

ppp Documentation

Daniel Bruder

Version 0.9.0

1 Abstract

ppp allows you to use pandoc in new ways by rendering ASCII-markup to beautiful pictures right from within pandoc's verbatim environments.

See below for illustrative examples and detailed usage instructions.

Bonus on top: Leaving out **ppp** from the typesetting pipeline will still render your document and the verbatims with the ASCII-markup will still stay readable!

Contents

1	Abstract	2
2	General usage	5
2.1	General Renderers	5
2.2	General Options	5
3	ditaa Diagrams	6
3.1	ditaa Options	6
3.2	ditaa Examples	6
4	dot Diagrams	8
4.1	dot Options	8
4.2	dot Examples	8
5	neato Diagrams	9
5.1	neato Options	9
5.2	neato Examples	9
6	yUML	10
6.1	yUML Options	10
6.2	yUML Examples	10
6.2.1	yUML Class diagrams	10
6.2.2	yuml Usecase diagrams	11
6.2.3	yuml Activity diagrams	12
7	plantuml	13
7.1	plantuml Options	13
7.2	plantuml Examples	13
7.2.1	plantuml Example 1	13
7.2.2	plantuml Example 2	15

8	rdfdot Diagrams	18
8.1	rdfdot Options	18
8.2	rdfdot Examples	18
9	List of options	20
10	List of homepages and documentation to renderers	21
11	Credits and further references	21

2 General usage

In each case, you will use pandoc's verbatim environment, set the rendering engine and additional options:

```
~~~~~ {.renderer .option1 .option2=value2}  
--- RENDERER-SPECIFIC MARKUP GOES HERE ---  
~~~~~
```

2.1 General Renderers

The renderers available to `ppp` are:

- ditaa
- yuml diagrams:
 - class diagrams (cf. Figure 6)
 - usecase diagrams (cf. Figure 7)
 - activity diagrams (cf. Figure 8)
- dot
- neato
- rfdot
- plantuml

2.2 General Options

This is a list of the general options, compatible with any type of renderer:

- `.scale=90%`
 - `.label=fig:my-figure`
 - `.title="Some label for the figure"`
-

3 ditaa Diagrams

In order to generate ditaa-diagrams, ditaa needs to be installed.

For an exhaustive list of options and possibilities, please check the [ditaa home-page](#).

3.1 ditaa Options

Apart from the [General Options](#), the possible options specific to ditaa are:

- .rounded-corners
- .no-shadows
- .no-antialias
- .no-separation

3.2 ditaa Examples

Using ditaa, the following markup will produce Figure ??.

```
~~~~~ {.ditaa .rounded-corners .no-shadows .scale=90%
      .title="The ppp and pandoc pipeline"
      .label=fig:pipeline-overview
      .no-antialias .no-separation}
# Caution! These lines actually would have to be on *one* line only!
+-----+ +-----+ +-----+ +-----+
| markdown source |----->| ppp |-----*-->| processed markdown |
+-----+ +-----+ | +-----+
|                                     | \--->| image files |
|                                     | +-----+
|                                     |
| diagram creation |
+-----+
| ditaa/dot/rdfdot |
+-----+
~~~~~
```

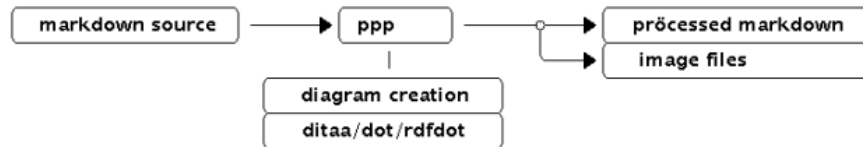


Figure 1: “The ppp and pandoc pipeline”

As a contrast, turning off several options, ditaa will produce an output as in Figure 3:

