sphinxcontrib-aafig Documentation

Release 1.0

Leandro Lucarella

November 15, 2009

Contents

author Leandro Lucarella <llucax@gmail.com>

1 About

This extension allows embedded ASCII art to be rendered as nice looking images using aafigure.

aafigure is a program and a reStructuredText directive to allow embedded ASCII art figures to be rendered as nice images in various image formats. The aafigure directive needs a *hardcoded* image format, so it doesn't goes well with Sphinx multi-format support.

This extension adds the `aafig` directive that automatically selects the image format to use acording to the Sphinx writer used to generate the documentation.

You can see the latest documentation at the sphinxcontrib-aafig website or download it in PDF format.

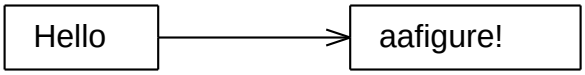
1.1 Quick Example

This source:

```
.. aafig::
    :aspect: 60
    :scale: 150
    :proportional:
    :textual:

    +-------+         +-----------+
    | Hello +-------->+ aafigure! |
    +-------+         +-----------+
```

is rendered as:



2 Download

You can see all the available versions at PyPI.

2.1 Requirements

- aafigure (0.3 or later).
- reportlab (for LaTeX/PDF output)
- PIL (for any image format other than SVG or PDF)

aafigure should be installed in the system for this extension to work. Alternatively you can download it to any folder and add that folder to the Python's path through the Sphinx `conf.py` file, for example:

```
import sys, os

sys.path.extend(os.path.abspath('path/to/aafigure')])
```

2.2 From source (tar.gz or checkout)

Unpack the archive, enter the sphinxcontrib-aafig-x.y directory and run:

```
python setup.py install
```

2.3 Setuptools/PyPI

Alternatively it can be installed from PyPI, either manually downloading the files and installing as described above or using:

easy_install -U sphinxcontrib-aafig

2.4 Enabling the extension in Sphinx

Just add `sphinxcontrib.aafig` to the list of extensions in the `conf.py` file. For example:

```
extensions = ['sphinxcontrib.aafig']
```

3 Usage

You can always use the original reStructuredText aafigure extension, but choosing a hardcoded format can be a bad idea when using Sphinx, because PDF might not be suitable for HTML and PNG can look ugly in a PDF document.

This extension uses the same aafigure code to add a more *Sphinxy* directive called `aafig`. This directive accepts the same options as the original aafigure directive (please, see aafigure documentation for more information). There are some differences though:

- The `:format:` option is not available, the format is selected automatically depending on the Sphinx builder you are using.
- `:scale:` and `:aspect:` options are specified using percentages (without the % sign), to match the reStructuredText image directive.

For an example on using the `aafig` directive see the Quick Example.

3.1 Configuration

A few configuration options are added (all optional, of course ;) to Sphinx so you can set them in the `conf.py` file:

`aafig_format` <dict>: image format used for the different builders. All `latex`, `html` and `text` builder are supported, and it should be trivial to add support for other builders if they correctly handle images (and if aafigure can render an image format suitable for that builder) by just adding the correct format mapping here.

A special format `None` is supported, which means not to use aafigure to render the image, just show the raw ASCII art as is in the resulting document (using a literal block). This is almost only useful for the text builder.

You can specify the format - builder mapping using a dict. For example:

```
aafig_format = dict(latex='pdf', html='svg', text=None)
```

These are the actual defaults.

`aafig_default_options` <dict>: default aafigure options. These options are used by default unless they are overridden explicitly in the `aafig` directive. The default aafigure options are used if this is not specified. You can provide partial defaults, for example:

```
aafig_default_options = dict(scale=1.5, aspect=0.5, proportional=True)
```

Note that in this case the `aspec` and `scale` options are specified as floats, as originally done by aafigure. See aafigure documentation for a complete list of options and their defaults.

4 TODO

- Add color validation for `fill`, `background` and `foreground` options.
- Add `aa` role for easily embed small images (like arrows).

5 ChangeLog

This file describes user-visible changes between the extension versions.

5.1 Version 1.0 (2009-11-15)

- Fix HTML builder rendering.
- Improve error reporting when aafigure package is missing.

5.2 Version 0.3 (2009-07-13)

- Add website.
- Change license to BOLA.
- Add partial format redefinition via the config option `aafig_format`.
- Add support for the text builder (using the `None` format).
- Add a special format `None` meaning not to render the figure using aafigure at all (just include the raw ASCII art as a literal block).
- Fix SVG support for HTML builders.
- Make aspect and scale options take a percentage instead of a float (this is more consistent with the image directive).
- Add support for image directive options: alt, align, width, height, class, target.
- Add dependency against aafigure package to ease installation.
- Improve error handling in many ways. For example, if the format is not supported, or if aafigure is not installed, the raw ASCII art is inserted into the document as a literal block.
- Fix a couple of bugs.

5.3 Version 0.2 (2009-06-07)

- Add `aafig_default_options` configuration option.
- Fix SVG output for HTML writer (this need aafigure r5978 or later).

5.4 Version 0.1 (2009-06-05)

- Initial version.

6 License

I don’t like licenses, because I don’t like having to worry about all this legal stuff just for a simple piece of software I don’t really mind anyone using. But I also believe that it’s important that people share and give back; so I’m placing this work under the following license.

BOLA - Buena Onda License Agreement (v1.0)

This work is provided ’as-is’, without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this work.

To all effects and purposes, this work is to be considered Public Domain.

However, if you want to be "buena onda", you should:

1. Not take credit for it, and give proper recognition to the authors.
2. Share your modifications, so everybody benefits from them.
3. Do something nice for the authors.
4. Help someone who needs it.
5. Don’t waste. Anything, but specially energy that comes from natural non-renewable resources.
6. Be tolerant. Everything that’s good in nature comes from cooperation.