex.</b>	%true()	2	3
<b>type</b>	int	bool	int (addition)
<b>ex.</b>	1	%false()	1
<b>type</b>	int	int	int (addition)
<b>ex.</b>	1	%intval("2")	3

## 25.22 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*)



## 26 Unicode

The PlantUML language use *letters* to define actor, usecase and soon.

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

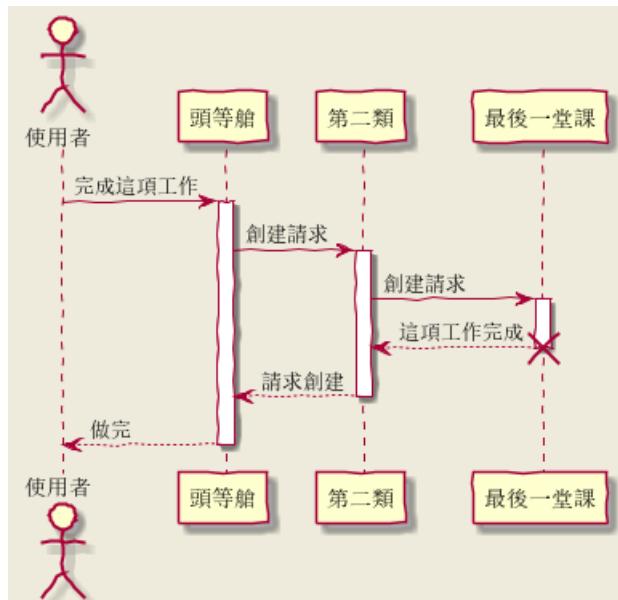
使用者 -> A: 完成這項工作
activate A

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
```

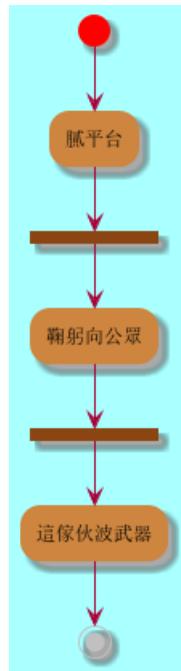


```
@startuml
(*) --> "膩平台"
--> === S1 ===
--> 鞠躬向公眾
```



```
--> === S2 ===
--> 這傢伙波武器
--> (*)
```

```
skinparam backgroundColor #AFFFFF
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
```

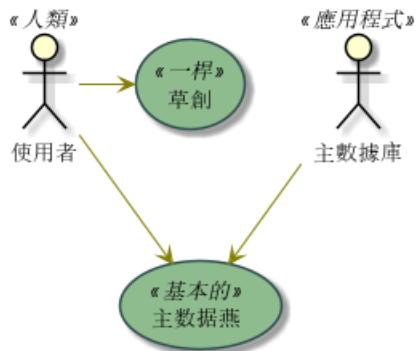
使用者 << 人類 >>  
 "主數據庫" as 數據庫 << 應用程式 >>  
 (草創) << 一桿 >>  
 "主数据燕" as (贏余) << 基本的 >>

使用者 -> (草創)  
 使用者 --> (贏余)

數據庫 --> (贏余)

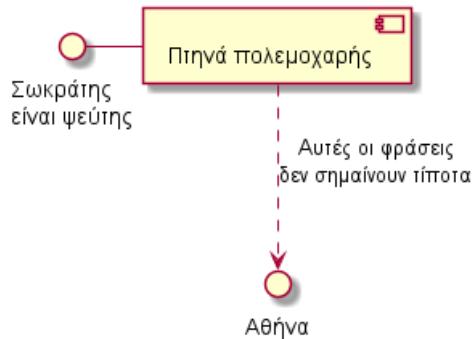
@enduml





```

@startuml
() "Σ" as Σ
Σ - [Π]
[Π] ..> () A : A
@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 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.



## 27 Standard Library

This page explains the official Standard Library (`stdlib`) for PlantUML. This Standard Library is now included in official releases of PlantUML. Including files follows the C convention for "C standard library".

Contents of the library come from third party contributors. We thank them for their useful contribution!

### 27.1 List of Standard Library

You can list standard library folders using the special diagram:

```
@startuml
stdlib
@enduml
```

```
archimate
Version 0.0.1
Delivered by https://github.com/ebbypeter/Archimate-PlantUML

aws
Version 18.02.22
Delivered by https://github.com/milo-minderbinder/AWS-PlantUML

awslib
Version 7.0.0
Delivered by https://github.com/awslabs/aws-icons-for-plantuml

azure
Version 2.1.0
Delivered by https://github.com/RicardoNiepel/Azure-PlantUML

c4
Version 2.0.0
Delivered by https://github.com/RicardoNiepel/C4-PlantUML

cloudinsight
Version 1.0.0
Delivered by https://github.com/rabelenda/cicon-plantuml-sprites/

cloudogu
Version 1.0.2
Delivered by https://github.com/cloudogu/plantuml-cloudogu-sprites

elastic
Version 0.0.1
Delivered by https://github.com/Crashedmind/PlantUML-Elastic-icons

kubernetes
Version 5.3.45
Delivered by https://github.com/michiel/plantuml-kubernetes-sprites

logos
Version 1.0.0
Delivered by https://github.com/rabelenda/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 2.2.0
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]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/archimate>
- <https://github.com/ebbypeter/Archimate-PlantUML>

This repository contains ArchiMate PlantUML macros and other includes for creating Archimate Diagrams easily and consistently.

```
@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)")

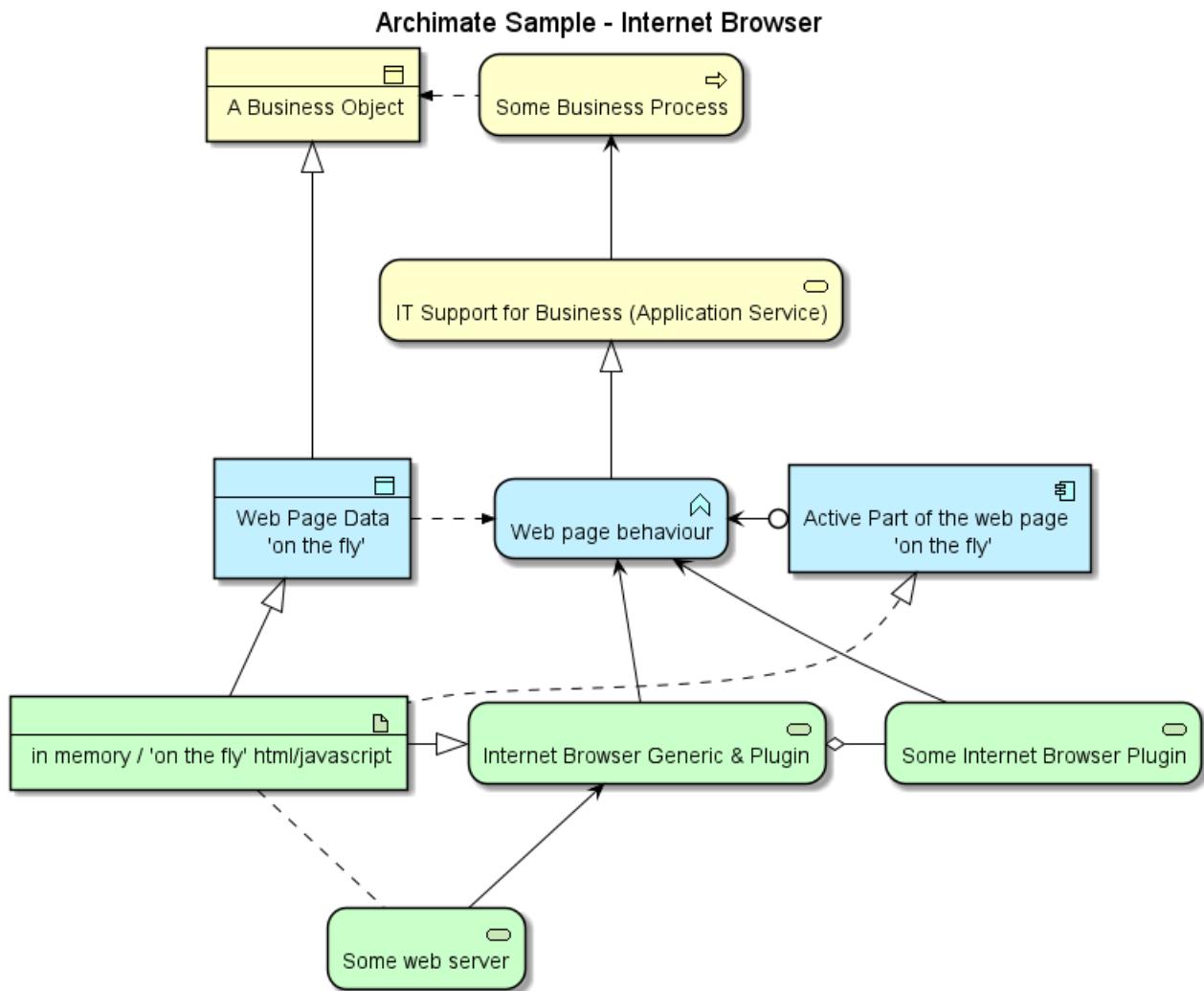
Application_DataObject(dataObject, "Web Page Data \n 'on the fly'")
Application_Function(webpageBehaviour, "Web page behaviour")
Application_Component(ActivePartWebPage, "Active Part of the web page \n 'on the fly')

