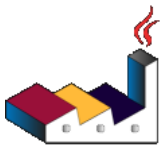
- Sequence
  - 简单示例（Basic examples）
  - 声明参与者（Declaring participant）
  - 在参与者中使用非字母符号（Use non-letters in participants）
  - 给自己发消息（Message to self）
  - 修改箭头样式（Change arrow style）
  - 修改箭头颜色（Change arrow color）
  - 对消息序列编号（Message sequence numbering）
  - 分割示意图(Splitting diagrams)
  - 组合消息
  - 给消息添加注释
  - 其他的注释
  - 改变备注框的形状
  - Creole和HTML
  - 分隔符
  - 引用
  - 延迟
  - 空间
  - 生命线的激活与撤销
  - 创建参与者
  - 进入和发出消息
  - 构造类型和圈点
  - 包裹参与者
  - 移除脚注
  - 外观参数(skinparam)
- Use Case
  - 用例
  - 角色
  - 用例描述
  - 基础示例
  - 继承
  - 使用注释
  - 构造类型
  - 改变箭头方向
  - 分割图示
  - 从左向右方向
  - 显示参数
  - 一个完整的例子
- Class
  - 类之间的关系
  - 关系上的标识
  - 添加方法
  - 定义可访问性
  - 抽象与静态
  - 高级类体
  - 备注和模板
  - 更多注释
  - 链接的注释
  - 抽象类和接口
  - 使用非字母字符
  - 隐藏属性、函数等
  - 隐藏类
  - 泛型（generics）
  - 指定标记（Spot）
  - 包
  - 包样式
  - 命名空间（Namespaces）
  - 自动创建命名空间
  - 棒棒糖 接口
  - 改变箭头方向
  - “关系”类
  - 皮肤参数
  - Skinned Stereotypes
  - Color gradient
  - Help on layout
  - 拆分大文件
- Activity
  - 简单活动
  - 箭头上的标签
  - 改变箭头方向
  - 分支
  - 更多分支
  - 同步

    - 长的活动描述
    - 注释
    - 分区
    - 显示参数
    - 八边形活动
    - 一个完整的例子
- Activity BETA
    - 简单活动图
    - 开始/结束
    - 条件语句
    - 重复循环
    - while循环
    - 并行处理
    - 注释
    - 颜色
    - 箭头
    - 组合(grouping)
    - 泳道(Swimlanes)
    - 分离(detach)
    - 特殊领域语言(SDL)
    - 一个完整的例子
- Component
    - 组件
    - 接口
    - 基础的示例
    - 使用注释
    - 组合组件
    - 改变箭头方向
    - 使用UML2标记符
    - Long description
    - 不同的颜色表示
    - Sprite in Stereotype
    - 显示参数
- State
    - 简单状态
    - 合成状态
    - 长名字
    - 并发状态
    - 箭头方向
    - 注释
    - 更多注释
    - 显示参数
- Object
    - 对象的定义
    - 对象之间的关系
    - 添加属性
    - 可见性
    - 定义注释
    - 使用包
    - 美化输出内容
    - 分割图片
- Deployment BETA
    - Declaring element
    - Linking
    - Packages
    - Round corner
- Timing BETA
    - Declaring participant
    - Adding message
    - Relative time
    - Participant oriented
    - Setting scale
    - Initial state

