



Plant UML

PlantUML 은 다이어그램을 빠르게 작성하기 위한 오픈 소스 프로젝트입니다.

Sault (Wireframe)

Salt is a subproject included in PlantUML that may help you to design graphical interface or [Website Wireframe or Page Schematic or Screen Blueprint](#).

The goal of this tool is to discuss about simple and sample windows.

You can use either `@startsalt` keyword, or `@startuml` followed by a line with `salt` keyword.

Basic widgets

A window must start and end with brackets. You can then define:

- Button using `[` and `]`.
- Radio button using `(` and `)`.
- Checkbox using `[` and `]`.
- User text area using `"`.
- Droplist using `^`.

```
@startsalt
{
    Just plain text
    [This is my button]
    () Unchecked radio
    (X) Checked radio
    [] Unchecked box
    [X] Checked box
    "Enter text here "
    ^This is a droplist^
}
@endsalt
```

Just plain text

This is my button

☐ Unchecked radio

☒ Checked radio

☐ Unchecked box

☒ Checked box

Enter text here

This is a droplist ▼

Here is an attempt to create a [text area](#):

```
@startsalt
{+
  This is a long
  text in a textarea
  .
  "
}
@endsalt
```

This is a long
text in a textarea

Note:

- the dot (.) to fill up vertical space;
- the last line of space (" ") to make the area wider.

Then you can add [scroll bar](#):

```
@startsalt
{SI
  This is a long
  text in a textarea
  .
  "
}
@endsalt
```

This is a long
text in a textarea

▲▼

```
@startsalt
{S-
  This is a long
  text in a textarea
  .
  " "
}
@endsalt
```

This is a long
text in a textarea

◀▶

Open, close droplist

You can open a droplist, by adding values enclosed by `^`, as:

```
@startsalt
{
  ^This is a closed droplist^ |
  ^This is an open droplist^^ item 1^^ item 2^ |
  ^This is another open droplist^ item 1^ item 2^
}
@endsalt
```

This is a closed droplist ▼	This is an open droplist ▼	This is another open droplist ▼
	item 1	item 1
	item 2	item 2

Using grid [| and #, !, -, +]

A table is automatically created when you use an opening bracket `{`. And you have to use `|` to separate columns.

For example:

```
@startsalt
{
  Login      | "MyName  "
  Password   | "****    "
  [Cancel]   | [ OK    ]
}
@endsalt
```

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
<input type="button" value="Cancel"/> <input type="button" value="OK"/>	

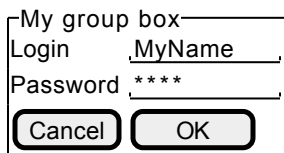
Symbol	Result
#	To display all vertical and horizontal lines
!	To display all vertical lines
-	To display all horizontal lines
+	To display external lines

```
@startsalt
{+
  Login      | "MyName  "
  Password   | "****    "
  [Cancel]   | [ OK    ]
}
@endsalt
```

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
<input type="button" value="Cancel"/> <input type="button" value="OK"/>	

Group box [^]

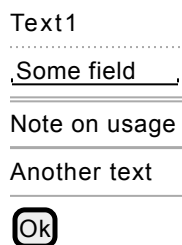
```
@startsalt
{^"My group box"
  Login      | "MyName  "
  Password   | "****    "
  [Cancel]   | [ OK    ]
}
@endsalt
```

A visual representation of the group box dialog. It shows a rectangular box with a title bar "My group box". Inside, there are two input fields: "Login" with the text "MyName" and "Password" with the text "****". Below the input fields are two buttons: "Cancel" and "OK".

Using separator [.. , == , ~~ , --]

You can use several horizontal lines as separator.

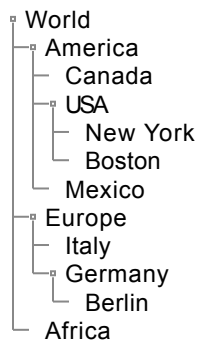
```
@startsalt
{
  Text1
  ..
  "Some field"
  ==
  Note on usage
  ~~
  Another text
  --
  [Ok]
}
@endsalt
```

A visual representation of the separator dialog. It shows a rectangular box with a title bar "Text1". Inside, there are four sections separated by horizontal lines: "Some field", "Note on usage", and "Another text". At the bottom is a button labeled "Ok".