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
```





### 27.3 AWS library [aws]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/aws>
- <https://github.com/milo-minderbinder/AWS-PlantUML>

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.

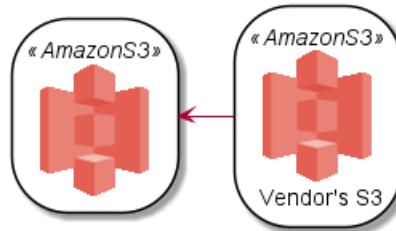
Example of usage:

```

@startuml
!include <aws/common>
!include <aws/Storage/AmazonS3/AmazonS3>

AMAZONS3(s3_internal)
AMAZONS3(s3_partner, "Vendor's S3")
s3_internal <- s3_partner
@enduml
  
```

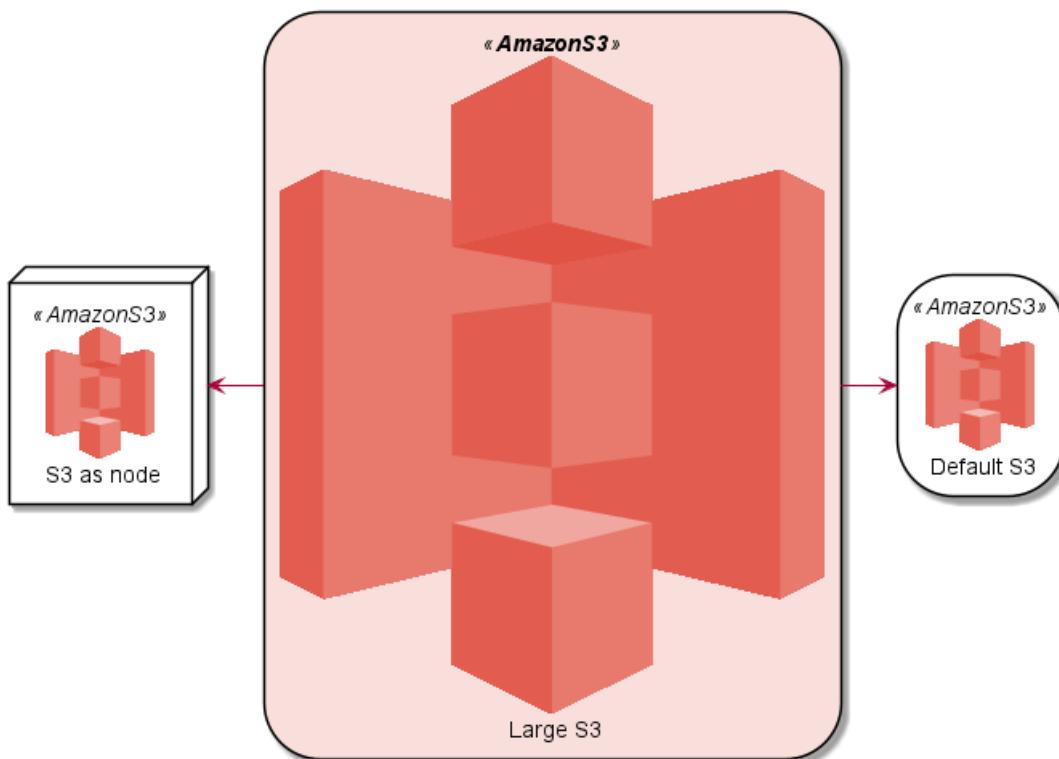




```
@startuml
!include <aws/common>
!include <aws/Storage/AmazonS3/AmazonS3>
!include <aws/Storage/AmazonS3/AmazonS3_LARGE>

skinparam nodeBackgroundColor White
skinparam storage<<**AmazonS3**>> {
    backgroundColor #F9DFDC
}
AMAZONS3(s3_internal,"Default S3")
AMAZONS3(s3_internal2,"S3 as node",node)
AMAZONS3_LARGE(s3_partner,"Large S3")

s3_internal2 <-r- s3_partner
s3_internal <-l- s3_partner
@enduml
```



## 27.4 Amazon Labs AWS Library [awslib]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/awslib>
- <https://github.com/awslabs/aws-icons-for-plantuml>

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
'Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
'SPDX-License-Identifier: MIT (For details, see https://github.com/awslabs/aws-icons-for-plantuml/bl

!include <awslib/AWSCommon>

' Uncomment the following line to create simplified view
' !include <awslib/AWSSimplified>

!include <awslib/General/Users>
!include <awslib/Mobile/APIGateway>
!include <awslib/SecurityIdentityAndCompliance/Cognito>
!include <awslib/Compute/Lambda>
!include <awslib/Database/DynamoDB>

left to right direction

Users(sources, "Events", "millions of users")
APIGateway(votingAPI, "Voting API", "user votes")
Cognito(userAuth, "User Authentication", "jwt to submit votes")
Lambda(generateToken, "User Credentials", "return jwt")
Lambda(recordVote, "Record Vote", "enter or update vote per user")
DynamoDB(voteDb, "Vote Database", "one entry per user")

sources --> userAuth
sources --> votingAPI
userAuth <--> generateToken
votingAPI --> recordVote
recordVote --> voteDb
@enduml
```

## 27.5 Azure library [azure]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/azure>
- <https://github.com/RicardoNiepel/Azure-PlantUML/>

The Azure library consists of Microsoft Azure icons.

Use it by including the file that contains the sprite, eg: !include <azure/Analytics/AzureEventHub.puml>. 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.puml>, 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.puml>
!include <azure/Analytics/AzureEventHub.puml>
!include <azure/Analytics/AzureStreamAnalytics.puml>
!include <azure/Database/AzureCosmosDb.puml>
```

left to right direction

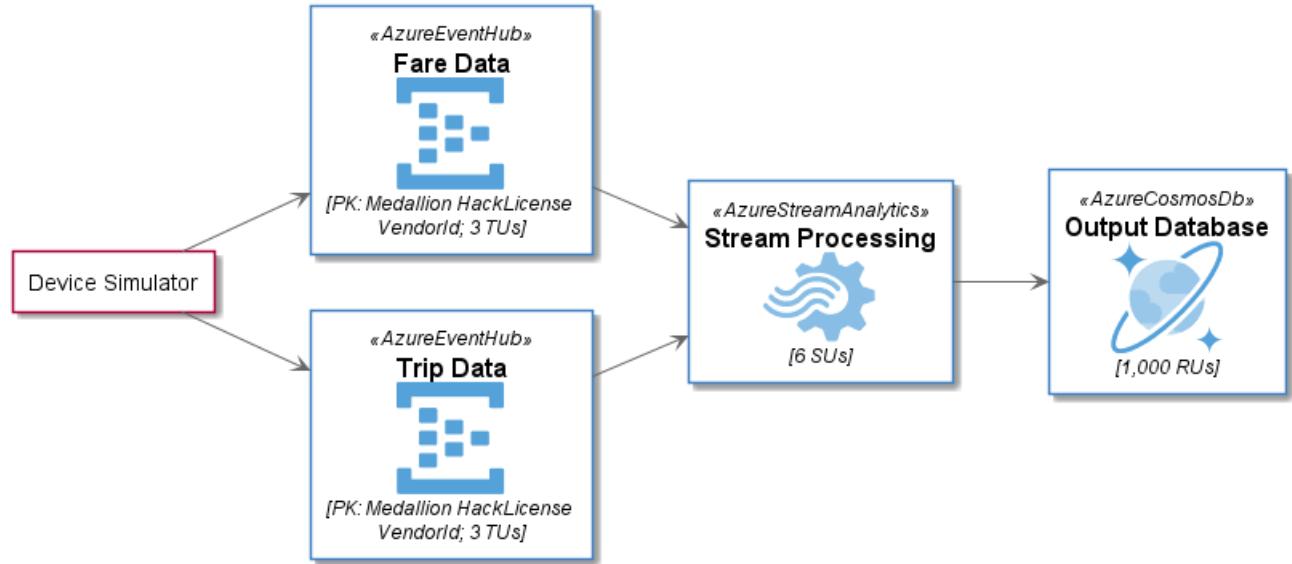
```
agent "Device Simulator" as devices #fff
```

```
AzureEventHub(fareDataEventHub, "Fare Data", "PK: Medallion HackLicense VendorId; 3 TUs")
```



```
AzureEventHub(tripDataEventHub, "Trip Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureStreamAnalytics(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")
```