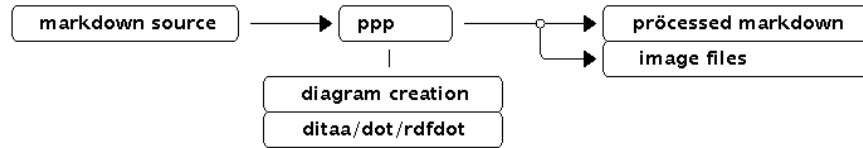


Figure 2: “The ppp and pandoc pipeline”

```

~~~~~ {.ditaa .scale=90% .title="The ppp and pandoc pipeline #2" .label=fig:pipeline-overview-2}
+-----+ +-----+ +-----+
| markdown source |----->| ppp |-----*-->| processed markdown |
+-----+ +-----+ | +-----+
| | | | |
| | | | |
| diagram creation | | |
+-----+ +-----+
| ditaa/dot/rdfdot | | |
+-----+ +-----+
~~~~~

```

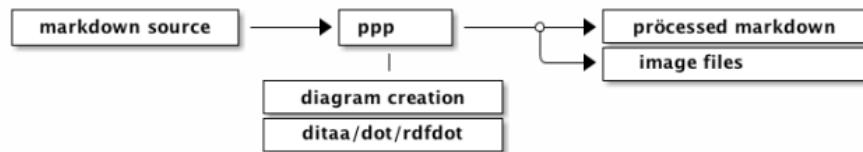


Figure 3: “The ppp and pandoc pipeline #2”

4 dot Diagrams

dot rendering is done through [GraphViz's](#) engine. Please cf. [Graphviz's Documentation](#) for exact usage specifics on the usage of dot.

4.1 dot Options

- currently none apart from the [General Options](#)

4.2 dot Examples

With dot as the *renderer*, the following markup produces the figure as seen in [Figure 4](#).

```
~~~~~ {.dot .scale=50% .title=dot Finite State Automaton .label=fig:dot-fsa}
digraph finite_state_machine {
    rankdir=LR;
    node [shape = doublecircle]; LR_0 LR_3 LR_4 LR_8;
    node [shape = circle];
    LR_0 -> LR_2 [ label = "SS(B)" ];
    LR_0 -> LR_1 [ label = "SS(S)" ];
    LR_1 -> LR_3 [ label = "S($end)" ];
    LR_2 -> LR_6 [ label = "SS(b)" ];
    LR_2 -> LR_5 [ label = "SS(a)" ];
    LR_2 -> LR_4 [ label = "S(A)" ];
    LR_5 -> LR_7 [ label = "S(b)" ];
    LR_5 -> LR_5 [ label = "S(a)" ];
    LR_6 -> LR_6 [ label = "S(b)" ];
    LR_6 -> LR_5 [ label = "S(a)" ];
    LR_7 -> LR_8 [ label = "S(b)" ];
    LR_7 -> LR_5 [ label = "S(a)" ];
    LR_8 -> LR_6 [ label = "S(b)" ];
    LR_8 -> LR_5 [ label = "S(a)" ];
}
~~~~~
```

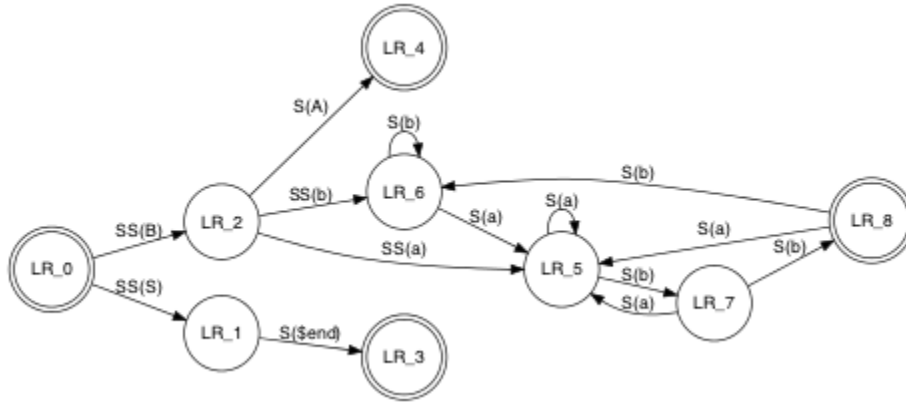


Figure 4: dot Finite State Automaton

5 neato Diagrams

neato diagrams behave very similarly to [dot Diagrams](#). Please cf [dot Diagrams](#) for more information

