# Socrates Documentation

***Release 0.9.3***

**Honza Pokorny**

**Sep 28, 2017**

# Contents


**1 Features** **3**
- 1.1 Installation . . . 3
- 1.2 Usage . . . 3
- 1.3 Configuration . . . 4
- 1.4 Writing posts . . . 6
- 1.5 Example posts . . . 8

**2 Indices and tables** **9**

Socrates is a simple static site generator. It's geared towards blogs. You write your posts in your favorite plain text to HTML language (e.g. Markdown, textile) and save them as text files on your harddrive. Socrates then takes them, and creates a full HTML site for you. For free, you will get a home page which lists latest posts, single post pages, category pages, archive pages, an about page and an atom feed.

# CHAPTER 1

# Features

- Familiar Django and Jinja2 templates
- Simple install via pip
- Markdown, reStructuredText, Textile support
- YAML configuration
- Atom feed
- Github pages compatible

Contents:

## Installation

Via PyPI:

```
pip install socrates
```

Or, get the latest code from Github:

```
pip install -e git://github.com/honza/socrates.git#egg=socrates
```

## Usage

First, you need to create a new blog:

```
$ socrates -i blog
```

This will create a `blog` directory with a simple blog structure:

```
blog
    posts
        2010-your-post.md
    pages
        about.md
        contact.md
    layout
        index.html
        single.html
        category.html
        ...
    media
        style.css
    config.yaml
```

The `posts` directory is where you will place your posts files. Anything prefixed with _ or `.` will be ignored. `layout` is your basic theme or a template. `config.yaml` is a site-wide configuration file. Don't forget to update the about file with relevant information.

You can also create a new blog in the current working directory:

```
$ socrates -i
```

When you are ready to generate your site, you run:

```
$ socrates -g blog
```

Or,

```
$ socrates -g
```

for current directory.

This will place all the generated files in `blog/deploy`. You can then take that directory and upload it to your server.

## Configuration

This is a reference for your blog's main configuration file. Any values that you add in there will be made available in your templates' context.

**`author`**
: Blog author. Defaults to *author*.

**`site_name`**
: Site name. Defaults to *Socrates site*.

**`posts_per_page`**
: Number of posts displayed per page. Used for pagination. Defaults to *10*. Setting this to *0* will display all posts on one page.

**`url`**
: Your site's URL. Defaults to *http://example.com*.

**`date_format`**
: Python strftime formatted date format. Defaults to *%B %d, %Y*.

**text_processor**
Which X to html processor to use; *markdown*, *textile*, *rst*, *html*, *extension*. Defaults to *markdown*. The 'extension' setting will decide on the processor to be used based on the post's file extension:

- **Markdown**
  - .md
  - .markdown
  - .mkdn
- **reStructuredText**
  - .rst
- **HTML**
  - .html
  - .htm
  - .txt
- **Textile**
  - .textile

**templates**
'django' or 'jinja2'. Defaults to *django*.

**append_slash**
Whether a slash should be appended to post urls. Defaults to *true*.

**url_include_day**
Whether to include the day with the month and year in the generated directories and urls. Defaults to *false*.

**initial_header_level**
By default, the first heading in your document will be `<h2>`. Only available for reStructuredText posts. Defaults to *2*.

**skip_archives**
If set to *true*, it won't bother generating archives. Defaults to *false*.

**skip_categories**
If set to *true*, it won't bother generating categories. Defaults to *false*.

**pygments**
Additional settings for the Pygments HTML Parser. It passes the arguments directly to the `HtmlFormatter` class when it's instanciated, so these settings include all of the available settings for `HtmlFormatter`

sample:

```
pygments:
    linenos: true
    noclasses: false
    style: 'pastie'
```

**style**
The style option has many default built in styles for your code blocks. The ones that ship with Pygments are: `monokai`, `manni`, `perldoc`, `borland`, `colorful`, `default`, `murphy`, `vs`, `trac`, `tango`, `fruity`, `autumn`, `bw`, `emacs`, `vim`, `pastie`, `friendly`, `native`

**inline_css**
Whether or not you want pygments to output a `pygments.css` file in your build directory for css. If set to `false` it will output the file.

**punctuation**
Whether common punctuation characters should be replaced with proper HTML characters. Defaults to `False`. E.g.:

- em dash
- en dash
- double quotes
- single quotes
- apostrophe

This functionality is provided by `smartypants` and `typogrify` and only works in Django at this time.