```
devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```



## 27.6 C4 Library [C4]

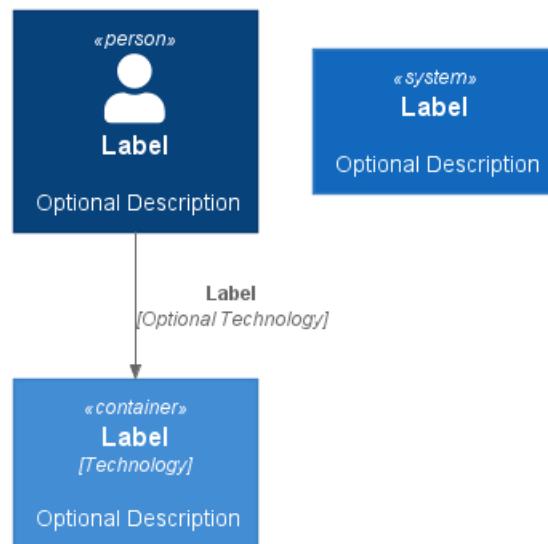
- <https://github.com/plantuml/plantuml-stdlib/tree/master/C4>
- <https://github.com/plantuml-stdlib/C4-PlantUML>

```
@startuml
!include <C4/C4_Container>
```

```
Person(personAlias, "Label", "Optional Description")
Container(containerAlias, "Label", "Technology", "Optional Description")
System(systemAlias, "Label", "Optional Description")
```

```
Rel(personAlias, containerAlias, "Label", "Optional Technology")
@enduml
```





## 27.7 Cloud Insight [cloudinsight]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/cloudinsight>
- <https://github.com/rabelenda/cicon-plantuml-sprites>

This repository contains PlantUML sprites generated from Cloudinsight icons, which can easily be used in PlantUML diagrams for nice visual representation of popular technologies.

```

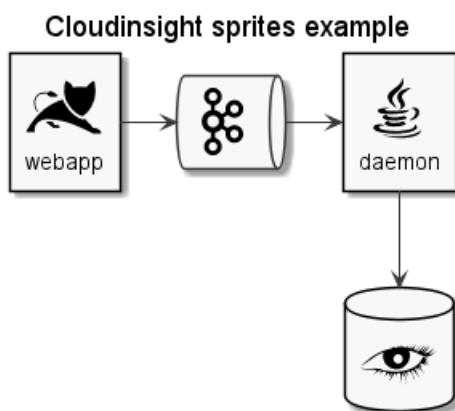
@startuml
!include <cloudinsight/tomcat>
!include <cloudinsight/kafka>
!include <cloudinsight/java>
!include <cloudinsight/cassandra>

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
  
```



## 27.8 Cloudogu [cloudogu]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/cloudogu>
- <https://github.com/cloudogu/plantuml-cloudogu-sprites>
- <https://cloudogu.com>

The Cloudogu library provides PlantUML sprites, macros, and other includes for Cloudogu services and resources.

```
@startuml
!include <cloudogu/common.puml>
!include <cloudogu/dogus/jenkins.puml>
!include <cloudogu/dogus/cloudogu.puml>
!include <cloudogu/dogus/scm.puml>
!include <cloudogu/dogus/smeagol.puml>
!include <cloudogu/dogus/nexus.puml>
!include <cloudogu/tools/k8s.puml>

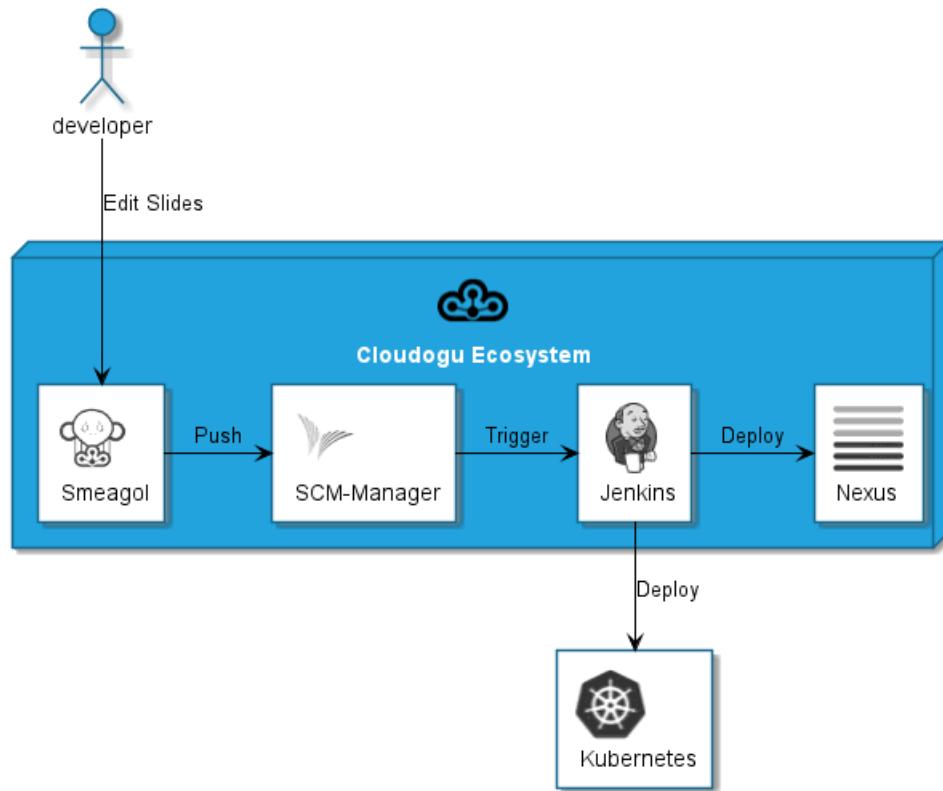
node "Cloudogu Ecosystem" <<$cloudogu>> {
    DOGU_JENKINS(jenkins, Jenkins) #ffffff
    DOGU_SCM(scm, SCM-Manager) #ffffff
    DOGU_SMEAGOL(smeagol, Smeagol) #ffffff
    DOGU_NEXUS(nexus, Nexus) #ffffff
}

TOOL_K8S(k8s, Kubernetes) #ffffff

actor developer

developer --> smeagol : "Edit Slides"
smeagol -> scm : Push
scm -> jenkins : Trigger
jenkins -> nexus : Deploy
jenkins --> k8s : Deploy
@enduml
```





### All cloudogu sprites

See all possible cloudogu sprites on [plantuml-cloudogu-sprites](#).

## 27.9 Elastic library [elastic]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/elastic>
- <https://github.com/Crashedmind/PlantUML-Elastic-icons>

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/elasticsearch/elasticsearch.puml. 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:

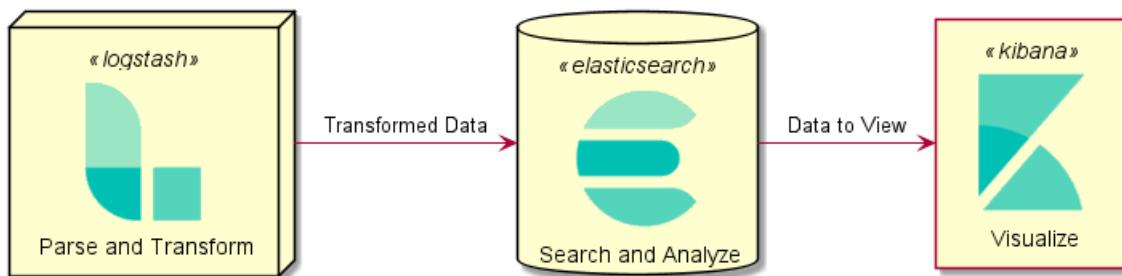
```

@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.puml>
!include <elastic/apm/apm.puml>
!include <elastic/app_search/app_search.puml>
!include <elastic/beats/beats.puml>
!include <elastic/cloud/cloud.puml>
!include <elastic/cloud_in_kubernetes/cloud_in_kubernetes.puml>
!include <elastic/code_search/code_search.puml>
!include <elastic/ece/ece.puml>
!include <elastic/eck/eck.puml>
' Beware of the difference between Crashedmind and plantuml-stdlib version: with '_' usage!
!include <elastic/elasticsearch/elasticsearch.puml>
!include <elastic/endpoint/endpoint.puml>
!include <elastic/enterprise_search/enterprise_search.puml>
!include <elastic/kibana/kibana.puml>
!include <elastic/logging/logging.puml>
!include <elastic/logstash/logstash.puml>
!include <elastic/maps/maps.puml>
!include <elastic/metrics/metrics.puml>
!include <elastic/siem/siem.puml>
!include <elastic/site_search/site_search.puml>
!include <elastic/stack/stack.puml>
!include <elastic/uptime/uptime.puml>