5.1 neato Options

- same as [dot Options](#)

5.2 neato Examples

The following example produces Figure 5.

```
~~~~~ {.neato .scale=80% .title=neato diagram .label=fig:neato-diagram}
graph G {
  n0 -- n1 -- n2 -- n3 -- n0;
}
~~~~~
```

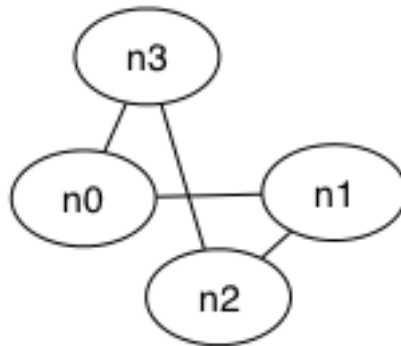


Figure 5: neato diagram

6 yUML

yUML needs a network connection and uses <http://yuml.me> as the rendering service.

6.1 yUML Options

Options specific to yUML can be:

- `.type=:` any of [class, activity, usecase]
- `.style=:` any of [scruffy, nofunky, plain]
- `.direction=:` any of [LR, RL, TD,]

6.2 yUML Examples

6.2.1 yUML Class diagrams

With *yUML* as the renderer, setting `.type=class` and using the style `.style=nofunky`, the following markup produces Figure 6.

```

~~~~ { .yuml .style=nofunky .type=class .direction=LR .scale=100% .title=yUML class diagram .label=fig:yuml-class-diagram }
[Customer] +1 -> * [Order]
[Order] ++1 -items> * [LineItem]
[Order] -0..1> [PaymentMethod]
~~~~

```

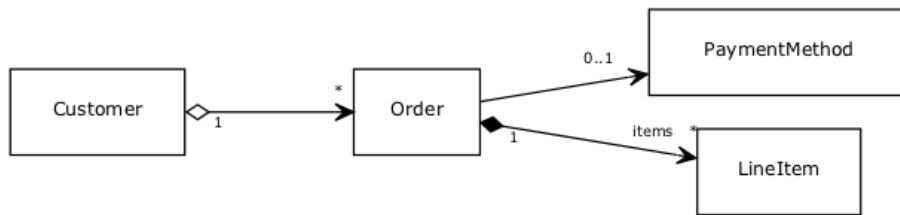


Figure 6: yUML class diagram

6.2.2 yuml Usecase diagrams

With **scruffy** style and **.type=usecase**, the following example produces Figure 7.

```

~~~~ {yuml .style=scruffy .type=usecase .title=yUML usecase diagram .label=fig:yuml-usecase-diagram}
// Cool Use Case Diagram
[Customer]-(Make Cup of Tea)
(Make Cup of Tea)<(Add Milk)
(Make Cup of Tea)>(Add Tea Bag)
~~~~

```

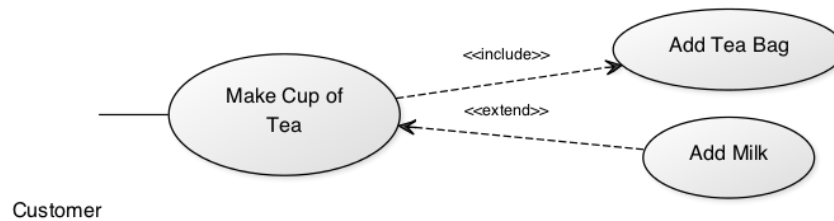


Figure 7: yUML usecase diagram

6.2.3 yuml Activity diagrams

Lastly, using `.type=activity` and `.style=plain` the following example produces Figure 8.

```
~~~~ {.yuml .style=plain .type=activity .title=yUML activity Diagram .label=fig:yuml-activity-diagram}
(start)->|a|,|a|->(Make Coffee)->|b|,|a|->(Make Breakfast)->|b|,|b|-><c>[want more coffee]->(Make Coffee),<c>[satisfied]->(end)
~~~~
```

