



Graphviz Flowcharts & Graphs

Vendor	MindTouch
Type	Native
Categories	Graphs
Requires	MindTouch 1.8.2 or later
OS Restriction	None
Status	Stable
License	Free/Open Source
SID (service id)	sid://mindtouch.com/2007/06/graphviz
Assembly	mindtouch.deki.services

Install Extension

URL of your MindTouch install (ex: <http://www.mindtouch.com>)

Configure Extension

Description

This extension contains functions for generating graphs. See attached files for documentation on the graph notation.

See also [How to add an extension](#), [Using the Extension Dialog](#), [Learn about DekiScript](#), [Extensions Directory](#).

Configuration:

Before the Graphviz service can be used, it must be configured with the locations of the required Graphviz applications. Graphviz applications can be [downloaded here](#). VM users with SSH access can simply type `apt-get install graphviz` (paths are all prefixed with `/usr/bin/`: e.g., `/usr/bin/dot`, `/usr/bin/neato`, etc.).

Installing on CentOS

Add the following repo to your CentOS repos: <http://www.graphviz.org/graphviz-rhel.repo>

Then run: `yum install graphviz`

Config Key	Description
dot-path	Path to <i>dot</i> application.
neato-path	Path to <i>neato</i> application.
twopi-path	Path to <i>twopi</i> application
circo-path	Path to <i>circo</i> application.

Functions:

1. `graphviz.circo`
 2. `graphviz.dot`
 3. `graphviz.neato`
 4. `graphviz.twopi`
-

graphviz.circo(graph : str) : uri

Generate graph using CIRCO layout.

Parameters:

Name	Type	Description
graph	str	Graph description.
format	str	Optional. Output format (8.08 or later) One of "png", "png+map", "svg", or "xdot". Default is "png". - "png" produces a PNG image. The return type is URI. - "png+map" produces a PNG image and a <code><usemap></code> element describing the clickable areas. The return type is XML. - "svg" produces an SVG image and the corresponding <code><embed></code> element to visualize it. The return type is XML. - "xdot" produces an annotated text representation of the input. The return type is STR.

Samples:

Output

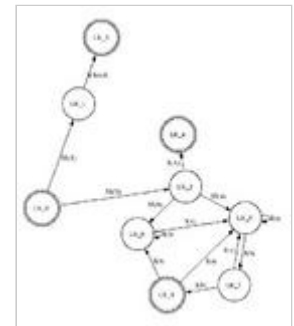
To generate a simple "Hello World" graph using CIRCO layout:

```
{{ graphviz.circo("digraph G { Hello -> World }") }}
```

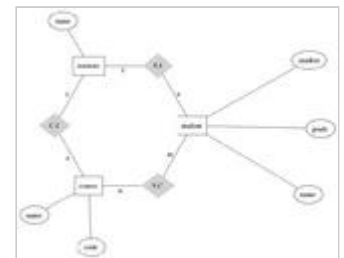


To generate a state-transition diagram using CIRCO layout:

```
{{graphviz.circo("digraph finite_state_machine {  
  rankdir=LR;  
  size=\"8,5\"  
  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)\"];  
}"} }}
```



To an entity-relation diagram using CIRCO layout:



```

{{graphviz.circo("graph ER {
    node [shape=box]; course; institute; student;
    node [shape=ellipse]; {node [label=\"name\"] name0;
name1; name2;}
        code; grade; number;
    node [shape=diamond,style=filled,color=lightgrey];
\"C-I\"; \"S-C\"; \"S-I\";
    name0 -- course;
    code -- course;
    course -- \"C-I\" [label=\"n\",len=1.00];
    \"C-I\" -- institute [label=\"1\",len=1.00];
    institute -- name1;
    institute -- \"S-I\" [label=\"1\",len=1.00];
    \"S-I\" -- student [label=\"n\",len=1.00];
    student -- grade;
    student -- name2;
    student -- number;
    student -- \"S-C\" [label=\"m\",len=1.00];
    \"S-C\" -- course [label=\"n\",len=1.00];
}}) }}

```

graphviz.dot(graph : str) : uri

Generate graph using DOT layout.