Tree widget [T]

To have a Tree, you have to start with `{T` and to use `+` to denote hierarchy.

```
@startsalt
{
  {T
    + World
    ++ America
    +++ Canada
    +++ USA
    ++++ New York
    ++++ Boston
    +++ Mexico
    ++ Europe
    +++ Italy
    +++ Germany
    ++++ Berlin
    ++ Africa
  }
}
@endsalt
```



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

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 _____

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object [Browse...](#)

Adding tabs [/]

You can add tabs using `{/` notation. Note that you can use HTML code to have bold text.

```

@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

```

General	<input type="checkbox"/> Fullscreen	<input type="checkbox"/> Behavior	<input type="checkbox"/> Saving
Open image in: <input type="text" value="Smart Mode"/>			
<input checked="" type="checkbox"/> Smooth images when zoomed			
<input checked="" type="checkbox"/> Confirm image deletion			
<input type="checkbox"/> Show hidden images			
<input type="button" value="Close"/>			

Tab could also be vertically oriented:

```

@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

```

General	Open image in: Smart Mode ▼
	<input checked="" type="checkbox"/> Smooth images when zoomed
Fullscreen	<input checked="" type="checkbox"/> Confirm image deletion
	<input type="checkbox"/> Show hidden images
Behavior	
	Close
Saving	

Using menu [*]

You can add a menu by using {*} notation.

```
@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
```

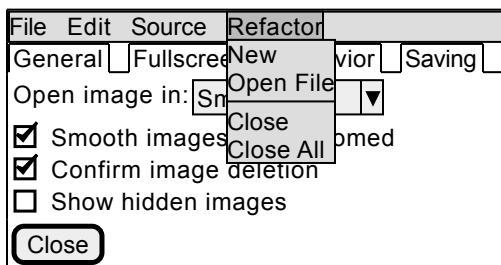
File	Edit	Source	Refactor
General	Fullscreen	Behavior	Saving
Open image in: Smart Mode ▼			
<input checked="" type="checkbox"/> Smooth images when zoomed			
<input checked="" type="checkbox"/> Confirm image deletion			
<input type="checkbox"/> Show hidden images			
Close			

It is also possible to open a menu:

```

@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

```

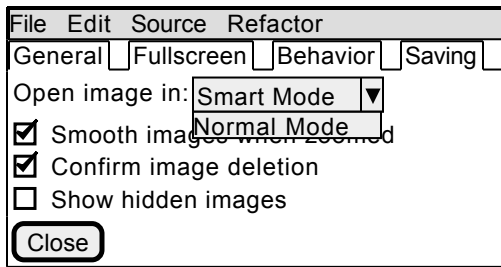


Like it is possible to open a droplist:

```

@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^^Normal Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt

```



Advanced table

You can use two special notations for table :

- `*` to indicate that a cell with span with left
- `.` to denote an empty cell

```
@startsalt
{#
. | Column 2 | Column 3
Row header 1 | value 1 | value 2
Row header 2 | A long cell | *
}
@endsalt
```

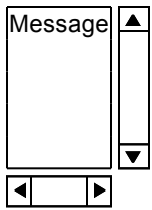
	Column 2	Column 3
Row header 1	value 1	value 2
Row header 2	A long cell	

Scroll Bars [S, SI, S-]

You can use `{S}` notation for [scroll bar](#) like in following examples:

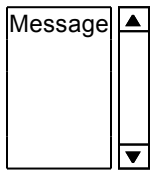
- `{S}` : for horizontal and vertical scrollbars

```
@startsalt
{S
Message
.
.
.
.
}
@endsalt
```



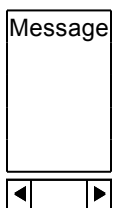
- `{SI}` : for vertical scrollbar only

```
@startsalt
{SI
Message
.
.
.
.
}
@endsalt
```



- `{S-}` : for horizontal scrollbar only

```
@startsalt
{S-
Message
.
.
.
.
}
@endsalt
```



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
```

Just plain text

This is my default button

This is my green button

This is my disabled button

☐ Unchecked box

☒ Checked box

Enter text here

This is a droplist

This is a disabled droplist

This is a red droplist

Creole on Salt

You can use [Creole](#) or [HTML Creole](#) on salt:

```

@startsalt
{{^==Creole
  This is bold
  This is //italics//
  This is ""monospaced""
  This is stricken-out
  This is underlined
  This is wave-underlined
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
}|
{^<b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
  -- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
}|
{^Creole line
You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}|
{^Creole list item

```



