
sphinxcontrib-programsscreenshot Documentation

Release 0.0.4

ponty

February 07, 2012

CONTENTS

1	About	2
2	Basic usage	3
3	How it works	4
4	Installation	5
4.1	General	5
4.2	Ubuntu	5
4.3	Uninstall	5
5	Usage	6
5.1	Configuration	6
5.2	Basic	6
5.3	shell extension	6
5.4	waiting	7
5.5	Options	7
5.6	Image options	10
6	TODO	12
7	Development	13
7.1	Tools	13
7.2	Install on ubuntu	13
7.3	Tasks	13

sphinxcontrib-programsscreenshot

Date February 07, 2012

PDF [sphinxcontrib-programsscreenshot.pdf](#)

ABOUT

This [Sphinx 1.0](#) extension executes programs during the build step and includes their screenshot into the documentation. GUI version of the [sphinxcontrib.programoutput](#) extension.

home: <https://github.com/ponty/sphinxcontrib-programscreenshot>

documentation: <http://ponty.github.com/sphinxcontrib-programscreenshot>

BASIC USAGE

```
.. program-screenshot:: xmessage hello  
    :prompt:
```

HOW IT WORKS

1. start Xvfb headless X server using `pyvirtualdisplay`
2. redirect program display to Xvfb server by setting `$DISPLAY` variable.
3. wait some seconds
4. take screenshot by `pyscreenshot` which needs `scrot`.
5. use `.. image::` directive to display image

INSTALLATION

4.1 General

- install `Xvfb` and `Xephyr`
- install `PIL`
- install `scrot`
- install `pip`
- install the program:

```
# as root
pip install sphinxcontrib-programsscreenshot
```

4.2 Ubuntu

```
sudo apt-get install python-pip
sudo apt-get install scrot
sudo apt-get install xvfb
sudo apt-get install xserver-xephyr
sudo apt-get install python-imaging
sudo pip install sphinxcontrib-programsscreenshot
```

4.3 Uninstall

```
# as root
pip uninstall sphinxcontrib-programsscreenshot
```


USAGE

5.1 Configuration

Add `sphinxcontrib.programsscreenshot` to extensions list in `conf.py`:

```
extensions = [  
    'sphinxcontrib.programsscreenshot',  
]
```

5.2 Basic

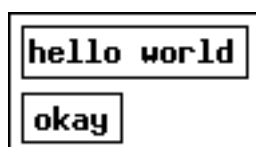
The main directive is *program-screenshot*:

```
.. directive:: program-screenshot
```

This directive accepts a single string as argument, which is the command to execute. By default, this command string is split using the `shlex` modules, which mostly works like common Unix shells like `bash` or `zsh`:

```
.. program-screenshot:: xmessage hello world
```

The above snippet would render like this:

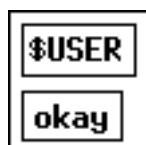


5.3 shell extension

Special shell features like parameter expansion are not supported:

```
.. program-screenshot:: xmessage "$USER"
```

The above snippet would render like this:



5.4 waiting

The program is waiting until something is displayed (test window is displayed after 3 sec):

```
.. program-screenshot:: python tests/tkmsgbox.py 3
   :prompt:
   :stdout:
   :stderr:
```

The above snippet would render like this:

```
$ python tests/tkmsgbox.py 3
[stdout] before messagebox
[stderr] before messagebox
```



If nothing happens, after timeout (:timeout:) assertion is raised.

5.5 Options

5.5.1 timeout

If nothing happens, after timeout (default 12 sec) exception is raised, you can change it with this option:

```
.. program-screenshot:: xmessage timeout
   :timeout: 120
```

5.5.2 prompt

Using the option prompt you can include the command, that produced the output:

```
.. program-screenshot:: xmessage prompt
   :prompt:
```

The above snippet would render like this:

```
$ xmessage prompt
```



5.5.3 stdout

Use option stdout to include anything from the standard output stream of the invoked program:

```
.. program-screenshot:: python tests/tkmsgbox.py 0
   :stdout:
```

The above snippet would render like this:

```
[stdout] before messagebox
```



5.5.4 stderr

Use option `stderr` to include anything from the standard error stream of the invoked program:

```
.. program-screenshot:: python tests/tkmsgbox.py 0
   :stderr:
```

The above snippet would render like this:

```
[stderr] before messagebox
```



5.5.5 wait

Use `wait` to wait at least N seconds after first window is displayed. This can be used to skip splash or loading screen.