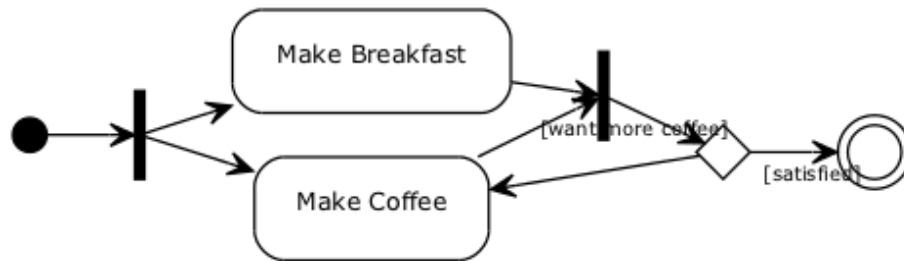


Figure 8: yUML activity Diagram

7 plantuml

plantuml – based on graphviz –, has an extensive feature set

7.1 plantuml Options

- *General Options*

7.2 plantuml Examples

7.2.1 plantuml Example 1

With *plantuml* as the renderer, the following markup produces Figure 9.

```
~~~~ {plantuml .scale=60% .title=PlantUML Example 1 .label=fig:plantuml-example-1}
@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
~~~~
```

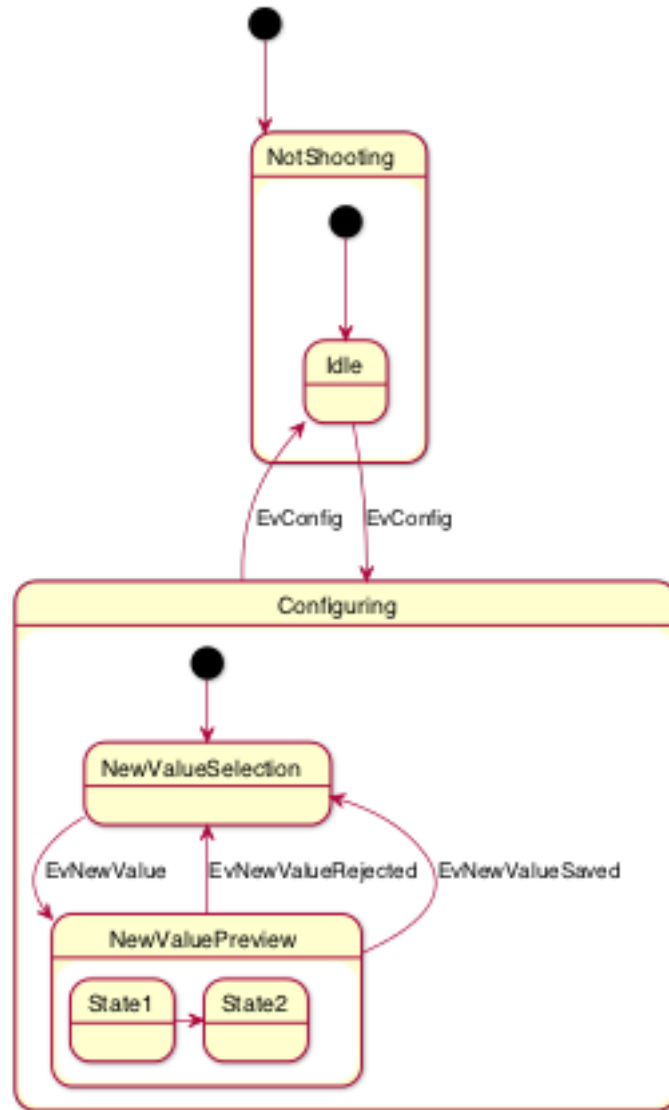


Figure 9: PlantUML Example 1

7.2.2 plantuml Example 2

If the colors don't match your taste exactly, add `skinparam monochrome true` to retrieve Figure 10.

```
~~~~~ {.plantuml .scale=120% .title=PlantUML Example 2 .label=fig:plantuml-example-2}
@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
~~~~~
```