# 时序图（Sequence Diagram）



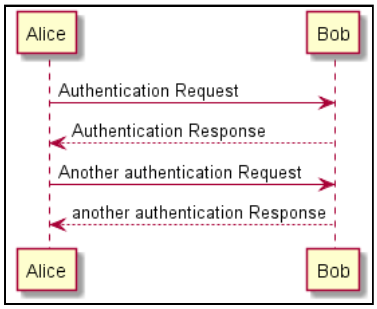
## 简单示例（Basic examples）

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

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



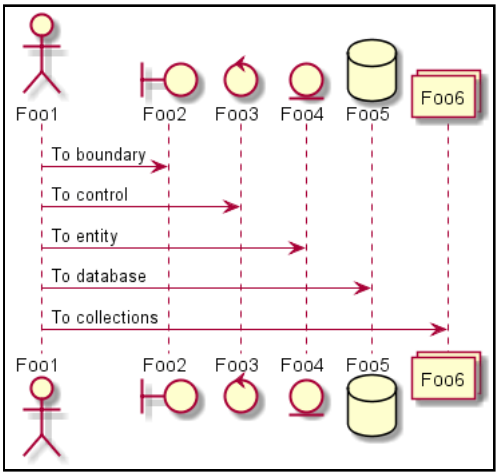

## 声明参与者（Declaring participant）

关键字 `participant` 用于改变参与者的先后顺序。

你也可以使用其它关键字来声明参与者：

- actor
- boundary
- control
- entity
- database

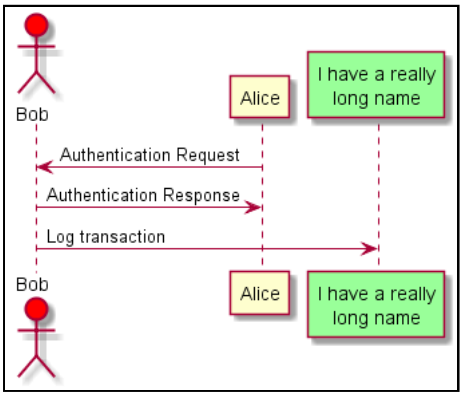
```
@startuml
actor Foo1
boundary Foo2
control Foo3
entity Foo4
database Foo5
collections Foo6
Foo1 -> Foo2 : To boundary
Foo1 -> Foo3 : To control
Foo1 -> Foo4 : To entity
Foo1 -> Foo5 : To database
Foo1 -> Foo6 : To collections

@enduml
```




关键字 as 用于重命名参与者

你可以使用RGB值或者颜色名修改 actor 或参与者的背景颜色。

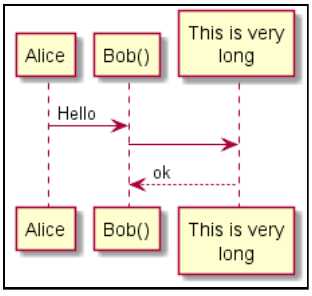
```
@startuml
actor Bob #red
' The only difference between actor
'and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
   participant L as "I have a really\nlong name"  #99FF99
  '/

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




## 在参与者中使用非字母符号（Use non-letters in participants）

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

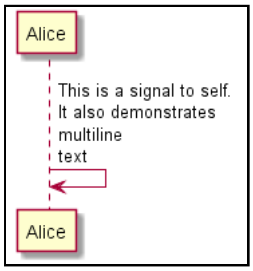



## 给自己发消息（Message to self）

参与者可以给自己发信息，

消息文字可以用\n来换行。

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

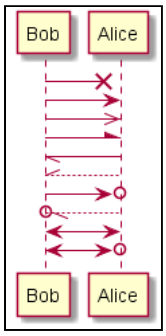



## 修改箭头样式（Change arrow style）

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：<->

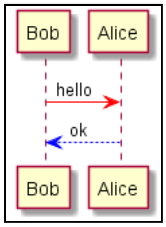
```
@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\- Alice
Bob //-- Alice

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

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




## 修改箭头颜色（Change arrow color）

你可以用以下记号修改箭头的颜色：

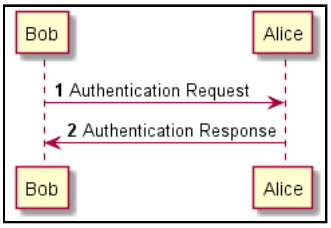
```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```




## 对消息序列编号（Message sequence numbering）

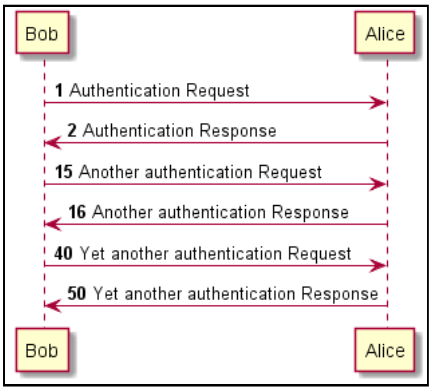
关键字 autonumber 用于自动对消息编号。