```

**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----

**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}|
{^Mix on salt
  ==<color:Blue>Just plain text
  [This is my default button]
  [<b><color:green>This is my green button]
  [ ---<color:#9a9a9a>This is my disabled button-- ]
  [] <size:20><color:red>Unchecked box
  [X] <color:green>Checked box
  "//Enter text here//  "
  ^This is a droplist^
  ^<color:#9a9a9a>This is a disabled droplist^
  ^<b><color:red>This is a red droplist^
}}
@endsalt

```

Creole

This is **bold**
 This is *italics*
 This is monospaced
 This is ~~stricken-out~~
 This is underlined
 This is wave-underlined

 This is ∞ long
 This is a ☹ icon
 Use image : (Cannot decode: http://plantuml.com/logo3.png)

HTML Creole

This is **bold**
 This is *italics*
 This is monospaced
 This is ~~stroked~~
 This is underlined
 This is waved
 This is ~~stroked~~
 This is underlined
 This is waved

 This is Blue
 This is Orange background
 This is big

Creole line

You can have horizontal

 Or double line

 Or strong line

 Or dotted line

 Or dotted title
and title...

 Or double-line title

 Or single-line title
 Enjoy!

Pseudo sprite [<<, >>]

Using << and >> you can define a pseudo-sprite or sprite-like drawing and reusing it latter.

```

@startsalt
{
  [X] checkbox | [] checkbox
  () radio | (X) radio
  This is a text | [This is my button] | This is another text
  "A field" | "Another long Field" | [A button]
  <<folder
  .....
  .XXXXX.....
  .X...X.....
  .XXXXXXXXXXX.
  .X.....X.
  .X.....X.
  .X.....X.
  .X.....X.
  .XXXXXXXXXXX.
  .....
  >> | <color:blue>other folder | <<folder>>
^Droplist^
}
@endsalt

```

☒ checkbox ☐ checkbox
☐ radio ☒ radio
 This is a text This is another text

OpenIconic

[OpenIconic](#) is a 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

```

Login 

Password 




























































































The complete list is available on [OpenIconic Website](https://openiconic.com), or you can use the following special diagram:

```
@startuml
listopeniconic
@enduml
```

List Open Iconic

Credit to

<https://useiconic.com/open>

↩ account-login	 bell	 cloud	≡ excerpt	≡ justify-right	 key
↩ account-logout	 bluetooth	 cloudy	⌵ expand-down	 key	 laptop
↶ action-redo	B bold	 code	⌵ expand-left	 layers	 lightbulb
↷ action-undo	 bolt	 cog	⌵ expand-right	 link-broken	 link-intact
≡ align-center	 book	⌵ collapse-down	⌵ expand-up	 list-rich	 list
≡ align-left	 bookmark	⌵ collapse-left	⌵ external-link	 location	 lock-locked
≡ align-right	 box	⌵ collapse-right	 eye	 lock-unlocked	 loop-circular
⊗ aperture	 briefcase	⌵ collapse-up	 eyedropper	 loop-square	 loop
↓ arrow-bottom	£ british-pound	% command	 file	 magnifying-glass	 map-marker
⌚ arrow-circle-bottom	 browser	 comment-square	 fire	 map	 media-pause
⌚ arrow-circle-left	 brush	 compass	 flag	 media-play	 media-record
⌚ arrow-circle-right	⚙ aperture	⊕ contrast	 flash	 media-record	 media-skip-backward
⌚ arrow-circle-top	 bullhorn	≡ copywriting	 folder	 media-skip-forward	 media-step-backward
← arrow-left	 calculator	 credit-card	 fork	 media-step-forward	 media-stop
→ arrow-right	 calendar	 crop	 fullscreen-enter	 medical-cross	 menu
↓ arrow-thick-bottom	 camera-slr	⌚ dashboard	 fullscreen-exit	 microphone	 minus
← arrow-thick-left	▼ caret-bottom	↓ data-transfer-download	 globe	 monitor	 moon
→ arrow-thick-right	◀ caret-left	↑ data-transfer-upload	 graph	 move	
↑ arrow-thick-top	▶ caret-right	 delete	 grid-four-up		
↑ arrow-top	▲ caret-top	 dial	 grid-three-up		
🔊 audio-spectrum	🗑 cart	 document	 grid-two-up		
🔊 audio	 chat	\$ dollar	 hard-drive		
🔔 badge	✓ check	” double-quote-sans-left	 header		
🚫 ban	▼ chevron-bottom	“ double-quote-sans-right	 headphones		
📊 bar-chart	◀ chevron-left	“ double-quote-serif-left	 heart		
🛒 basket	▶ chevron-right	” double-quote-serif-right	 home		
🔋 battery-empty	^ chevron-top	 droplet	 image		
🔋 battery-full	⊙ circle-check	 eject	 inbox		
🧴 beaker	⊗ circle-x	 elevator	 infinity		
	 clipboard	… ellipses	 info		
	⌚ clock	 envelope-closed	<i>I</i> italic		
	 cloud-download	 envelope-open	≡ justify-center		
	 cloud-upload	€ euro	≡ justify-left		