Parameters:

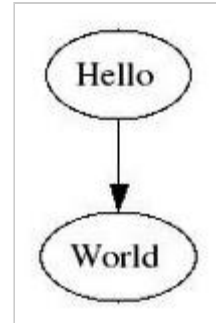
Name	Type	Description
graph	str	Graph description.
format	str	Optional. Output format (8.08 or later) One of "png", "png+map", "svg", or "xdot". Default is "png". - "png" produces a PNG image. The return type is URI. - "png+map" produces a PNG image and a <usemap> element describing the clickable areas. The return type is XML. - "svg" produces an SVG image and the corresponding <embed> element to visualize it. The return type is XML. - "xdot" produces an annotated text representation of the input. The return type is STR.

Samples:

	Output

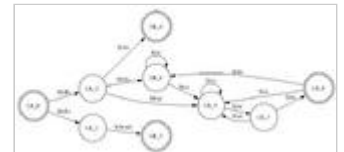
To generate a simple "Hello World" graph using DOT layout:

```
{{ graphviz.dot("digraph G { Hello -> World }") }}
```

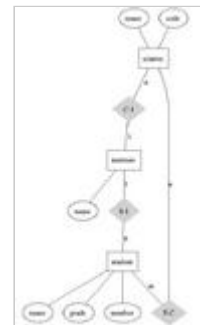


To generate a state-transition diagram using DOT layout:

```
{{graphviz.dot("digraph finite_state_machine {
rankdir=LR;
size=\"8,5\"
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)\" ];
}"} }}
```



To an entity-relation diagram using DOT layout:



```

{{graphviz.dot("graph ER {
    node [shape=box]; course; institute; student;
    node [shape=ellipse]; {node [label=\"name\"] name0;
name1; name2;}
        code; grade; number;
    node [shape=diamond,style=filled,color=lightgrey];
\"C-I\"; \"S-C\"; \"S-I\";
    name0 -- course;
    code -- course;
    course -- \"C-I\" [label=\"n\",len=1.00];
    \"C-I\" -- institute [label=\"1\",len=1.00];
    institute -- name1;
    institute -- \"S-I\" [label=\"1\",len=1.00];
    \"S-I\" -- student [label=\"n\",len=1.00];
    student -- grade;
    student -- name2;
    student -- number;
    student -- \"S-C\" [label=\"m\",len=1.00];
    \"S-C\" -- course [label=\"n\",len=1.00];
}) }}

```

graphviz.neato(graph : str) : uri

Generate graph using NEATO layout.

Parameters:

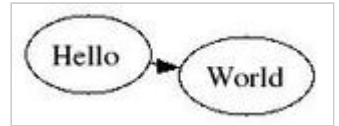
Name	Type	Description
graph	str	Graph description.
format	str	Optional. Output format (8.08 or later) One of "png", "png+map", "svg", or "xdot". Default is "png". - "png" produces a PNG image. The return type is URI. - "png+map" produces a PNG image and a <usemap> element describing the clickable areas. The return type is XML. - "svg" produces an SVG image and the corresponding <embed> element to visualize it. The return type is XML. - "xdot" produces an annotated text representation of the input. The return type is STR.

Samples:

	Output

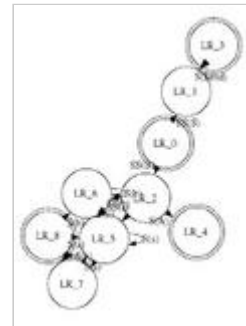
To generate a simple "Hello World" graph using NEATO layout:

```
{{ graphviz.neato("digraph G { Hello -> World }") }}
```