```

skinparam agentBackgroundColor White

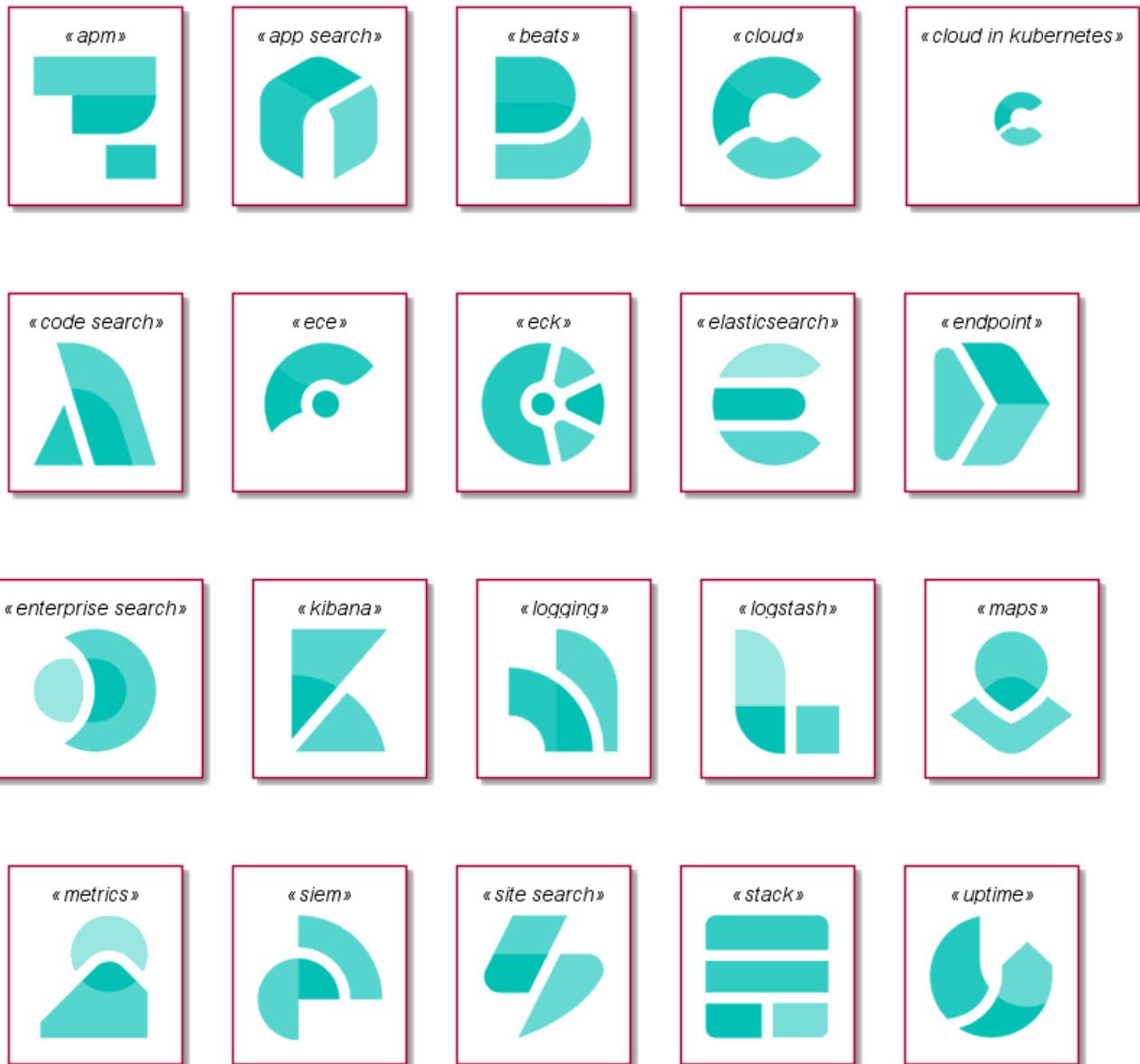
```

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
```



## 27.10 Google Material Icons [material]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/material>
- <https://github.com/Templarian/MaterialDesign>

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
```

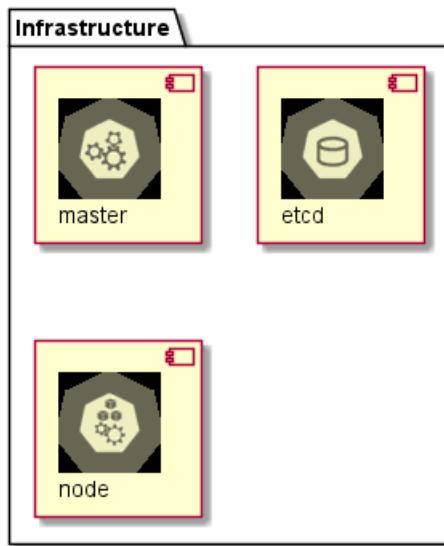


## 27.11 Kubernetes [kubernetes]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/kubernetes>
- <https://github.com/michiel/plantuml-kubernetes-sprites>

```
@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
```





## 27.12 Logos [logos]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/logos>
- <https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites>

This repository contains PlantUML sprites generated from Gil Barbara's logos, which can easily be used in PlantUML diagrams for nice visual aid.

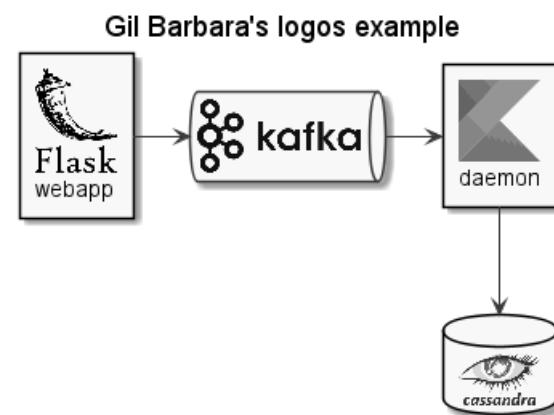
```
@startuml
!include <logos/flask.puml>
!include <logos/kafka.puml>
!include <logos/kotlin.puml>
!include <logos/cassandra.puml>
```

```
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
```



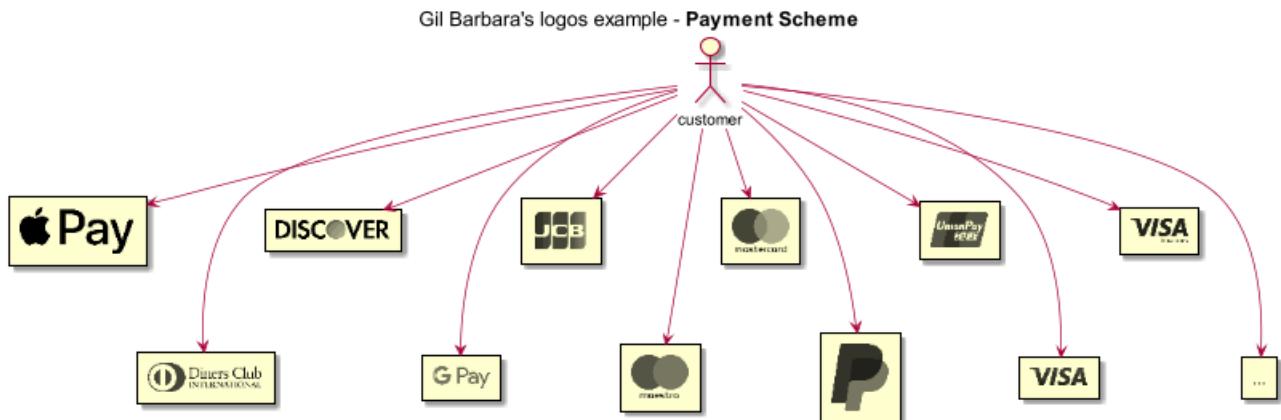
```
@startuml
scale 0.7
!include <logos/apple-pay.puml>
!include <logos/dinersclub.puml>
!include <logos/discover.puml>
!include <logos/google-pay.puml>
!include <logos/jcb.puml>
!include <logos/maestro.puml>
!include <logos/mastercard.puml>
!include <logos/paypal.puml>
!include <logos/unionpay.puml>
!include <logos/visaelectron.puml>
!include <logos/visa.puml>
' ...