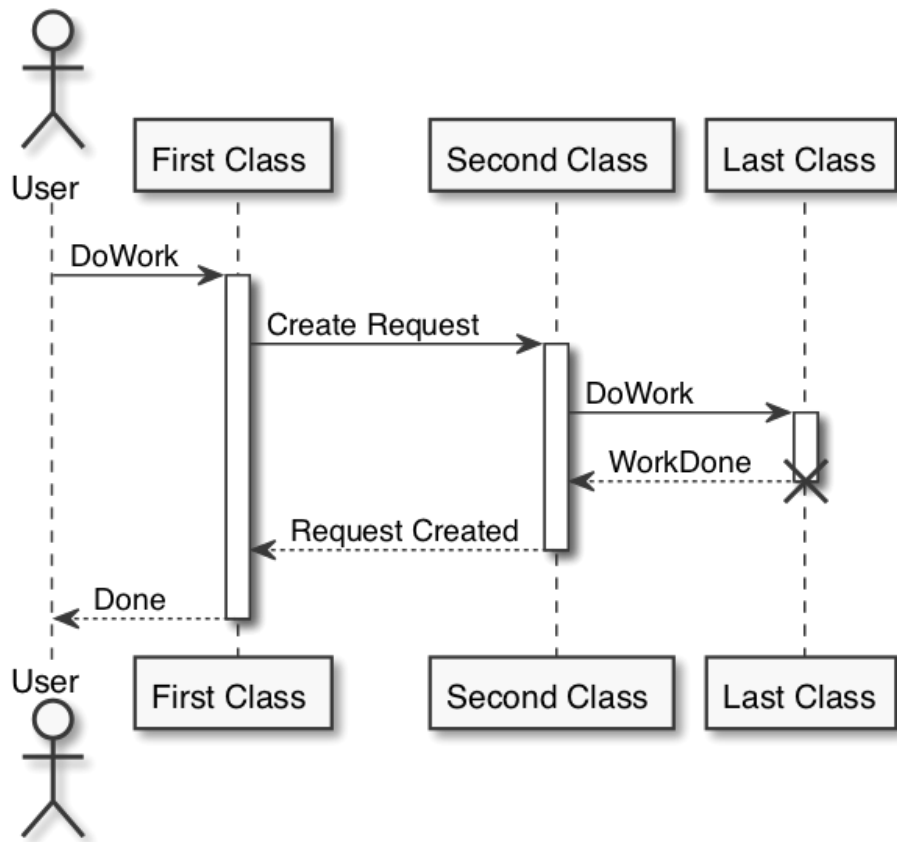


Figure 10: PlantUML Example 2

8 rdfdot Diagrams

8.1 rdfdot Options

- currently none apart from the [General Options](#)

8.2 rdfdot Examples

The following example produces [Figure 11](#) on [page 19](#).

```
~~~~~ {.rdfdot .scale=150% .title=rdfdot Diagram .label="fig:rdfdot-diagram"}
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@base <http://example.com/> .
<alice> foaf:name "Alice" ;
        foaf:knows [ foaf:name "Bob" ] .
~~~~~
```