**ligatures**
Defaults to `False`

**deploy_dir**
Defaults to `deploy`

# Writing posts

All of your posts will typically be contained in the `posts` directory in your main blog directory. Your post file can be called anything you want, and as long as you're not mixing and matching different text processors, the file extension can be anything, too. I use the following naming convetion:

```
2011-07-29-name-of-post.rst
```

This way my posts are automatically ordered by publish date when I run `ls` in the `posts` directory.

## Markdown, textile and HTML

When you're writing your posts in Markdown (or textile or HTML), you need to add a bit of text to the top of your file to provide Socrates with some metadata about your post.

```
---------------------------------------------------------------------
title: Title of your post
date: 2011-07-29 13:00:00
categories:
    - Photos
    - Vacation
---------------------------------------------------------------------
```

The text between the two horizontal line is written in YAML syntax. Note that the horizontal line should have at least 79 characters.

## reStructuredText

If you want to write your post in reStructuredText, you should use the rst native way to specify document metadata. Include this at the top of your post file:

```
:title: Title of your post
:date: 2011-07-29 13:00:00
:categories: Photos, Vacation
```

This way, your posts can be processed by the native Docutils utility functions such as `rst2html.py` or `rst2latex.py`.

## Options

**`title`**
The title of the post

**`slug`** (optional)
The url-ized transformation of your title. This is useful if you need to maintain an existing url format.

**`date`**
Publish date; `YYYY-MM-DD HH:MM`

**`categories`**
A list of categories

**`template`**
You can override the default template that the post is going to be rendered with.

## Syntax highlighting

Socrates has built-in support for syntax highlighting via pygments. It's only available for the markdown and reStructuredText flavors.

### Markdown

For markdown files, we use the same syntax as Github for specifying code blocks.

````
Paragraph text

```python
import datetime
print datetime.datetime.utcnow()
```

More text
````

### reStructuredText

In reStructuredText, you can use the `code-block` or `sourcecode` directives.

```
Paragraph text

.. code-block:: python

    import datetime
    print datetime.datetime.utcnow()

More text
```

## Example posts

Here is an example post written in reStructuredText.

```
:title: `How to make a function in Python`
:date: `2011-07-20 09:00`
:categories: `Code, Python`

How to make a function in Python
================================

A function in Python starts with the keyword ``def``, followed by the
function name and a set of parenthesis.

.. code-block:: python

    def say_hello():
        ...

You can read more about this in the `Python manual`_.

.. _Python manual: http://docs.python.org/tutorial/controlflow.html#defining-functions
```

And here is the same post in Markdown.

```
---------------------------------------------------------------------------
title: How to make a function in Python
date: 2011-07-20 09:00:00
categories:
    - Code
    - Python
---------------------------------------------------------------------------

How to make a function in Python
================================

A function in Python starts with the keyword `def`, followed by the
function name and a set of parenthesis.

    def say_hello():
        ...

You can read more about this in the [Python manual][1].

[1]: http://docs.python.org/tutorial/controlflow.html#defining-functions
```

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Index


## A

append_slash
    command line option, 5
author
    command line option, 4

## C

categories
    command line option, 7
command line option
    append_slash, 5
    author, 4
    categories, 7
    date, 7
    date_format, 4
    deploy_dir, 6
    initial_header_level, 5
    inline_css, 5
    ligatures, 6
    posts_per_page, 4
    punctuation, 6
    pygments, 5
    site_name, 4
    skip_archives, 5
    skip_categories, 5
    slug (optional), 7
    style, 5
    template, 7
    templates, 5
    text_processor, 4
    title, 7
    url, 4
    url_include_day, 5

## D

date
    command line option, 7
date_format
    command line option, 4
deploy_dir
    command line option, 6

## I

initial_header_level
    command line option, 5
inline_css
    command line option, 5

## L

ligatures
    command line option, 6

## P

posts_per_page
    command line option, 4
punctuation
    command line option, 6
pygments
    command line option, 5

## S

site_name
    command line option, 4
skip_archives
    command line option, 5
skip_categories
    command line option, 5
slug (optional)
    command line option, 7
style
    command line option, 5

## T

template
    command line option, 7
templates
    command line option, 5
text_processor

command line option, 4
title
command line option, 7

## U

url
command line option, 4
url_include_day
command line option, 5