PypiMonitor Documentation

Release 0.5.0

Louis Paternault

Mar 05, 2024

CONTENTS

1 Modules **3**
- 1.1 `pypimonitor` . . . 3
- 1.2 `pypimonitor.httpd` . . . 3

2 YAML configuration files **5**
- 2.1 Processing YAML files . . . 5
- 2.2 Writing YAML files . . . 5

3 Cell plugins **9**
- 3.1 List of plugins . . . 9
- 3.2 Write your own . . . 11

4 Example **15**

5 Indices and tables **17**

Python Module Index **19**

Index **21**

An HTML dashboard to monitor your PyPI packages. It displays a line charts showing the evolution of downloads across versions, and a set of badges (download statistics, readthedocs badge, continuous integration, etc.). See *the example* below.

It can only monitor stuff that is not on Pypi, but less information will be available.

It is available as a `command line interface` that generates the HTML code, and as a `web server`, to generate and serve this dashboard.

Contents:

CHAPTER

ONE

MODULES

1.1 pypimonitor

This plugin can be called to generate an HTML page, whose code is printed to standard output.

Produce an HTML dashboard to monitor your PyPI packages.

```
usage: pypimonitor [-h] [--version] [-u USER] [-c CELL] [-p PACKAGE] [yaml]
```

1.1.1 Positional Arguments

yaml	Configuration file.

1.1.2 Named Arguments

--version	Show version
-u, --user	A comma-separated list of users, whose packages are to be monitored. Default: []
-c, --cell	A comma-separated list of cells to show. Default: []
-p, --package	A comma-separated list of packages to monitor. Default: []

1.2 pypimonitor.httpd

Alternatively, this module can sever the web pages, to be accessible from a web browser. If served on http://localhost, the following URLs are available:

- http://localhost (and http://localhost/index.html): If no GET arguments are given, display an index page, with a form to ask to render some pages. It also accepts GET arguments to specify which packages to process, and which cell plugins to use.
- http://localhost/foo/bar: When running the server, a directory `DIR` is given as argument. When calling this URL, a file `DIR/foo/bar.yaml` is searched, and if it exists, this file *is processed* to render the HTML page.

CHAPTER

TWO

YAML CONFIGURATION FILES

Some plugins try to guess the appropriate required values, but it is not always possible. For instance, plugin *readthedocs* cannot guess the documentation URL, since it can differ from `http://<PypiPackageName>.readthedocs.io` (for instance, the documentation URL of sphinxcontrib-packages is http://packages.readthedocs.io and not http://spinxcontrib-packages.readthedocs.io). Thus, it may be necessary to provide additional information. This can be done using YAML files.

2.1 Processing YAML files

Configuration files can be processed by both `pypimonitor` command line interface, and by the `pypimonitor.httpd` web sever. See the relevant documentation for more information.

2.2 Writing YAML files

2.2.1 Example

The example page is produced using the following YAML file:

```yaml
default:
  CI:
    cell: gitlabci
    server: //framagit.org
    user: spalax
  Coverage:
    cell: gitlabcoverage
    server: //framagit.org
    user: spalax
cells:
  - color
  - homepage
  - pypiversion
  - pythonversions
  - pypimdownloads
  - pypiwdownloads
  - pypiddownloads
  - readthedocs
  - CI
  - Coverage
```

(continues on next page)

(continued from previous page)

```yaml
packages:
  argdispatch:
  annales-math:
    homepage:
      homepage: //framagit.org/lpaternault/annales-math
    CI:
      cell: gitlabci
      server: //framagit.org
      user: lpaternault
    Coverage:
      cell: gitlabcoverage
      server: //framagit.org
      user: lpaternault
  cahier:
  chval:
  clachievements:
  devoir:
  dummypdf:
  evariste:
  fullcoverage:
  jouets:
  mklog:
  papersize:
  paste2sms:
  pdfautonup:
  pdfimpose:
  pdfimpose-web:
  pypimonitor:
  scal:
  sphinxcontrib-packages:
    readthedocs:
      slug: packages
  sphinxcontrib-proof:
  sphinxcontrib-stuffcounter:
    homepage:
      homepage: //framagit.org/spalax/sphinxcontrib-stuffcounter
  spix:
  squelette:
  toto2titi:
    CI:
      cell: gitlabci
      server: //framagit.org
      user: spalax
      slug: paste2sms
    Coverage:
      cell: gitlabcoverage
      server: //framagit.org
      user: spalax
      slug: paste2sms
    readthedocs:
      slug: paste2sms
```

2.2.2 Configuration options

The YAML configuration is a dictionary, with the following keys : *default*, *cells*, *packages*. There can be additionnal keys, used by somme `cell plugins` (at the time I am writing this, no plugin uses this).