```
@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
```

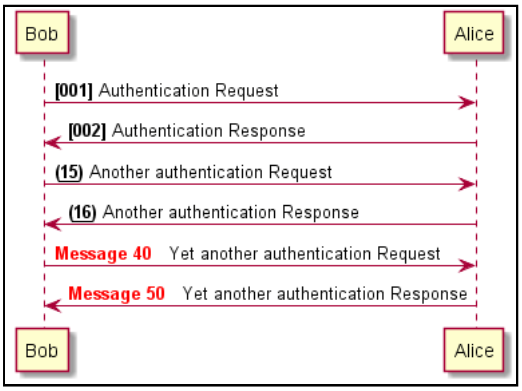



你可以在双引号内指定编号的格式。

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。

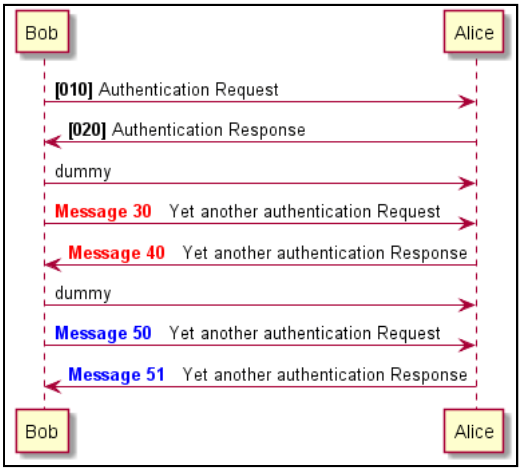
```
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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

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

@enduml
```




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

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

autonumber stop
Bob -> Alice : dummy

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

autonumber stop
```

```
Bob -> Alice : dummy

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



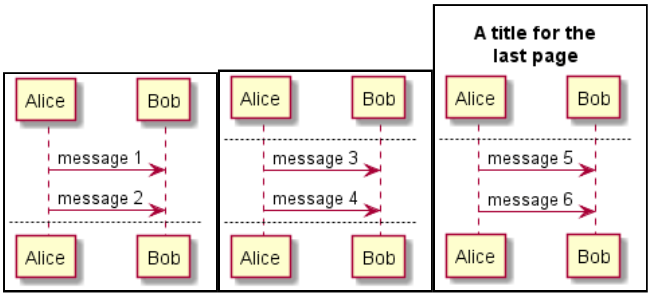

## 分割示意图(Splitting diagrams)

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
```




## 组合消息

我们可以通过以下关键词将组合消息：

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

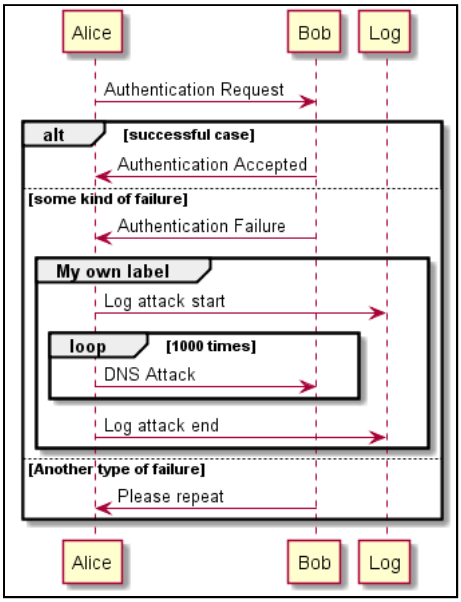
可以在标头(header)添加需要显示的文字(`group`除外)。

关键词 `end` 用来结束分组。

注意，分组可以嵌套使用。

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

alt successful case

    Bob -> Alice: Authentication Accepted
```

```
else some kind of failure

    Bob -> Alice: Authentication Failure
    group My own label
        Alice -> Log : Log attack start
        loop 1000 times
            Alice -> Bob: DNS Attack
        end
        Alice -> Log : Log attack end
    end

else Another type of failure

   Bob -> Alice: Please repeat

end
@enduml
```

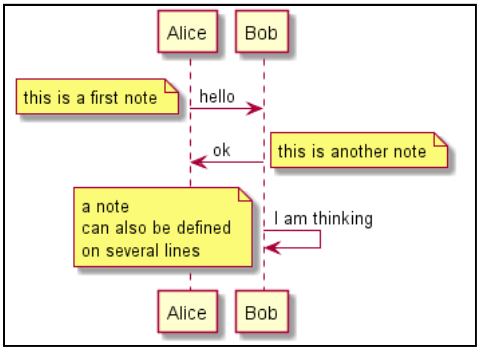



## 给消息添加注释

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
```