Add title, header, footer, caption or legend

```
@startsalt
title My title
header some header
footer some footer
caption This is caption
legend
The legend
end legend
```

```
{+
  Login      | "MyName    "
  Password   | "****      "
  [Cancel]   | [ OK      ]
}
```

```
@endsalt
```

some header

My title

Login	MyName
Password	****
Cancel	OK

The legend

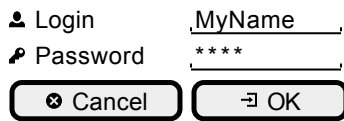
This is caption

some footer

Zoom, DPI

Whitout zoom (by default)

```
@startsalt
{
  <&person> Login      | "MyName    "
  <&key> Password   | "****      "
  [ <&circle-x> Cancel ] | [ <&account-login> OK      ]
}
@endsalt
```



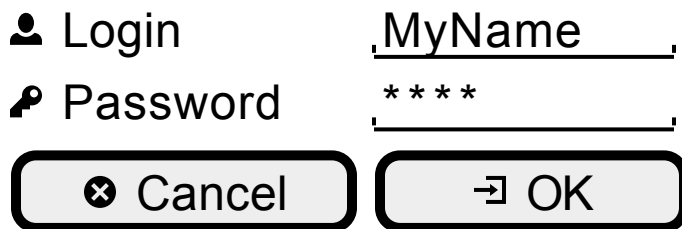
Person icon Login MyName
Key icon Password ****
Cancel OK

Scale

You can use the `scale` command to zoom the generated image.

You can use either a number or a fraction to define the scale factor. You can also specify either width or height (in pixel). And you can also give both width and height: the image is scaled to fit inside the specified dimension.

```
@startsalt
scale 2
{
  <&person> Login | "MyName  "
  <&key> Password | "****  "
  [ <&circle-x> Cancel ] | [ <&account-login> OK ]
}
@endsalt
```







Person icon Login MyName
Key icon Password ****
Cancel OK

DPI

You can also use the `skinparam dpi` command to zoom the generated image.

```
@startsalt
skinparam dpi 200
{
  <&person> Login | "MyName  "
  <&key> Password | "****  "
  [ <&circle-x> Cancel ] | [ <&account-login> OK ]
}
@endsalt
```

 Login	<u>MyName</u>
 Password	<u>****</u>
 Cancel	 OK

Include Salt "on activity diagram"