In the following example, the YAML configuration file is reffered as a Python `dict`.

`cell` option

The cell plugin used to render column *foo* of line *mypackage* is the plugin having as keyword (by order of precedence):

- value of *config['packages']['mypackage']['foo']['cell']* (to explicitely set the plugin to use for a single package);
- value of *config['default']['foo']['cell']* (to explicitely set the default plugin to use for a whole column);
- *foo* (at last, the column reference is used as the cell plugin keyword);
- the default *error* plugin.

`packages`

This is a dictionary of dictionaries: the keys are the pypi package names, and the values are either nothing (if the default values are sufficient to process this package), or a dictionary of cell options: the keys of this "sub-dictionary" are the cell names, and the values are dictionary of cell options.

For instance, in the *example*:

- package *fullcoverage* uses only default values, so it is mentionned without options;
- however, package *sphinxcontrib-packages* has, as a value, `{'readthedocs': {'slug': 'packages'}}`, which means that options `{'slug': 'packages'}` is passed to the *readthedocs* plugin (which means that the documentation URL is http://packages.readthedocs.io instead of http://sphinxcontrib-packages.readthedocs.io).

`cells`

If *config['cells']* is not defined, the list of columns is deduced from the cells used in the package options. This option has two purposes:

- explicitely set the list of columns (for instance, in *the example*, since *color* is never referenced in package options (every package use the default options for this plugin), it would not appear in the generated HTML file if it were not present in the *config['cells']* list);
- set the order of those columns.

The values of this list can be:

- a `cell plugin` keyword, in which case, unless *otherwise specified*, the plugin used to render this cell is the corresponding plugin, and the title of the column is the title of this plugin;
- or an arbitrary text, in which case each package has to *explicitely define* its cell plugin for this cell, or the cell plugin to use has to be defined in the `default` value (see *default*).

default

Default cell parameters can be set, and apply to every package (unless a different parameter is set specifically for this program). This option is a dictonary, where:

- keys are the column names (as referenced in the `cells` option);
- values are dictionary of options, which are applied to every package, unless the package explicitely specified a different option.

For instance, in *the example* the `default` configuration contains:

```
CI:
  cell: gitlabci
  server: http://framagit.org
  user: spalax
```

This means that, unless a package specifies something else:

- the plugin used to render the cells is *gitlabci*;
- the gitlab server is http://framagit.org (and not the plugin default http://gitlab.com);
- the user is spalax.

CHAPTER

THREE

CELL PLUGINS

The content of cells is rendered by plugins. A *predefined list* is detailed below, but you can also *Write your own*.

3.1 List of plugins

3.1.1 Base plugins

`color`

Produce a square of the same color of the corresponding download chart line. Takes no arguments.

`empty`

Produce an empty cell. Takes no arguments.

`error`

Produce a text corresponding to an error (this is used internally). Takes no arguments.

`html`

Copy raw html code.

- `html` (required) The raw html code to render.

`link`

Produce a link to an URL.

- `href` (required) URL of the ressource to link to.
- `content` (defaults to the *href* argument) The text of the link.

3.1.2 Gitlab

gitlabci

Produce a badge for latest gitlabCI build.

- `server` (default *http://gitlab.com*) URL of the gitlab server.
- `user` (required) Name of the user owning the package.
- `slug` Repository name, if different from the pypi package name.

Those options, combined, should produce the package URL: `{server}/{user}/{slug}`.

gitlabcicoverage

Produce a test coverage badge for latest gitlabCI build.

- `server` (default *http://gitlab.com*) URL of the gitlab server.
- `user` (required) Name of the user owning the package.
- `slug` Repository name, if different from the pypi package name.

Those options, combined, should produce the package URL: `{server}/{user}/{slug}`.

3.1.3 PyPI

homepage

Package home page, retrieved from Pypi metadata.

- `homepage` (default to pypi home page value) Project home page, if different from the value retrieved from pypi.

3.1.4 Readthedocs

readthedocs

Readthedocs build badge.

- `slug` Repository name, if different from the pypi package name.
- `branch` Branch name, if the default branch should not be used.
- `lang` Documentation language, if the default lang should not be used.

3.1.5 Shields

pypiddownloads

Badge displaying Pypi daily download statistics.

pypiwdownloads

Badge displaying Pypi weekly download statistics.

pypimdownloads

Badge displaying Pypi monthly download statistics.

pypiversion

Badge displaying Pypi version.

pythonversions

Badge displaying supported Python versions.