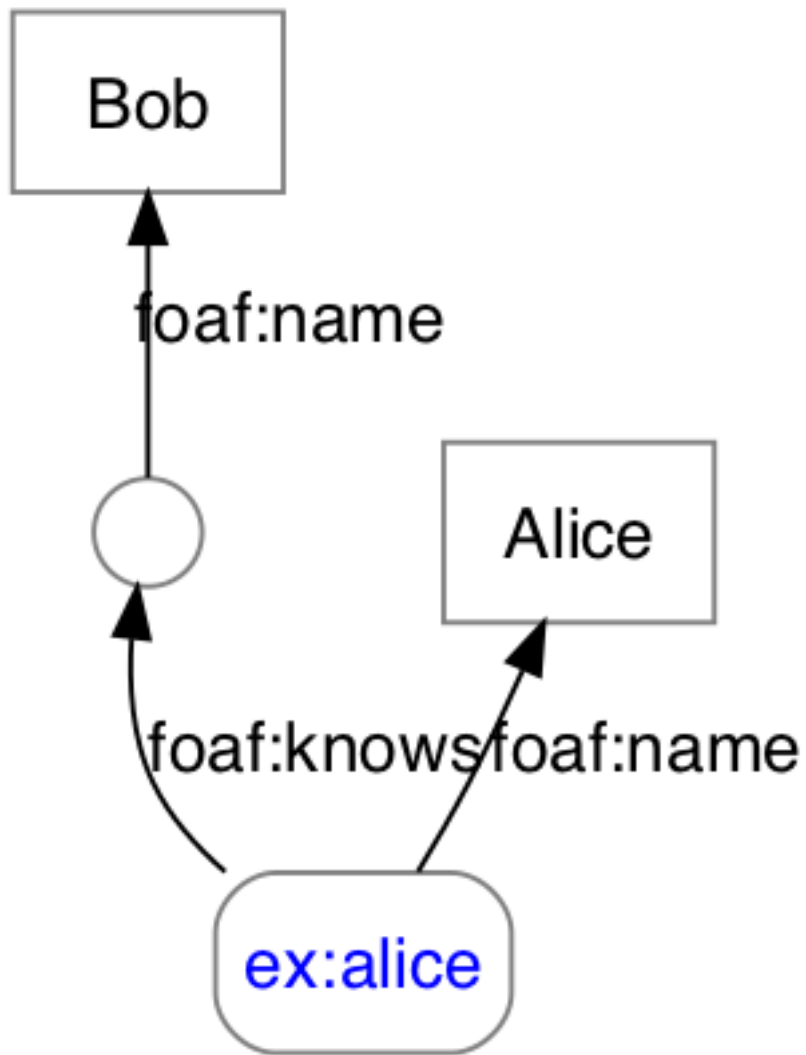


Figure 11: rdfdot Diagram

9 List of options

Renderer	Option	possible values
<i>General</i>	<code>.scale</code>	e.g. 120%
	<code>.label</code>	<code>fig:my-figure</code>
	<code>.title</code>	"Some label for the figure"
ditaa	<code>.rounded-corners</code>	
	<code>.no-shadows</code>	
	<code>.no-antialias</code>	
	<code>.no-separation</code>	
dot	N/A	
neato	N/A	
yUML	<code>.type=</code>	any of [<code>class</code> , <code>activity</code> , <code>usecase</code>]
	<code>.style=</code>	any of [<code>scruffy</code> , <code>nofunky</code> , <code>plain</code>]
	<code>.direction=</code>	any of [<code>LR</code> , <code>RL</code> , <code>TD</code> ,]
rdfdot	N/A	

Table 1: List of options

10 List of homepages and documentation to renderers

Renderer	Links
ppp	(this document) https://metacpan.org/release/App-pandoc-preprocess https://github.com/xdbr/p5-App-pandoc-preprocess
ditaa	http://ditaa.sourceforge.net/
dot	http://www.graphviz.org/
neato	http://www.graphviz.org/
yUML	http://yuml.me/ https://github.com/wandernauta/yuml
rdfdot	https://metacpan.org/pod/RDF::Trine::Exporter::GraphViz
plantuml	http://plantuml.sourceforge.net/

Table 2: List of options

11 Credits and further references

- <http://www.asciiflow.com/#Draw>: an excellent helper for all things diagram
- <http://randomdeterminism.wordpress.com/2012/06/01/how-i-stopped-worrying-and-started-using-markdown/>: general introduction to another approach to typesetting and using `gpp`
- <https://github.com/nichtich/ditaa-markdown>: This is where the original idea came from
- `gpp`: <http://files.nothingisreal.com/software/gpp/gpp.html>

List of Figures

1	“The ppp and pandoc pipeline”	6
2	“The ppp and pandoc pipeline”	7
3	“The ppp and pandoc pipeline #2”	7
4	dot Finite State Automaton	9
5	neato diagram	10

6	yUML class diagram	11
7	yUML usecase diagram	11
8	yUML activity Diagram	12
9	PlantUML Example 1	14
10	PlantUML Example 2	16
11	rdfdot Diagram	19