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
```





### 27.13 Office [office]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/office>
- <https://github.com/Roemer/plantuml-office>

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:

```

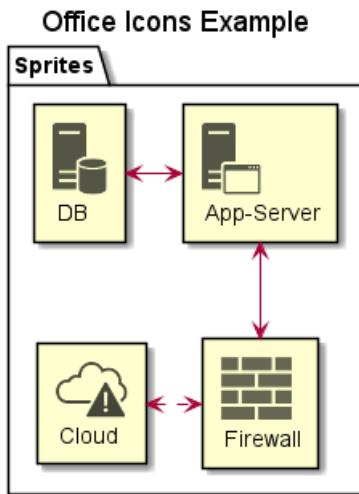
@startuml
!include <tupadr3/common>

!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

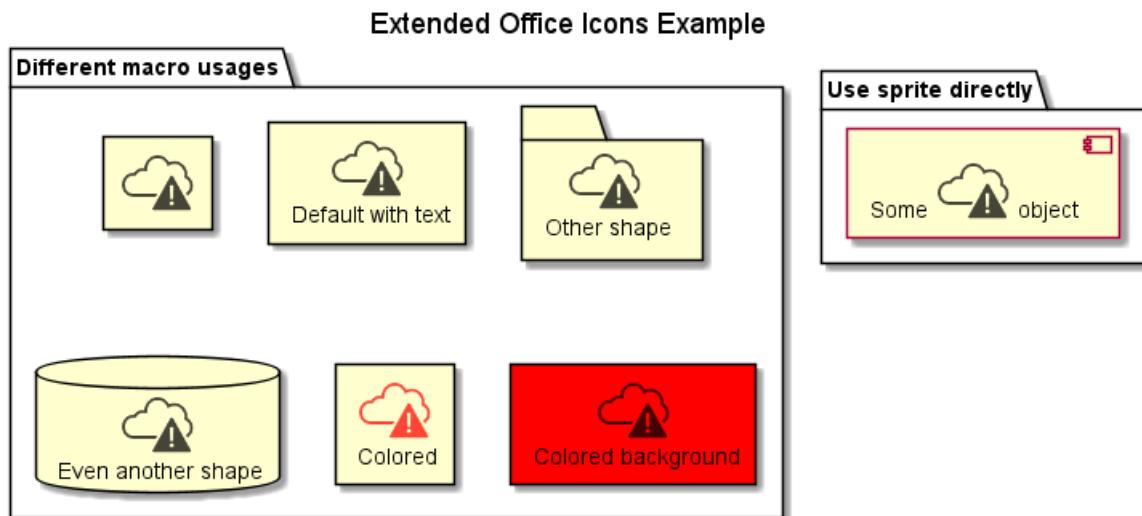
title Extended Office Icons Example

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

```





## 27.14 Open Security Architecture (OSA) [osa]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/osa>
- <https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons>
- <https://www.opensecurityarchitecture.org>

```
@startuml
'Adapted from https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons/blob/master/all.puml
scale .5
!include <osa/arrow/green/left/left.puml>
!include <osa/arrow/yellow/right/right.puml>
!include <osa/awareness/awareness.puml>
!include <osa/contract/contract.puml>
!include <osa/database/database.puml>
!include <osa/desktop/desktop.puml>
!include <osa/desktop/imac/imac.puml>
!include <osa/device_music/device_music.puml>
!include <osa/device_scanner/device_scanner.puml>
!include <osa/device_usb/device_usb.puml>
!include <osa/device_wireless_router/device_wireless_router.puml>
!include <osa/disposal/disposal.puml>
!include <osa/drive_optical/drive_optical.puml>
!include <osa/firewall/firewall.puml>
!include <osa/hub/hub.puml>
!include <osa/ics/drive/drive.puml>
!include <osa/ics/plc/plc.puml>
!include <osa/ics/thermometer/thermometer.puml>
!include <osa/id/card/card.puml>
!include <osa/laptop/laptop.puml>
!include <osa/lifecycle/lifecycle.puml>
!include <osa/lightning/lightning.puml>
!include <osa/media_flash/media_flash.puml>
!include <osa/media_optical/media_optical.puml>
!include <osa/media_tape/media_tape.puml>
!include <osa/mobile/pda/pda.puml>
!include <osa/padlock/padlock.puml>
!include <osa/printer/printer.puml>
!include <osa/site_branch/site_branch.puml>
!include <osa/site_factory/site_factory.puml>
!include <osa/vpn/vpn.puml>
```



```
!include <osa/wireless/network/network.puml>

rectangle "OSA" {
rectangle "Left: <$left>" 
rectangle "Right: <$right>" 
rectangle "Awareness: <$awareness>" 
rectangle "Contract: <$contract>" 
rectangle "Database: <$database>" 
rectangle "Desktop: <$desktop>" 
rectangle "Imac: <$imac>" 
rectangle "Device_music: <$device_music>" 
rectangle "Device_scanner: <$device_scanner>" 
rectangle "Device_usb: <$device_usb>" 
rectangle "Device_wireless_router: <$device_wireless_router>" 
rectangle "Disposal: <$disposal>" 
rectangle "Drive_optical: <$drive_optical>" 
rectangle "Firewall: <$firewall>" 
rectangle "Hub: <$hub>" 
rectangle "Drive: <$drive>" 
rectangle "Plc: <$plc>" 
rectangle "Thermometer: <$thermometer>" 
rectangle "Card: <$card>" 
rectangle "Laptop: <$laptop>" 
rectangle "Lifecycle: <$lifecycle>" 
rectangle "Lightning: <$lightning>" 
rectangle "Media_flash: <$media_flash>" 
rectangle "Media_optical: <$media_optical>" 
rectangle "Media_tape: <$media_tape>" 
rectangle "Pda: <$pda>" 
rectangle "Padlock: <$padlock>" 
rectangle "Printer: <$printer>" 
rectangle "Site_branch: <$site_branch>" 
rectangle "Site_factory: <$site_factory>" 
rectangle "Vpn: <$vpn>" 
rectangle "Network: <$network>" 
}
@enduml
```





## 27.15 Tupadr3 library [tupadr3]

- <https://github.com/plantuml/plantuml-stdlib/tree/master/tupadr3>
- <https://github.com/tupadr3/plantuml-icon-font-sprites>

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/common>`. 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`

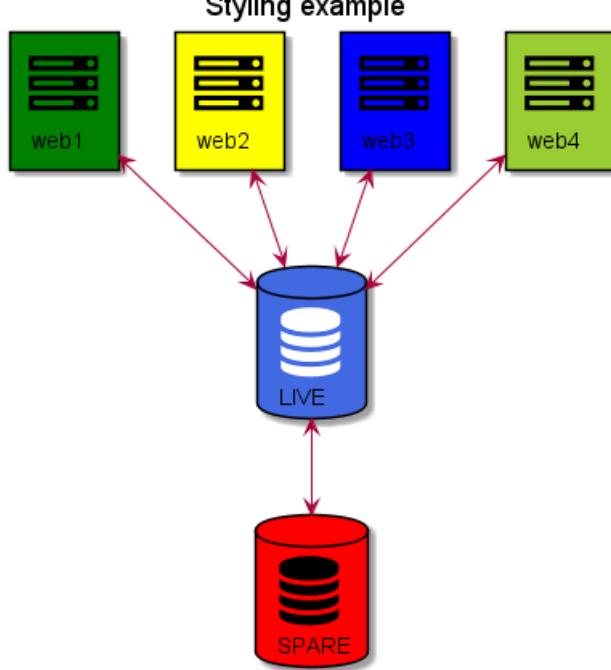
```
FA_SERVER(web1,web1) #Green
FA_SERVER(web2,web2) #Yellow
FA_SERVER(web3,web3) #Blue
FA_SERVER(web4,web4) #YellowGreen
```



```
FA_DATABASE(db1,LIVE,database,white) #RoyalBlue  
FA_DATABASE(db2,SPARE,database) #Red
```

db1 <--> db2

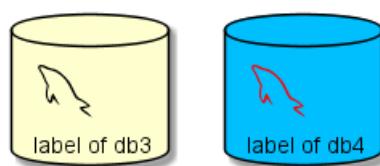
```
web1 <--> db1  
web2 <--> db1  
web3 <--> db1  
web4 <--> db1  
@enduml
```