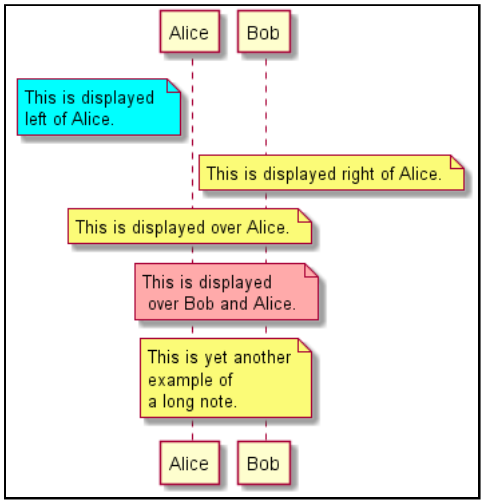

## 其他的注释

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。

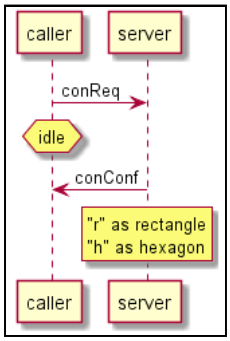
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
        This is displayed
        left of Alice.
end note

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
```




## 改变备注框的形状

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

```
@startuml
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
 "r" as rectangle
 "h" as hexagon
endrnote
@enduml
```

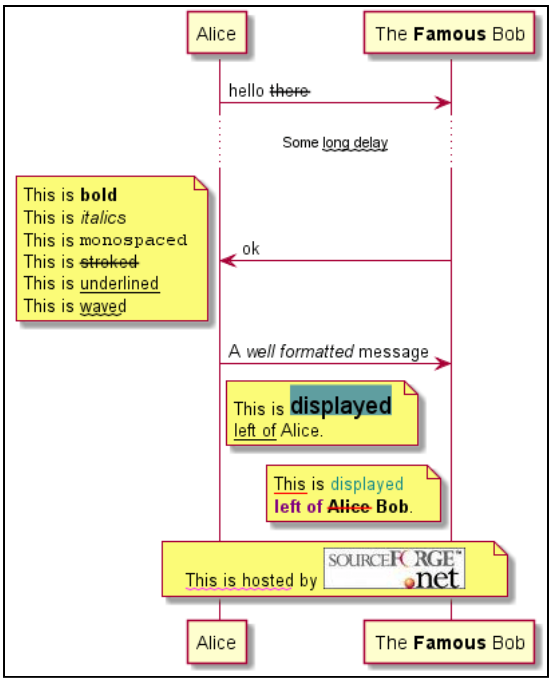



## Creole和HTML

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
```

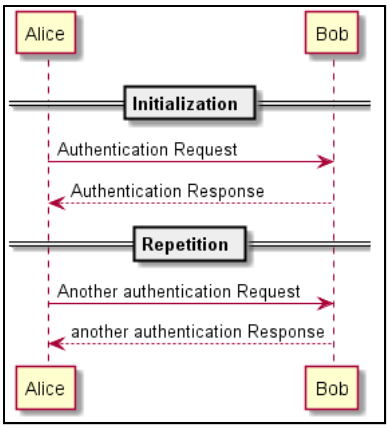
```
note right of Alice
 This is <back:cadetblue><size:18>displayed</size></back>
 __left of__ Alice.
end note
note left of Bob
 <u:red>This</u> is <color #118888>displayed</color>
 **<color purple>left of</color> <s:red>Alice</strike> Bob**.
end note
note over Alice, Bob
 <w:#FF33FF>This is hosted</w> by <img sourceforge.jpg>
end note
@enduml
```




## 分隔符

你可以通过使用 == 关键词来将你的图表分割多个步骤。

```
@startuml

== Initialization ==

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

== Repetition ==

Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response

@enduml
```

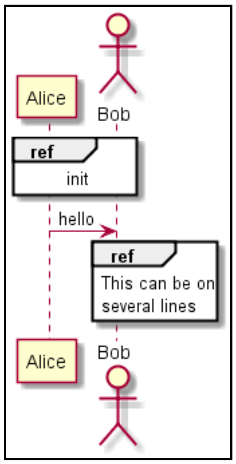



## 引用

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
```

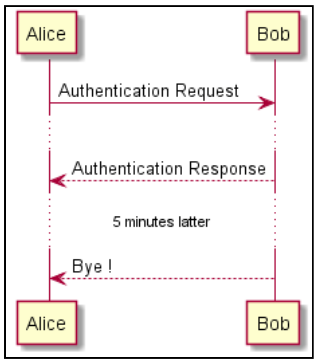
```
ref over Bob
  This can be on
  several lines
end ref
@enduml
```




## 延迟

你可以使用...来表示延迟，并且还可以给延迟添加注释。

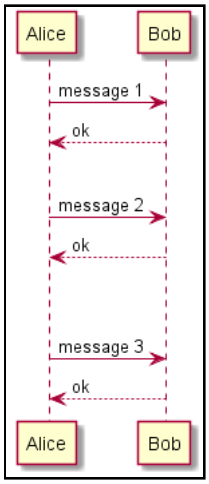
```
@startuml

Alice -> Bob: Authentication Request
...
Bob --> Alice: Authentication Response
...5 minutes latter...
Bob --> Alice: Bye !