3.1.6 Travis

travisci

Travis badge.

- `user` (required) Travis username.
- `slug` Repository name, if different from the pypi package name.

3.2 Write your own

The HTML page is generated using the Jinja2 template system, but you don't *have to* use it for your plugins.

A plugin is an object, subclass of `Cell`. The `Cell.render()` method must be implemented: it is the method that is called to fill a cell. Since for many usage, your plugin will simply be a template, you can use the `Jinja2` class, which makes this a lot easier. Both classes makes it easy to define *default and required arguments*, and to *log errors*.

3.2.1 Raw

class pypimonitor.cell.**Cell**(*renderer*)

Render some piece of information about a package as HTML code.

keyword = None

Keyword referencing the plugin, used in the *YAML configuration files* file to enable this plugin. If `None`, the class is an abstract class that cannot be used directly.

title = ''

Title of the column.

default = {}

Default values for package arguments. See *Default and Required arguments* for more details.

required = []

List of names of required package arguments. See *Default and Required arguments* for more details.

render(*context*, *package*, *cell*)

Return the HTML code corresponding to this cell.

Parameters

- **context** – Current Jinja2 context.
- **package** (*str*) – Package name.
- **cell** (*dict*) – Package arguments for this cell.

Return type

str

Returns

The HTML code to display in the given cell.

static render_error(*context*, *cell*, *package*, *message*)

Return the HTML code corresponding to an error.

Parameters

- **context** – Current Jinja2 context.
- **cell** (*str*) – Cell name (plugin keyword).
- **package** (*str*) – Package name.
- **message** (*str*) – Human readable error message.

Return type

str

Returns

The HTML code to display in the given cell.

3.2.2 Jinja2

class pypimonitor.cell.**Jinja2**(*renderer*)

Generic class for cells that are barely more than a template.

When this class is used to render a cell, it renders template `self.keyword`. When doing so, the template has access to the following variables:

- *package*: the name of the package being processed, as a string;
- *pypi*: the information about this package got from pypi, as a dictionary (for instance https://pypi.org/pypi/pypimonitor/json);
- *cell*: the cell options (as defined in the *YAML configuration files* configuration, maybe completed with *default values*) as a dictionary.

property template

Return template path.

By default, this is `cells/KEYWORD.html`. One can redefine this class to provide alternative template path.

keyword = None

Keyword referencing the plugin, used in the *YAML configuration files* file to enable this plugin. If `None`, the class is an abstract class that cannot be used directly.

3.2.3 Default and Required arguments

A built-in feature eases processing default and required arguments, for simple cases. This is automatically done before calling the `Cell.render()` method.

If your plugin has absolute default values for some arguments, those can be set in the `Cell.default` dictionary. Keys are arguments, and values are default values for these arguments.

If your plugin has required arguments (plugin cannot work without those arguments), they can be listed in the `Cell.required` list. Using a cell without setting one of those arguments will display an error.

For more complex cases (either *foo* or *bar* should be set; *foo* default value is *bar* if *baz* is not set, *None* otherwise; *foo* is required only if *bar* is not set; etc.), you have to implement it by hand in the `Cell.render()` method.

3.2.4 Errors

To display an error (both in the generated HTML page, and in the log), the `Cell` class has a `Cell.render_error()` method. This is to be used as:

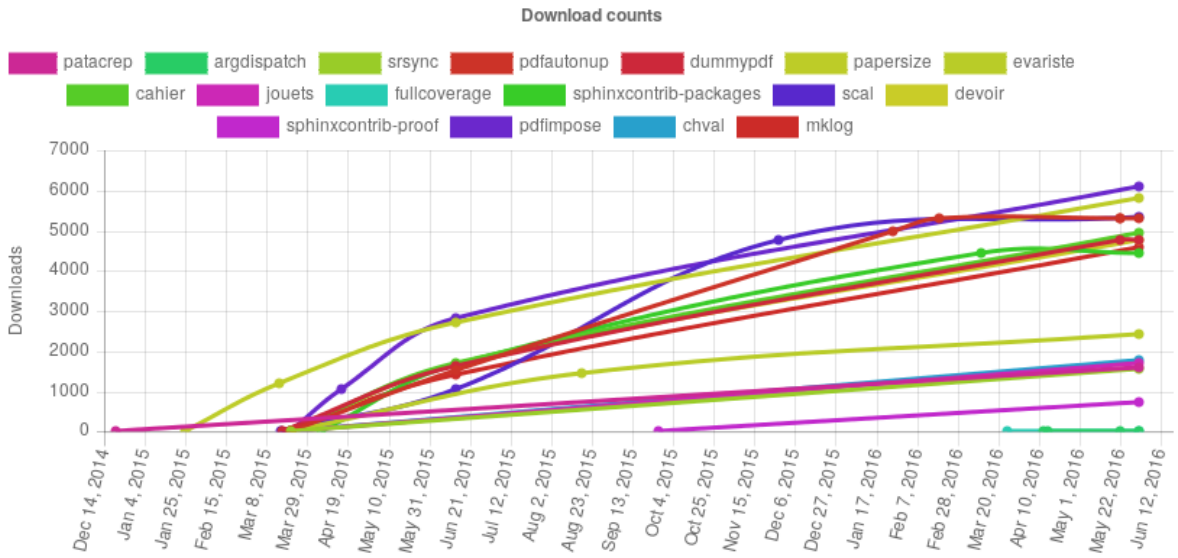
```
class Foo(Cell):
    keyword = 'foo'

    def render(self, context, package, cell):
        if not 'bar' in cell:
            return self.render_error(context, self.keyword, package, "Error: argument
↪'bar' missing.")
        return "<p>" + cell['bar'] + "</p>"
```