`mintinstall` is loading data by start (without wait):

```
.. program-screenshot:: mintinstall
   :prompt:
   :scale: 50 %
```

The above snippet would render like this:

```
$ mintinstall
```



`mintinstall` after loading data (with wait):

```
.. program-screenshot:: mintinstall
    :prompt:
    :scale: 50 %
    :wait: 10
```

The above snippet would render like this:

```
$ mintinstall
```

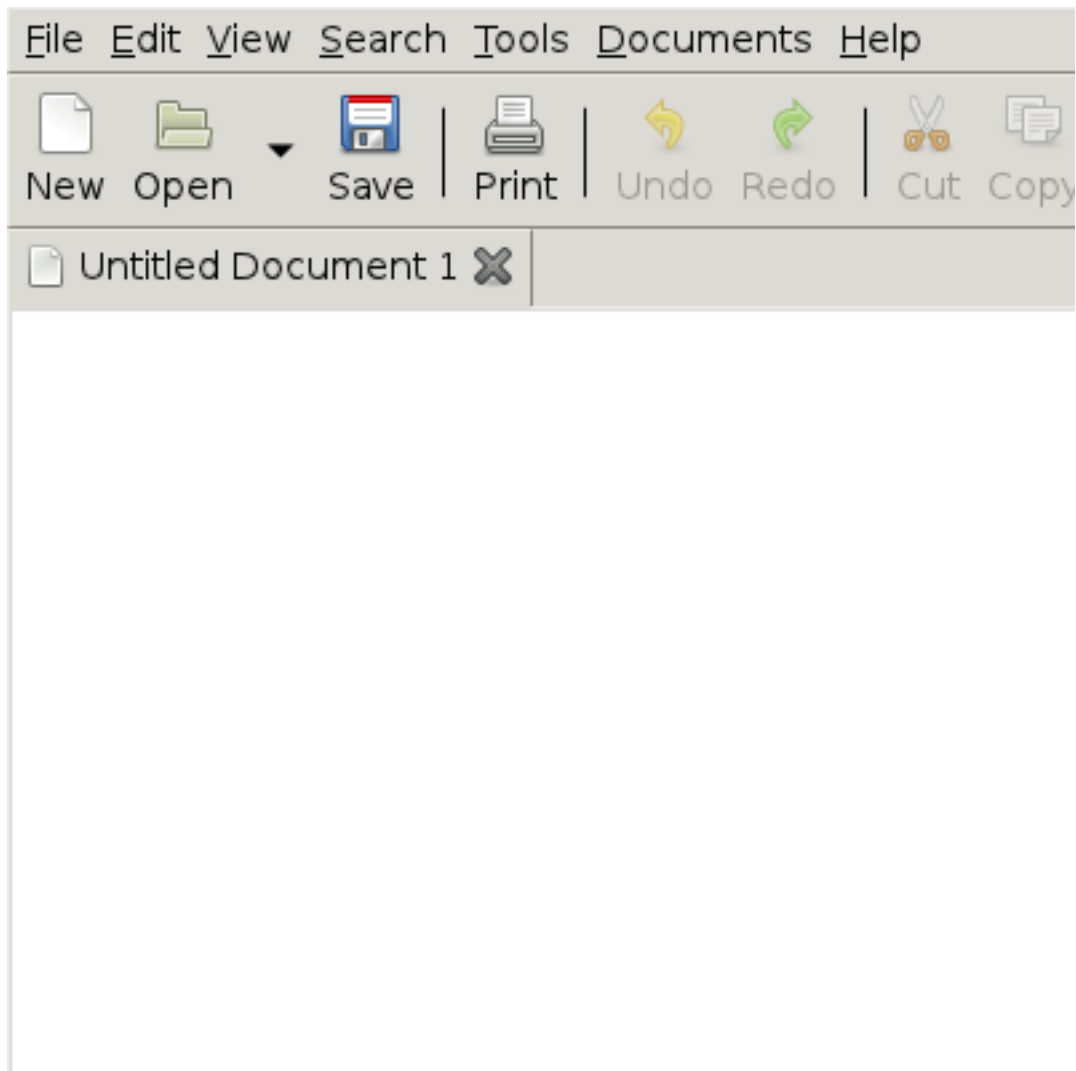


5.5.6 screen

Using the option `screen` you can set the screen size, default is 1024x768:

```
.. program-screenshot:: gedit
    :screen: 400x400
```