@enduml
```




## 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
```




## 生命线的激活与撤销

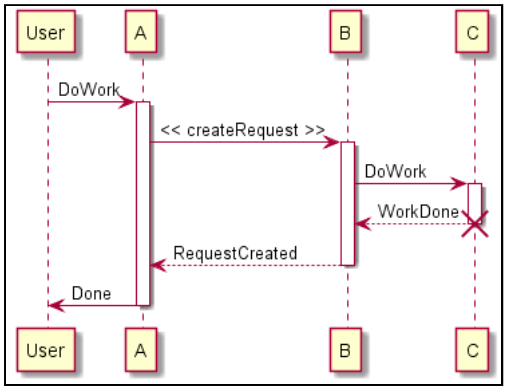
关键字activate和deactivate用来表示参与者的生命活动。

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。

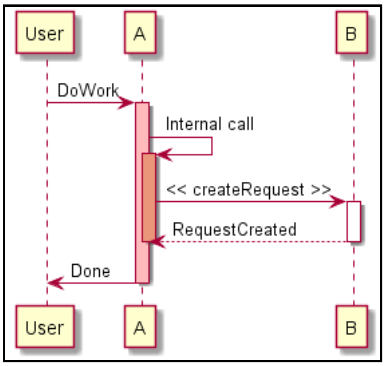
destroy表示一个参与者的生命线的终结。

```
@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。

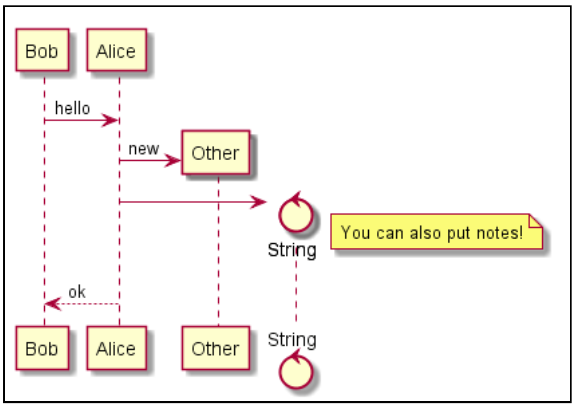
```
@startuml
participant User

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
```




## 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
```

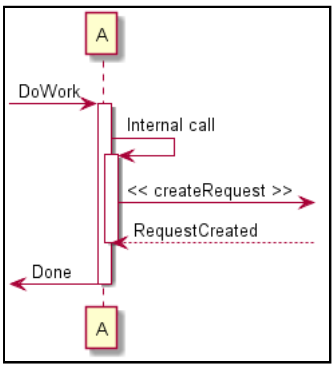



## 进入和发出消息

如果只想关注部分图示，你可以使用进入和发出箭头。

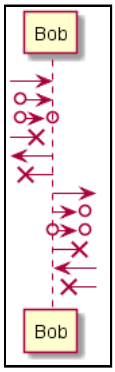
使用方括号[和]表示图示的左、右两侧。

```
@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
[-> Bob
[o-> Bob
[o->o Bob
[x-> Bob

[<- Bob
[x<- Bob

Bob ->]
Bob ->o]
Bob o->o]
Bob ->x]

Bob <-]
Bob x<-]
@enduml
```

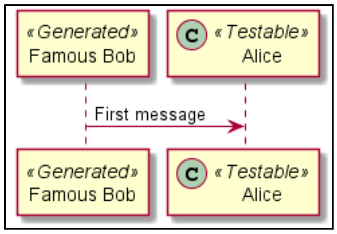



## 构造类型和圈点

可以使用<<和>>给参与者添加构造类型。

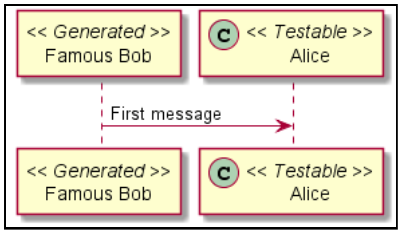
在构造类型中，你可以使用(X,color)格式的语法添加一个圆圈圈起来的字符。

```
@startuml

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

Bob->Alice: First message

@enduml
```