CHAPTER

FOUR

EXAMPLE

The *example configuration file* produces the following output (click to enlarge).



Home	Version	Monthly	Weekly	Daily	Doc	CI
argdispatch	pypi v0.1.1	downloads 2/month	downloads 1/week	downloads 0/day	docs latest	build success
Cahier	pypi v0.1.1	downloads 1/month	downloads 1/week	downloads 0/day	docs latest	build success
chval	pypi v0.6.7	downloads 0/month	downloads 0/week	downloads 0/day	docs next	build unknown
Devoir	pypi v0.1.1	downloads 1/month	downloads 1/week	downloads 1/day	docs latest	build success
DummyPdf	pypi v0.2.0	downloads 2/month	downloads 1/week	downloads 0/day	docs latest	build unknown
Evariste	pypi v0.1.0	downloads 0/month	downloads 0/week	downloads 0/day	docs latest	build success
fullcoverage	pypi v0.1.0	downloads 7/month	downloads 0/week	downloads 0/day	docs latest	build success
Jouets	pypi v0.1.0	downloads 0/month	downloads 0/week	downloads 0/day	docs latest	build success
mklog	pypi v0.3.3	downloads 1/month	downloads 1/week	downloads 0/day	docs latest	build success
PaperSize	pypi v0.1.3	downloads 0/month	downloads 0/week	downloads 0/day	docs latest	build success
patacrep	pypi v4.0.0	downloads 0/month	downloads 0/week	downloads 0/day	docs latest	build passing
PdfAutoNup	pypi v0.4.1	downloads 1/month	downloads 0/week	downloads 0/day	docs latest	build unknown
PdfImpose	pypi v0.1.1	downloads 3/month	downloads 1/week	downloads 1/day	docs latest	build success
Scal	pypi v0.3.0	downloads 3/month	downloads 1/week	downloads 0/day	docs latest	build success
sphinxcontrib-packages	pypi v0.1.2	downloads 1/month	downloads 0/week	downloads 0/day	docs latest	build success
sphinxcontrib-proof	pypi v0.1.0	downloads 1/month	downloads 0/week	downloads 0/day	docs latest	build success
srsync	pypi v1.0.0-dev	downloads 1/month	downloads 0/week	downloads 0/day	docs python3	build unknown

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

pypimonitor, 3
pypimonitor.cell, 9
pypimonitor.httpd, 3

INDEX

C

`Cell` (*class in pypimonitor.cell*), 11

D

`default` (*pypimonitor.cell.Cell attribute*), 11

J

`Jinja2` (*class in pypimonitor.cell*), 12

K

`keyword` (*pypimonitor.cell.Cell attribute*), 11
`keyword` (*pypimonitor.cell.Jinja2 attribute*), 12

M

`module`
    `pypimonitor`, 3
    `pypimonitor.cell`, 8
    `pypimonitor.httpd`, 3

P

`pypimonitor`
    `module`, 3
`pypimonitor.cell`
    `module`, 8
`pypimonitor.httpd`
    `module`, 3

R

`render()` (*pypimonitor.cell.Cell method*), 12
`render_error()` (*pypimonitor.cell.Cell static method*), 12
`required` (*pypimonitor.cell.Cell attribute*), 11

T

`template` (*pypimonitor.cell.Jinja2 property*), 12
`title` (*pypimonitor.cell.Cell attribute*), 11