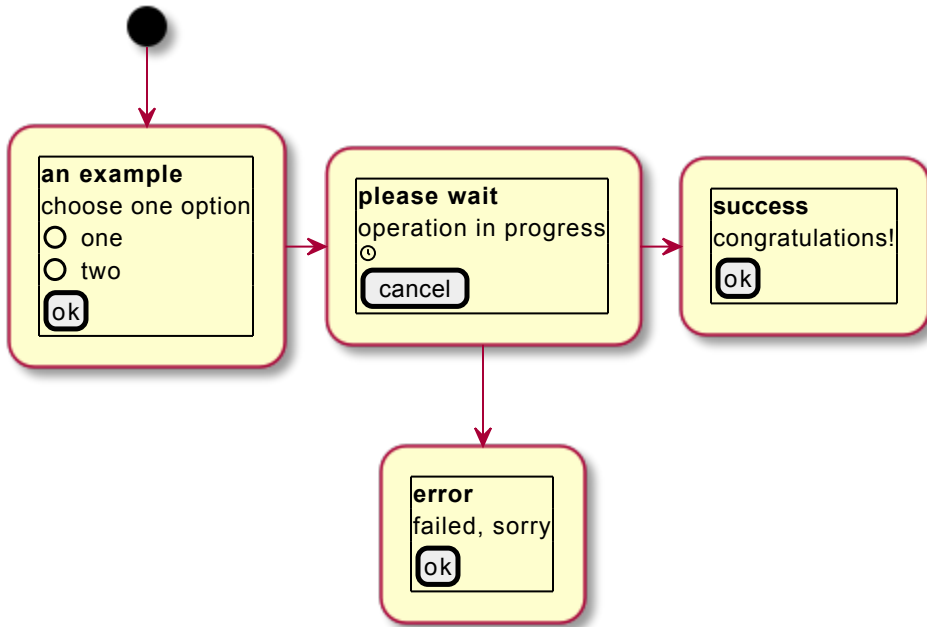
You can read the [following explanation](#).

```
@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

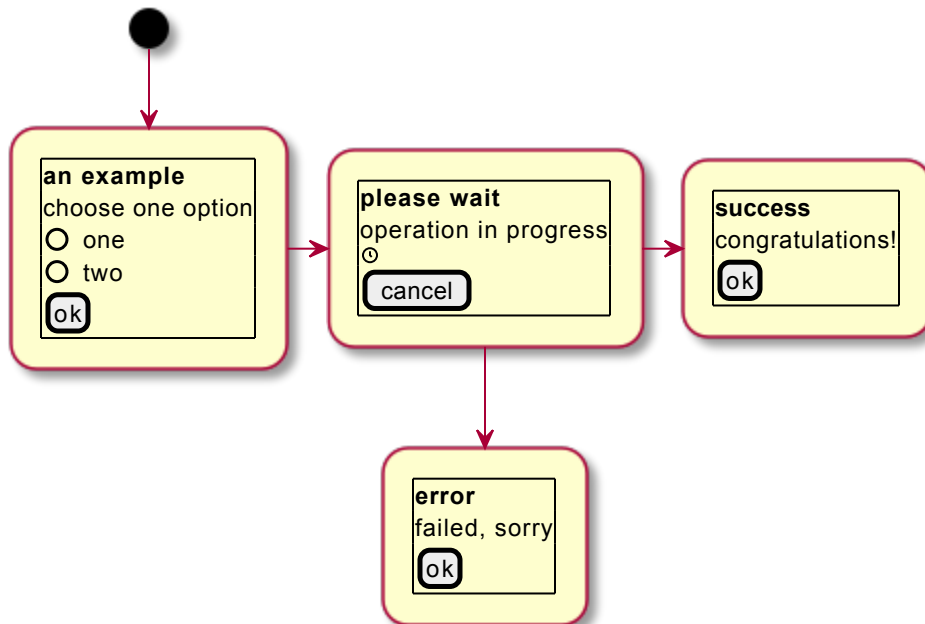
!procedure _success()
{+
<b>success
congratulations!
[ok]
}
!endprocedure

!procedure _error()
{+
<b>error
failed, sorry
[ok]
}
!endprocedure

(*) --> SALT(choose)
-right-> SALT(wait)
wait -right-> SALT(success)

```

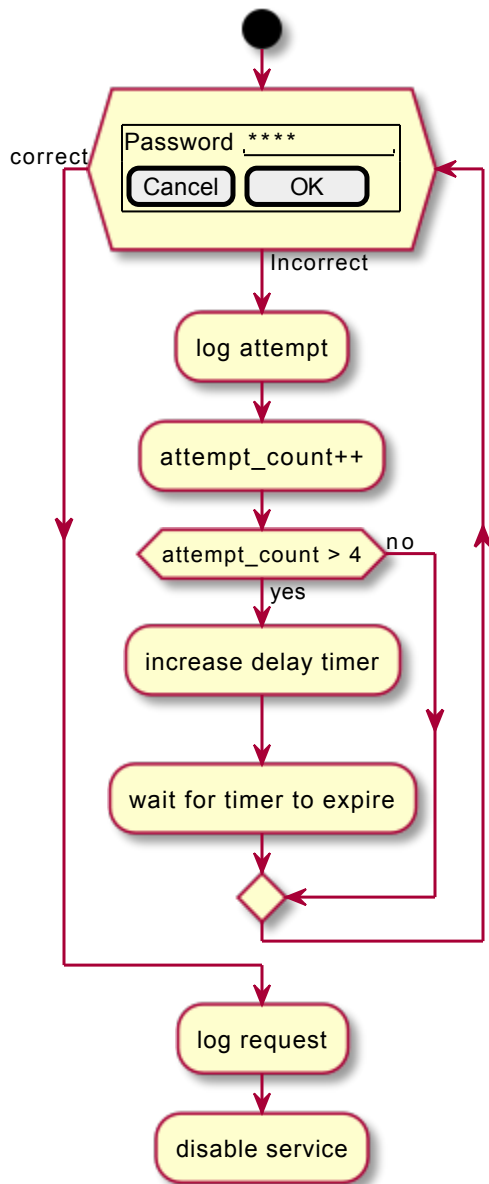
```
wait -down-> SALT(error)
@enduml
```