```
@startuml  
!include <tupaddr3/common>  
!include <tupaddr3/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
```

The image shows two rectangular boxes side-by-side. Both boxes contain a black silhouette of a shark's dorsal fin. The box on the left contains a single question mark character ('?') positioned below the fin. The box on the right contains the text 'label of db2' positioned below the fin.



## Contents

<b>1 时序图</b>	<b>1</b>
1.1 简单示例 . . . . .	1
1.2 声明参与者 . . . . .	1
1.3 在参与者中使用非字母符号 . . . . .	3
1.4 给自己发消息 . . . . .	3
1.5 文本对齐 . . . . .	4
1.5.1 让响应信息显示在箭头下面 . . . . .	4
1.6 修改箭头样式 . . . . .	4
1.7 修改箭头颜色 . . . . .	5
1.8 对消息序列编号 . . . . .	5
1.9 页面标题, 页眉, 页脚 . . . . .	7
1.10 分割示意图 . . . . .	8
1.11 组合消息 . . . . .	8
1.12 次级分组标签 . . . . .	9
1.13 给消息添加注释 . . . . .	10
1.14 其他的注释 . . . . .	11
1.15 改变备注框的形状 [hnote 和 rnote] . . . . .	12
1.16 在多个参与者添加备注 [across] . . . . .	13
1.17 在同一级对齐多个备注 [/] . . . . .	13
1.18 Creole 和 HTML . . . . .	14
1.19 分隔符 . . . . .	15
1.20 引用 . . . . .	16
1.21 延迟 . . . . .	16
1.22 文本换行 . . . . .	17
1.23 空间 . . . . .	17
1.24 生命线的激活与撤销 . . . . .	18
1.25 返回 . . . . .	20
1.26 创建参与者 . . . . .	20
1.27 激活、撤销和创建的快捷语法。 . . . . .	21
1.28 进入和发出消息 . . . . .	23
1.29 缩短的进入信息与发出信息箭头 . . . . .	24
1.30 锚定和持续时间 . . . . .	25
1.31 构造类型和圈点 . . . . .	25
1.32 更多标题信息 . . . . .	26
1.33 包裹参与者 . . . . .	28
1.34 移除脚注 . . . . .	28
1.35 外观参数 (skinparam) . . . . .	29
1.36 填充区设置 . . . . .	31
1.37 附录: 箭头类型大全 . . . . .	32
1.37.1 普通箭头 . . . . .	32
1.37.2 进入信息和发出信息 (使用'[, ]') . . . . .	33
1.37.3 进入信息 (使用'[') . . . . .	33
1.37.4 发出信息 (使用'])') . . . . .	34
1.37.5 短进入信息和短发出信息 (使用'??') . . . . .	35
1.37.6 短进入信息 (使用'??') . . . . .	35
1.37.7 短发出信息 (使用'??') . . . . .	37
1.38 特定外观参数 . . . . .	38
1.38.1 默认情况下 . . . . .	38
1.38.2 生命线策略 . . . . .	39
1.38.3 style strictuml . . . . .	39
1.39 隐藏孤立参与者 . . . . .	40
<b>2 用例图</b>	<b>41</b>
2.1 用例 . . . . .	41
2.2 角色 . . . . .	41
2.3 改变角色的样式 . . . . .	42
2.3.1 火柴人 默认 . . . . .	42



2.3.2 用户头像 . . . . .	42
2.3.3 透明人 . . . . .	42
2.4 用例描述 . . . . .	43
2.5 使用包 . . . . .	43
2.6 基础示例 . . . . .	45
2.7 继承 . . . . .	45
2.8 使用注释 . . . . .	46
2.9 构造类型 . . . . .	46
2.10 改变箭头方向 . . . . .	47
2.11 分割图示 . . . . .	48
2.12 从左向右方向 . . . . .	48
2.13 显示参数 . . . . .	49
2.14 一个完整的例子 . . . . .	50
2.15 Business Use Case . . . . .	50
2.15.1 Business Usecase . . . . .	51
2.15.2 Business Actor . . . . .	51
2.16 Change arrow color and style (inline style) . . . . .	51
2.17 Change element color and style (inline style) . . . . .	52
<b>3 类图</b> . . . . .	<b>53</b>
3.1 元素声明 . . . . .	53
3.2 类之间的关系 . . . . .	53
3.3 关系上的标签 . . . . .	54
3.4 添加方法 . . . . .	55
3.5 定义可访问性 . . . . .	56
3.6 抽象与静态 . . . . .	57
3.7 高级类体 . . . . .	57
3.8 备注和模板 . . . . .	58
3.9 更多注释 . . . . .	59
3.10 Note on field (field, attribute, member) or method . . . . .	60
3.10.1 Note on field or method . . . . .	60
3.10.2 Note on method with the same name . . . . .	60
3.11 链接的注释 . . . . .	61
3.12 抽象类和接口 . . . . .	61
3.13 使用非字母字符 . . . . .	62
3.14 隐藏属性、函数等 . . . . .	63
3.15 隐藏类 . . . . .	64
3.16 Remove classes . . . . .	64
3.17 Hide or Remove unlinked class . . . . .	64
3.18 泛型 (generics) . . . . .	65
3.19 指定标记 (Spot) . . . . .	66
3.20 包 . . . . .	66
3.21 包样式 . . . . .	67
3.22 命名空间 (Namespaces) . . . . .	68
3.23 自动创建命名空间 . . . . .	68
3.24 棒棒糖接口 . . . . .	69
3.25 改变箭头方向 . . . . .	69
3.26 “关系”类 . . . . .	70
3.27 Association on same classe . . . . .	71
3.28 皮肤参数 . . . . .	72
3.29 Skinned Stereotypes . . . . .	72
3.30 Color gradient . . . . .	73
3.31 辅助布局 . . . . .	74
3.32 拆分大文件 . . . . .	74
3.33 Extends and implements . . . . .	75
3.34 Bracketed relations (linking or arrow) style . . . . .	76
3.34.1 Line style . . . . .	76
3.34.2 Line color . . . . .	77



3.34.3 Line thickness . . . . .	77
3.34.4 Mix . . . . .	78
3.35 Change relation (linking or arrow) color and style (inline style) . . . . .	78
3.36 Change class color and style (inline style) . . . . .	79
3.37 Arrows from/to class members . . . . .	80
<b>4 对象图</b>	<b>82</b>
4.1 对象的定义 . . . . .	82
4.2 对象之间的关系 . . . . .	82
4.3 Associations objects . . . . .	82
4.4 添加属性 . . . . .	83
4.5 类图中的通用特性 . . . . .	83
4.6 Map table or associative array . . . . .	84
<b>5 活动图</b>	<b>86</b>
5.1 简单活动 . . . . .	86
5.2 箭头上的标签 . . . . .	86
5.3 改变箭头方向 . . . . .	86
5.4 分支 . . . . .	87
5.5 更多分支 . . . . .	88
5.6 同步 . . . . .	89
5.7 长的活动描述 . . . . .	90
5.8 注释 . . . . .	90
5.9 分区 . . . . .	91
5.10 显示参数 . . . . .	92
5.11 八边形活动 . . . . .	93
5.12 一个完整的例子 . . . . .	93
<b>6 活动图 (新语法)</b>	<b>96</b>
6.1 简单活动图 . . . . .	96
6.2 开始/结束 . . . . .	96
6.3 条件语句 . . . . .	97
6.4 Conditional with stop on an action [kill, detach] . . . . .	98
6.5 重复循环 . . . . .	99
6.6 Break on a repeat loop [break] . . . . .	100
6.7 while 循环 . . . . .	101
6.8 并行处理 . . . . .	102
6.9 Split processing . . . . .	103
6.9.1 Split . . . . .	103
6.9.2 Input split (multi-start) . . . . .	103
6.9.3 Output split (multi-end) . . . . .	104
6.10 注释 . . . . .	105
6.11 颜色 . . . . .	106
6.12 Lines without arrows . . . . .	106
6.13 箭头 . . . . .	107
6.14 连接器 (Connector) . . . . .	108
6.15 Color on connector . . . . .	108
6.16 组合 (grouping) . . . . .	109
6.17 泳道 (Swimlanes) . . . . .	110
6.18 分离 (detach) . . . . .	110
6.19 特殊领域语言 (SDL) . . . . .	111
6.20 一个完整的例子 . . . . .	112
6.21 Condition Style . . . . .	114
6.21.1 Inside style (by default) . . . . .	114
6.21.2 Diamond style . . . . .	115
6.21.3 InsideDiamond (or <i>Foo1</i> ) style . . . . .	116
6.22 Condition End Style . . . . .	117
6.22.1 Diamond style (by default) . . . . .	117
6.22.2 Horizontal line (hline) style . . . . .	118



<b>7 组件图</b>	<b>120</b>
7.1 组件 . . . . .	120
7.2 接口 . . . . .	120
7.3 基础的示例 . . . . .	120
7.4 使用注释 . . . . .	121
7.5 组合组件 . . . . .	121
7.6 改变箭头方向 . . . . .	123
7.7 Use UML2 notation . . . . .	124
7.8 使用 UML1 标记符 . . . . .	125
7.9 Use rectangle notation (remove UML notation) . . . . .	125
7.10 长描述 . . . . .	125
7.11 不同的颜色表示 . . . . .	126
7.12 在定型组件中使用精灵图 . . . . .	126
7.13 显示参数 . . . . .	127
7.14 Specific SkinParameter . . . . .	128
7.14.1 componentStyle . . . . .	128
7.15 Hide or Remove unlinked component . . . . .	129
<b>8 部署图</b>	<b>131</b>
8.1 声明元素 . . . . .	131
8.2 Declaring element (using short form) . . . . .	133
8.2.1 Actor . . . . .	133
8.2.2 Component . . . . .	133
8.2.3 Interface . . . . .	133
8.2.4 Usecase . . . . .	133
8.3 链接 . . . . .	134
8.4 Bracketed arrow style . . . . .	135
8.4.1 Line style . . . . .	136
8.4.2 Line color . . . . .	136
8.4.3 Line thickness . . . . .	137
8.4.4 Mix . . . . .	137
8.5 Change arrow color and style (inline style) . . . . .	138
8.6 Change element color and style (inline style) . . . . .	138
8.7 Nestable elements . . . . .	139
8.8 包装 . . . . .	140
8.9 Alias . . . . .	140
8.9.1 Simple alias with as . . . . .	140
8.9.2 Examples of long alias . . . . .	141
8.10 圆角 . . . . .	143
8.11 Specific SkinParameter . . . . .	143
8.11.1 roundCorner . . . . .	143