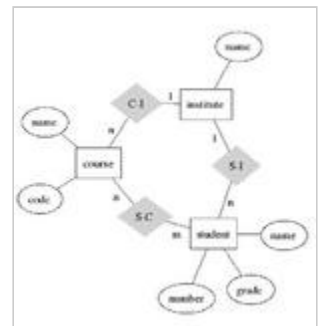


To generate a state-transition diagram using NEATO layout:

```
{{graphviz.neato("digraph finite_state_machine {
rankdir=LR;
size=\"8,5\"
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)\"];
}"} }}
```



To an entity-relation diagram using NEATO layout:



```

{{graphviz.neato("graph ER {
    node [shape=box]; course; institute; student;
    node [shape=ellipse]; {node [label=\"name\"] name0;
name1; name2;}
        code; grade; number;
    node [shape=diamond,style=filled,color=lightgrey];
\"C-I\"; \"S-C\"; \"S-I\";
    name0 -- course;
    code -- course;
    course -- \"C-I\" [label=\"n\",len=1.00];
    \"C-I\" -- institute [label=\"1\",len=1.00];
    institute -- name1;
    institute -- \"S-I\" [label=\"1\",len=1.00];
    \"S-I\" -- student [label=\"n\",len=1.00];
    student -- grade;
    student -- name2;
    student -- number;
    student -- \"S-C\" [label=\"m\",len=1.00];
    \"S-C\" -- course [label=\"n\",len=1.00];
}"} }}

```

graphviz.twopi(graph : str) : uri

Generate graph using TWOPi layout.

Parameters:

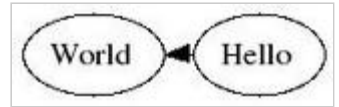
Name	Type	Description
graph	str	Graph description.
format	str	Optional. Output format (8.08 or later) One of "png", "png+map", "svg", or "xdot". Default is "png". - "png" produces a PNG image. The return type is URI. - "png+map" produces a PNG image and a <usemap> element describing the clickable areas. The return type is XML. - "svg" produces an SVG image and the corresponding <embed> element to visualize it. The return type is XML. - "xdot" produces an annotated text representation of the input. The return type is STR.

Samples:

	Output

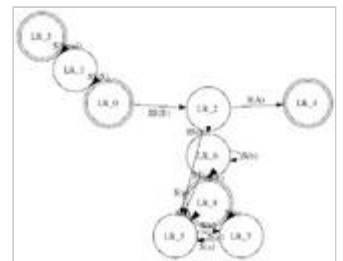
To generate a simple "Hello World" graph using TWOP layout:

```
{{ graphviz.twopi("digraph G { Hello -> World }") }}
```

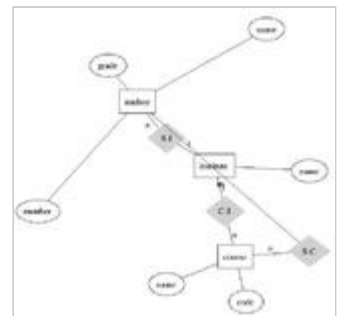


To generate a state-transition diagram using TWOP layout:

```
{{graphviz.twopi("digraph finite_state_machine {
rankdir=LR;
size=\"8,5\"
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)\" ];
}"} }}
```



To an entity-relation diagram using TWOP layout:



```

{{graphviz.twopi("graph ER {
    node [shape=box]; course; institute; student;
    node [shape=ellipse]; {node [label="name\"]; name0;
name1; name2;}
        code; grade; number;
    node [shape=diamond,style=filled,color=lightgrey];
\C-I\"; \S-C\"; \S-I\";
    name0 -- course;
    code -- course;
    course -- \C-I\" [label="n\",len=1.00];
\C-I\" -- institute [label="1\",len=1.00];
    institute -- name1;
    institute -- \S-I\" [label="1\",len=1.00];
\S-I\" -- student [label="n\",len=1.00];
    student -- grade;
    student -- name2;
    student -- number;
    student -- \S-C\" [label="m\",len=1.00];
\S-C\" -- course [label="n\",len=1.00];
}) }}

```