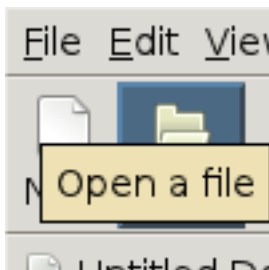
The above snippet would render like this:



Other resolution:

```
.. program-screenshot:: gedit
   :screen: 100x100
```

The above snippet would render like this:



5.6 Image options

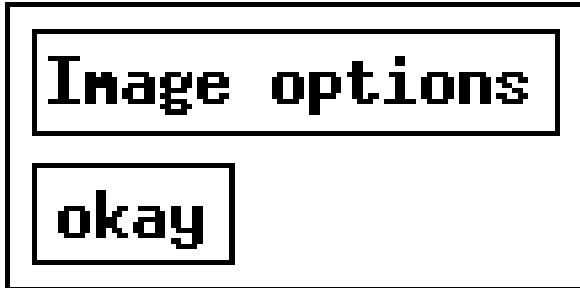
Same as in <http://docutils.sourceforge.net/docs/ref/rst/directives.html#image>

5.6.1 scale, alt

Example:

```
.. program-screenshot:: xmessage Image options
   :scale: 200 %
   :alt: alternate text
```

The above snippet would render like this:

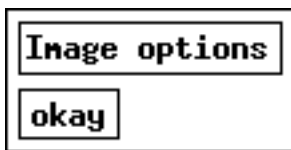


5.6.2 height, width

Example:

```
.. program-screenshot:: xmessage Image options
   :height: 100px
   :width: 100 px
```

The above snippet would render like this:

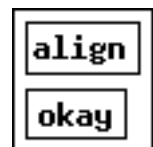


5.6.3 align

Example:

```
.. program-screenshot:: xmessage align
   :align: right
```

The above snippet would render like this:



TODO

- option `shell` like in `programoutput`
- option `color-depth` (`mono,8,16,24`)
- option `ellipsis` like in `programoutput`
- configuration `programoutput_prompt_template` like in `programoutput`
- configuration `programoutput_use_ansi` like in `programoutput`

DEVELOPMENT

7.1 Tools

1. `setuptools`
2. `Paver`
3. `nose`
4. `ghp-import`
5. `pyflakes`
6. `pychecker`
7. `paved fork`
8. `Sphinx`
9. `sphinxcontrib-programsscreenshot`
10. `sphinxcontrib-paverutils`
11. `autorun` from `sphinx-contrib` (there is no simple method, you have to download/unpack/setup)

7.2 Install on ubuntu

```
sudo apt-get install python-setuptools
sudo apt-get install python-paver
sudo apt-get install python-nose
sudo easy_install ghp-import
sudo apt-get install pyflakes
sudo apt-get install pychecker
sudo easy_install https://github.com/ponty/paved/zipball/master
sudo apt-get install scrot
sudo apt-get install xvfb
sudo apt-get install xserver-xephyr
sudo apt-get install python-imaging
sudo apt-get install python-sphinx
sudo easy_install sphinxcontrib-programsscreenshot
sudo easy_install sphinxcontrib-programoutput
sudo easy_install sphinxcontrib-paverutils
```

7.3 Tasks

`Paver` is used for task management, settings are saved in `pavement.py`. `Sphinx` is used to generate documentation.

print [paver](#) settings:

```
paver printoptions
```

clean generated files:

```
paver clean
```

generate documentation under *docs/_build/html*:

```
paver cog pdf html
```

upload documentation to [github](#):

```
paver ghpages
```

run unit tests:

```
paver nose  
#or  
nosetests --verbose
```

check python code:

```
paver pyflakes  
paver pychecker
```

generate python distribution:

```
paver sdist
```

upload python distribution to [PyPI](#):

```
paver upload
```