8.12 Appendix: All type of arrow line . . . . .	144
8.13 Appendix: All type of arrow head or '0' arrow . . . . .	145
8.13.1 Type of arrow head . . . . .	145
8.13.2 Type of '0' arrow or circle arrow . . . . .	146
8.14 Appendix: Test of inline style on all element . . . . .	147
8.14.1 Simple element . . . . .	147
8.14.2 Nested element . . . . .	148
8.14.3 Without sub-element . . . . .	148
8.14.4 With sub-element . . . . .	149
8.15 Appendix: Test of style on all element . . . . .	150
8.15.1 Simple element . . . . .	150
8.15.2 Global style (on componentDiagram) . . . . .	150
8.15.3 Style for each element . . . . .	151
8.15.4 Nested element (without level) . . . . .	154
8.15.5 Global style (on componentDiagram) . . . . .	154
8.15.6 Style for each nested element . . . . .	155
8.15.7 Nested element (with one level) . . . . .	157



8.15.8 Global style (on componentDiagram) . . . . .	157
8.15.9 Style for each nested element . . . . .	158
<b>9 状态图</b>	<b>161</b>
9.1 简单状态 . . . . .	161
9.2 Change state rendering . . . . .	161
9.3 合成状态 . . . . .	162
9.4 长名字 . . . . .	163
9.5 History [[H], [H*]] . . . . .	164
9.6 Fork [fork, join] . . . . .	164
9.7 并发状态 . . . . .	165
9.8 Conditional [choice] . . . . .	166
9.9 Stereotypes full example [choice, fork, join, end] . . . . .	167
9.10 Point [entryPoint, exitPoint] . . . . .	168
9.11 Pin [inputPin, outputPin] . . . . .	169
9.12 Expansion [expansionInput, expansionOutput] . . . . .	170
9.13 箭头方向 . . . . .	171
9.14 Change line color and style . . . . .	171
9.15 注释 . . . . .	172
9.16 Note on link . . . . .	173
9.17 更多注释 . . . . .	173
9.18 Inline color . . . . .	174
9.19 显示参数 . . . . .	175
9.20 Changing style . . . . .	176
9.21 Change state color and style (inline style) . . . . .	177
<b>10 定时图</b>	<b>179</b>
10.1 声明参与者 . . . . .	179
10.2 Binary and Clock . . . . .	179
10.3 增加消息 . . . . .	180
10.4 相对时间 . . . . .	180
10.5 Anchor Points . . . . .	181
10.6 Participant oriented . . . . .	182
10.7 Setting scale . . . . .	182
10.8 Initial state . . . . .	182
10.9 Intricated state . . . . .	183
10.10 Hidden state . . . . .	184
10.11 Hide time axis . . . . .	184
10.12 Using Time and Date . . . . .	184
10.13 Adding constraint . . . . .	185
10.14 Highlighted period . . . . .	186
10.15 Adding texts . . . . .	187
10.16 Complete example . . . . .	187
10.17 Digital Example . . . . .	188
10.18 Adding color . . . . .	190
<b>11 Display JSON Data</b>	<b>191</b>
11.1 Complex example . . . . .	191
11.2 Highlight parts . . . . .	192
11.3 JSON basic element . . . . .	192
11.3.1 Synthesis of all JSON basic element . . . . .	192
11.4 JSON array or table . . . . .	193
11.4.1 Array type . . . . .	193
11.4.2 Minimal array or table . . . . .	194
11.4.3 Number array . . . . .	194
11.4.4 String array . . . . .	194
11.4.5 Boolean array . . . . .	194
11.5 JSON numbers . . . . .	194
11.6 JSON strings . . . . .	195



11.6.1 JSON Unicode . . . . .	195
11.6.2 JSON two-character escape sequence . . . . .	195
11.7 Minimal JSON examples . . . . .	196
11.8 Using (global) style . . . . .	197
11.8.1 Without style ( <i>by default</i> ) . . . . .	197
11.8.2 With style . . . . .	197
<b>12 Display YAML Data</b>	<b>199</b>
12.1 Complex example . . . . .	199
12.2 Specific key (with symbols or unicode) . . . . .	200
12.3 Highlight parts . . . . .	200
12.3.1 Normal style . . . . .	200
12.3.2 Customised style . . . . .	201
12.4 Using (global) style . . . . .	201
12.4.1 Without style ( <i>by default</i> ) . . . . .	201
12.4.2 With style . . . . .	202
<b>13 Network diagram (nwdiag)</b>	<b>204</b>
13.1 Simple diagram . . . . .	204
13.2 Define multiple addresses . . . . .	204
13.3 Grouping nodes . . . . .	205
13.3.1 Define group inside network definitions . . . . .	205
13.3.2 Define group outside of network definitions . . . . .	206
13.3.3 Define several groups on same network . . . . .	206
13.3.4 Example with 2 group . . . . .	206
13.3.5 Example with 3 groups . . . . .	207
13.4 Extended Syntax (for network or group) . . . . .	208
13.4.1 Network . . . . .	208
13.4.2 Group . . . . .	209
13.5 Using Sprites . . . . .	210
13.6 Using OpenIconic . . . . .	211
13.7 Same nodes on more than two networks . . . . .	212
13.8 Peer networks . . . . .	213
13.9 Peer networks and group . . . . .	213
13.9.1 Without group . . . . .	213
13.9.2 Group on first . . . . .	214
13.9.3 Group on second . . . . .	215
13.9.4 Group on third . . . . .	216
13.10 Add title, caption, header, footer or legend on network diagram . . . . .	217
13.11 Change width of the networks . . . . .	218
13.12 Other internal networks . . . . .	220
<b>14 Salt (线框图)</b>	<b>223</b>
14.1 基本部件 . . . . .	223
14.2 使用表格 . . . . .	223
14.3 分组框 . . . . .	224
14.4 使用分隔符 . . . . .	224
14.5 树形外挂 . . . . .	225
14.6 Tree table [T] . . . . .	225
14.7 Enclosing brackets [{, }]	227
14.8 添加选项卡 . . . . .	227
14.9 使用菜单 . . . . .	228
14.10 高级表格 . . . . .	229
14.11 Scroll Bars [S, SI, S-] . . . . .	229
14.12 Colors . . . . .	230
14.13 Pseudo sprite [«, »]	231
14.14 OpenIconic . . . . .	231
14.15 Include Salt "on activity diagram"	232
14.16 Include salt "on while condition of activity diagram"	235



<b>15 Archimate Diagram</b>	<b>236</b>
15.1 Archimate keyword . . . . .	236
15.2 Defining Junctions . . . . .	236
15.3 Example 1 . . . . .	237
15.4 Example 2 . . . . .	238
15.5 List possible sprites . . . . .	239
15.6 ArchiMate Macros . . . . .	239
15.6.1 Archimate Macros and Library . . . . .	239
15.6.2 Archimate elements . . . . .	239
15.6.3 Archimate relationships . . . . .	240
15.6.4 Appendix: Examples of all Archimate RelationTypes . . . . .	241
<b>16 Gantt Diagram</b>	<b>243</b>
16.1 Declaring tasks . . . . .	243
16.1.1 Duration . . . . .	243
16.1.2 Start . . . . .	243
16.1.3 End . . . . .	243
16.1.4 Start/End . . . . .	244
16.2 One-line declaration (with the and conjunction) . . . . .	244
16.3 Adding constraints . . . . .	244
16.4 Short names . . . . .	245
16.5 Customize colors . . . . .	245
16.6 Completion status . . . . .	245
16.7 Milestone . . . . .	245
16.7.1 Relative milestone (use of constraints) . . . . .	246
16.7.2 Absolute milestone (use of fixed date) . . . . .	246
16.7.3 Milestone of maximum end of tasks . . . . .	246
16.8 Hyperlinks . . . . .	246
16.9 Calendar . . . . .	247
16.10 Coloring days . . . . .	247
16.11 Changing scale . . . . .	247
16.11.1 Daily ( <i>by default</i> ) . . . . .	248
16.11.2 Weekly . . . . .	248
16.11.3 Monthly . . . . .	249
16.12 Close day . . . . .	249
16.13 Simplified task succession . . . . .	250
16.14 Separator . . . . .	250
16.15 Working with resources . . . . .	250
16.16 Complex example . . . . .	251
16.17 Comments . . . . .	252
16.18 Using style . . . . .	252
16.18.1 Without style ( <i>by default</i> ) . . . . .	252
16.18.2 With style . . . . .	252
16.19 Add notes . . . . .	254
16.20 Pause tasks . . . . .	256
16.21 Change link colors . . . . .	256
16.22 Tasks or Milestones on the same line . . . . .	257