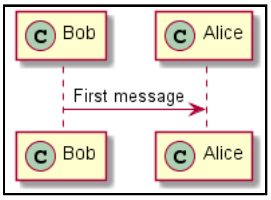



默认使用 *guillemet* 字符来显示构造类型。 你可以使用外观参数 `guillemet` 来修改显示行为。

```
@startuml

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

Bob->Alice: First message

@enduml
```



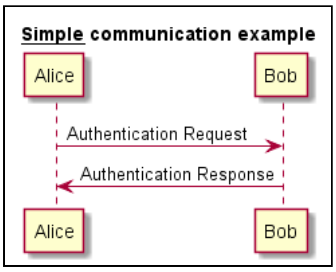

```
@startuml

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

Bob->Alice: First message

@enduml
```




## 更多标题信息

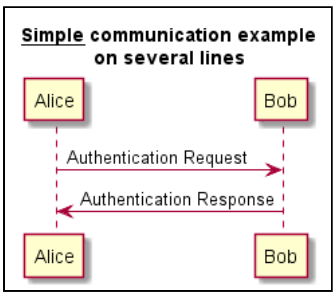
你可以在标题中使用creole格式。

```
@startuml

title __Simple__ **communication** example

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

@enduml
```



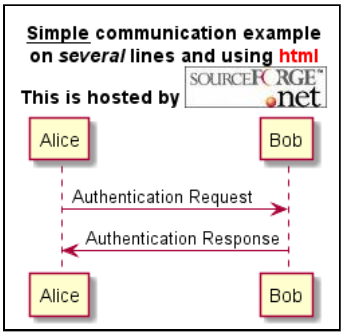

在标题描述中使用\n表示换行。

```
@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
```

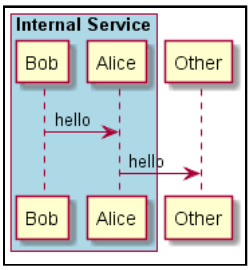



## 包裹参与者

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。

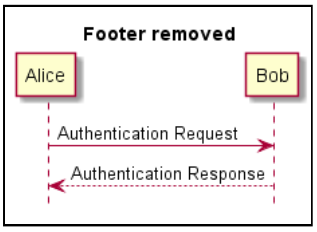
```
@startuml

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

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

@enduml
```




## 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
```




## 外观参数(skinparam)

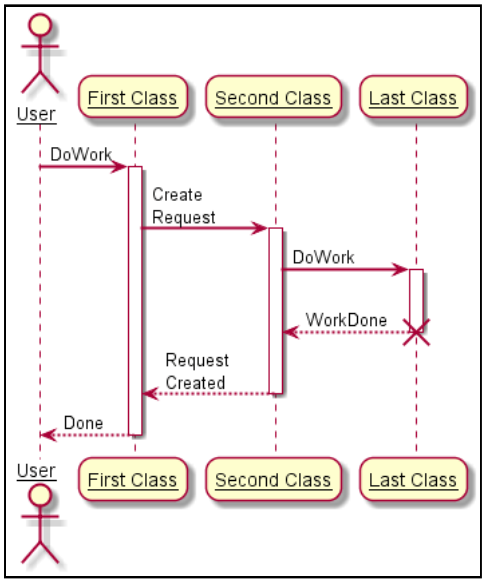
使用skinparam命令改变颜色和字体。

如下场景可以使用这一命令：

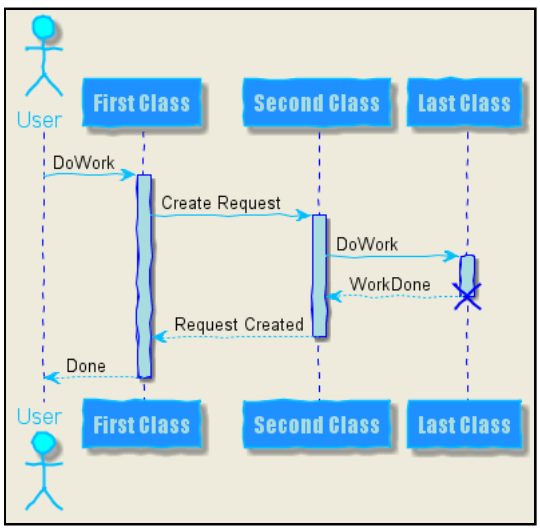
- 在图示定义中，
- 在一个包含文件中,
- 在由命令行或者ANT任务提供的配置文件中。

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
```



## 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
```