Include salt "on while condition of activity diagram"

You can include `salt` on while condition of activity diagram.

```
@startuml
start
while (\n{\nsalt\n{+\nPassword | "****          "\n[Cancel] | [ OK  ]}\n}) 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
```



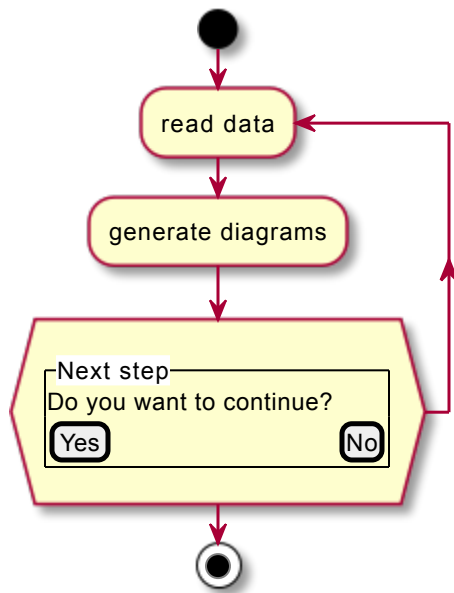
Include salt "on repeat while condition of activity diagram"

You can include `salt` on 'repeat while' condition of activity diagram.

```

@startuml
start
repeat :read data;
  :generate diagrams;
repeat while (\n{\nsalt\n{"Next step"\n Do you want to continue? \n[Yes]|[No]\n}\n})
stop
@enduml

```



Skinparam

You can use **[only]** some [skinparam](#) command to change the skin of the drawing.

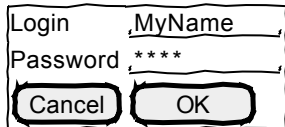
Some example:

```

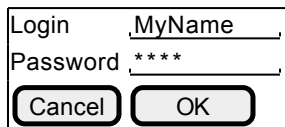
@startsalt
skinparam Backgroundcolor palegreen
{+
  Login      | "MyName  "
  Password   | "****    "
  [Cancel]   | [ OK    ]
}
@endsalt
  
```

A visual representation of the skinparam example. It shows a green rectangular box containing a login form. The form has two input fields: 'Login' with the text 'MyName' and 'Password' with the text '****'. Below the input fields are two buttons: 'Cancel' and 'OK'.

```
@startsalt
skinparam handwritten true
{+
    Login      | "MyName  "
    Password   | "****    "
    [Cancel]   | [ OK   ]
}
@endsalt
```



```
@startsalt
skinparam defaultFontName monospaced
{+
    Login      | "MyName  "
    Password   | "****    "
    [Cancel]   | [ OK   ]
}
@endsalt
```



Style

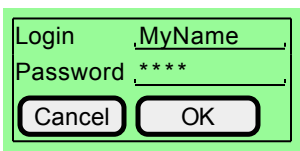
You can use **[only]** some [style](#) command to change the skin of the drawing.

Some example:

```

@startsalt
<style>
saltDiagram {
  BackgroundColor palegreen
}
</style>
{+
  Login      | "MyName  "
  Password   | "****   "
  [Cancel]   | [ OK   ]
}
@endsalt

```



```

@startsalt
<style>
saltDiagram {
  Fontname Monospaced
  FontSize 10
  FontStyle italic
  LineThickness 0.5
  LineColor red
}
</style>
{+
  Login      | "MyName  "
  Password   | "****   "
  [Cancel]   | [ OK   ]
}
@endsalt

```