16.23 Highlight today . . . . .	257
16.24 Task between two milestones . . . . .	257
16.25 Grammar and verbal form . . . . .	258
16.26 Add title, header, footer, caption or legend on gantt diagram . . . . .	258
16.27 Removing Foot Boxes . . . . .	258
<b>17 思维导图</b>	<b>261</b>
17.1 OrgMode 语法 . . . . .	261
17.2 Markdown 语法 . . . . .	261
17.3 运算符 . . . . .	262
17.4 Multilines . . . . .	262
17.5 Colors . . . . .	263



17.5.1 With inline color . . . . .	263
17.5.2 With style color . . . . .	264
17.6 去除外边框 . . . . .	266
17.7 改变图形方向 . . . . .	266
17.8 完整示例 . . . . .	267
17.9 Changing style . . . . .	268
17.9.1 node, depth . . . . .	268
17.9.2 boxless . . . . .	269
17.10 Word Wrap . . . . .	269
<b>18 Work Breakdown Structure (WBS)</b>	<b>271</b>
18.1 OrgMode syntax . . . . .	271
18.2 Change direction . . . . .	271
18.3 Arithmetic notation . . . . .	272
18.4 Removing box . . . . .	272
18.4.1 Boxless on Arithmetic notation . . . . .	273
18.4.2 Several boxless node . . . . .	273
18.4.3 All boxless node . . . . .	273
18.4.4 Boxless on OrgMode syntax . . . . .	274
18.4.5 Several boxless node . . . . .	274
18.4.6 All boxless node . . . . .	274
18.5 Colors (with inline or style color) . . . . .	275
18.6 Using style . . . . .	276
18.7 Word Wrap . . . . .	277
<b>19 简介</b>	<b>280</b>
19.1 独立图 . . . . .	280
19.2 这是如何工作的?	281
<b>20 Entity Relationship Diagram</b>	<b>282</b>
20.1 Information Engineering Relations . . . . .	282
20.2 Entities . . . . .	282
20.3 Complete Example . . . . .	283
<b>21 通用命令</b>	<b>285</b>
21.1 注释 . . . . .	285
21.2 缩放 . . . . .	285
21.3 标题 . . . . .	285
21.4 图片标题 . . . . .	286
21.5 页眉和页脚 . . . . .	287
21.6 图例说明 . . . . .	287
21.7 Appendix: Examples on all diagram . . . . .	288
21.7.1 Activity . . . . .	288
21.7.2 Archimate . . . . .	288
21.7.3 Class . . . . .	289
21.7.4 Component, Deployment, Use-Case . . . . .	290
21.7.5 Gantt project planning . . . . .	290
21.7.6 Object . . . . .	291
21.7.7 MindMap . . . . .	291
21.7.8 Network (nwdiag) . . . . .	292
21.7.9 Sequence . . . . .	293
21.7.10 State . . . . .	293
21.7.11 Timing . . . . .	294
21.7.12 Work Breakdown Structure (WBS) . . . . .	295
21.7.13 Wireframe (SALT) . . . . .	295
21.8 Appendix: Examples on all diagram with style . . . . .	296
21.8.1 Activity . . . . .	297
21.8.2 Archimate . . . . .	298
21.8.3 Class . . . . .	300



21.8.4 Component, Deployment, Use-Case . . . . .	301
21.8.5 Gantt project planning . . . . .	302
21.8.6 Object . . . . .	303
21.8.7 MindMap . . . . .	305
21.8.8 Network (nwdiag) . . . . .	306
21.8.9 Sequence . . . . .	307
21.8.10 State . . . . .	309
21.8.11 Timing . . . . .	310
21.8.12 Work Breakdown Structure (WBS) . . . . .	312
21.8.13 Wireframe (SALT) . . . . .	313
<b>22 Creole</b>	<b>315</b>
22.1 Emphasized text . . . . .	315
22.2 Lists . . . . .	315
22.3 Escape character . . . . .	316
22.4 Horizontal lines . . . . .	316
22.5 Headings . . . . .	317
22.6 Legacy HTML . . . . .	317
22.7 Code . . . . .	318
22.8 Table . . . . .	319
22.8.1 Create a table . . . . .	319
22.8.2 Add color on rows or cells . . . . .	320
22.8.3 Add color on border and text . . . . .	320
22.8.4 No border or same color as the background . . . . .	320
22.8.5 Bold header or not . . . . .	321
22.9 Tree . . . . .	321
22.10 Special characters . . . . .	323
22.11 OpenIconic . . . . .	323
22.12 Appendix: Examples of "Creole List" on all diagrams . . . . .	324
22.12.1 Activity . . . . .	324
22.12.2 Class . . . . .	325
22.12.3 Component, Deployment, Use-Case . . . . .	326
22.12.4 Gantt project planning . . . . .	327
22.12.5 Object . . . . .	327
22.12.6 MindMap . . . . .	328
22.12.7 Network (nwdiag) . . . . .	328
22.12.8 Note . . . . .	328
22.12.9 Sequence . . . . .	329
22.12.10 State . . . . .	329
22.13 Appendix: Examples of "Creole horizontal lines" on all diagrams . . . . .	329
22.13.1 Activity . . . . .	329
22.13.2 Class . . . . .	330
22.13.3 Component, Deployment, Use-Case . . . . .	331
22.13.4 Gantt project planning . . . . .	332
22.13.5 Object . . . . .	332
22.13.6 MindMap . . . . .	332
22.13.7 Network (nwdiag) . . . . .	333
22.13.8 Note . . . . .	333
22.13.9 Sequence . . . . .	334
22.13.10 State . . . . .	334
22.14 Style equivalent (between Creole and HTML) . . . . .	334
<b>23 Defining and using sprites</b>	<b>336</b>
23.1 Changing colors . . . . .	337
23.2 Encoding Sprite . . . . .	337
23.3 Importing Sprite . . . . .	337
23.4 Examples . . . . .	337
23.5 StdLib . . . . .	338
23.6 Listing Sprites . . . . .	338



<b>24 Skinparam 命令</b>	<b>340</b>
24.1 使用 . . . . .	340
24.2 内嵌 . . . . .	340
24.3 黑白 (Black and White) . . . . .	340
24.4 Shadowing . . . . .	341
24.5 颜色翻转 (Reverse colors) . . . . .	342
24.6 颜色 (Colors) . . . . .	343
24.7 字体颜色、名称、大小 (Font color, name and size) . . . . .	343
24.8 文本对齐 (Text Alignment) . . . . .	344
24.9 Examples . . . . .	344
24.10 所有 skinparam 的参数列表 (List of all skinparam parameters) . . . . .	348
<b>25 预处理</b>	<b>351</b>
25.1 迁移说明 . . . . .	351
25.2 定义变量 . . . . .	351
25.3 Boolean expression . . . . .	352
25.3.1 Boolean representation [0 is false] . . . . .	352
25.3.2 Boolean operation and operator [&&,   , ()] . . . . .	352
25.3.3 Boolean builtin functions [%false(), %true(), %not(<exp>)] . . . . .	352
25.4 条件 . . . . .	352
25.5 While loop [!while, !endwhile] . . . . .	353
25.6 空函数 . . . . .	354
25.7 返回函数 . . . . .	355
25.8 参数默认值 . . . . .	356
25.9 非引号函数 . . . . .	357
25.10 Keywords arguments . . . . .	357
25.11 Including files or URL [!include, !include_many, !include_once] . . . . .	358
25.12 Including Subpart [!startsub, !endsub, !includesub] . . . . .	359
25.13 Builtin functions [%] . . . . .	359
25.14 Logging [!log] . . . . .	360
25.15 Memory dump [!memory_dump] . . . . .	360
25.16 Assertion [!assert] . . . . .	361
25.17 Building custom library [!import, !include] . . . . .	361
25.18 Search path . . . . .	362
25.19 Argument concatenation [##] . . . . .	362
25.20 Dynamic invocation [%invoke_procedure(), %call_user_func()] . . . . .	362
25.21 Evaluation of addition depending of data types [+] . . . . .	363
25.22 Preprocessing JSON . . . . .	364
<b>26 Unicode</b>	<b>365</b>
26.1 Examples . . . . .	365
26.2 Charset . . . . .	367
<b>27 Standard Library</b>	<b>368</b>
27.1 List of Standard Library . . . . .	368
27.2 ArchiMate [archimate] . . . . .	369
27.3 AWS library [aws] . . . . .	370
27.4 Amazon Labs AWS Library [awslib] . . . . .	371
27.5 Azure library [azure] . . . . .	372
27.6 C4 Library [C4] . . . . .	373
27.7 Cloud Insight [cloudinsight] . . . . .	374
27.8 Cloudogu [cloudogu] . . . . .	375
27.9 Elastic library [elastic] . . . . .	376
27.10 Google Material Icons [material] . . . . .	378
27.11 Kubernetes [kubernetes] . . . . .	379
27.12 Logos [logos] . . . . .	380
27.13 Office [office] . . . . .	382
27.14 Open Security Architecture (OSA) [osa] . . . . .	384
27.15 Tupadr3 library [tupadr3] . . . . .	386

