rstblog Documentation

Release v0.1

luciano de falco alfano

Aug 24, 2020

Contents:

`rstblog` project is about a simple blog driven by articles written using reStructuredText markup language.

This is its documentation.

CHAPTER 1

author manual

author manual table of contents

– *Author manual end*

Below *default* and *optional* items are indicative by now. In this version of `rstblog` an optional field is set to its default value in case of load of a new article. Otherwise, i.e. changing an article already present in DB, an optional field is ignored, causing the previous value to survive. If you wish to change it, set its value explicitly.

1.1 Article organization

Let's speak how organize an article that we'll publish using `rstblog`.

It is written in a file with two parts. The first part is about fields categorizing our article. The second part has content of the article.

The two parts are separated by a line containing the string:

```
.. hic sunt leones
```

Yes: two dots, space, and the phrase *hic sunt leones*. No more, no less, in a single line[1].

So, reiterating the concept, we have this schema:

fields area (generally: one field every line)
optional empty line
`.. hic sunt leones`
optional empty line
article contents

Hereafter, as example, we report first lines of an article about Marco Tullius Cicero, whose content is extracted from wikipedia. It's written using reST:

```
:markup: restructuredtext
:language: en
:title: Speaking about Cicero
:created: 2018-08-05 10:00:00
:modified: 2018-08-05 10:00:00
:slug: speaking-about-cicero
:summary:  The Marcus Cicero's profile: informations and life.
:authors:   Wikipedia
:category: latin literature history

.. hic sunt leones

Speaking about Cicero
=====================

*Marcus Tullius Cicero* was a Roman statesman, orator, lawyer
and philosopher, who served as consul in the year 63 BC.
He came from a wealthy municipal family of the Roman equestrian
order, and is considered one of Rome's greatest orators
and prose stylists.

...
```

[1] A point to rember. If you wish, this signal could be changed by the *site manager*. And an anecdote. People say that this phrase was used in the maps of ancient Rome, to indicate unexplored territories of Africa. But there is no firm evidence that this is true. In this context we adopt it to indicate that from here on we enter the unknown meanders of the creation of the article.

1.2 Article filename

Some speech about about article file. How to name it? Our advice is: create a rule and follow it. So you'll have a clearer working area.

As example, you can use a progressive number and a very short title note.

But whatever rule you'll adopt, it will be right: `rstblog` is filename agnostic. Just a caution: it would be better if file extension is related to the format used; i.e.: `.md` for *markdown* text, `.rst` for *reST* text and `.html` for *html* text.

Two **warnings about filenames**:

- 1st: you cannot use the same filename to write two different articles; this is obvious: on your PC, if you try to save a new article using a used filename, you'll scratch the old article;
- 2nd: you cannot change filename to an article already uploaded; this is less obvious, but trust me: it is true; if you need to change filename to an old article, you must tell it to the site administrator: he knows how to do it.

Now we'll speak in more detail about fields and content areas.

1.3 Article fields (aka: attributes)

As we saw, fields categorize our article. So they are vital.

`rstblog` uses fields shown in previous example *article about Marco Tullius Cicero*. There is one more, but we'll talk about it in a while.

By now, we exhort you to use all the fields shown in the example and to pay attention to typos. At this early stage of development (v0.2 as we write) there aren't a lot of controls about syntax errors.

A single field has structure:

> **:fieldname:** *fieldvalue*

`rstblog` decides **fieldname**(s). So you must use the right fieldname without typos. Instead what to put in *fieldvalue* is up to you.

Let's see the single fields meaning.

1.3.1 markup

This specify what markup language you use *to write article content*. Note the phrase *article content*. In fact field area is ever written using reST syntax.

Acceptable values for this field are: `markdown`, `restructuredtext`[2], `html`.

Optional: no.

Example:

```
:markup: restructuredtext
```

[2] Note the use of the full name of the sintax type.

1.3.2 language

This is about what language you use to write the article content.

Acceptable values are defined from your site configuration. And it's the site master responsability to configure it. Probably, at least english (written as `en`) would be available. Languages are invoked using their abbreviations; i.e. `it` for italian, `fr` for french, `es` for spanish, and so on.

Optional: no.

Example:

```
:language: it
```

1.3.3 title

This is the article title. It is shown in the blog index to identify your article and as a link to read it.

Acceptable values: whatever you want, provided that there are no other articles with the same title in the blog. Article title must be unique in the site. The maximum length is 250 characters.

Optional: no.

Example:

```
:title: Speaking about Cicero
```

1.3.4 created and modified

These are two fields showing:

- the first the article creation date and time;
- and the second the article last modified date and time.

Acceptable values. Whatever, in the format: **YYYY–MM–DD HH : MM : SS**

Optional: yes.

Default value: current date.

Example:

```
:created: 2018-08-05 10:00:00
:modified: 2018-08-05 10:00:00
```

1.3.5 slug

Slug is the last piece of information used in the URL to reach your article. Usually it reflects the article title to help the reader (and the web crawler programs) to remember the article title.

Acceptable values. As titles, even slugs must be unique in the blog. Futhermore, they must be composed of a subset of ansi characters. To stay smooth, it's usual to use only lowercase regular letters, with puntuation marks and spaces substitued by dashes. Maximum length is 250 characters.

Optional: no.

Example. If your article would be reached by this url: `https://my.blog.org/blog/show/speaking-about-cicero`, you'll use:

```
:slug: speaking-about-cicero
```

1.3.6 summary

This field value summarizes your article content. It is shown in the blog index page after the title of article.

Accepted values. No restrictions here. And this field can accept even multiple lines contents. If you want to use multiple lines, you need to indent it from the second line on.

Optional: yes.

Default: the empty string.

Example of multiple lines summary:

> **summary** The Marcus Cicero's profile: informations and life. From wikipedia in english language.

1.3.7 authors

Put here the name(s) of author(s) of the article (your name, I suppose :-). In case of multiple authors, keep them in one line and separate them using a comma (,).

Accepted values. Usually Author name must be present in blog database. It is responsability of site manager to insert the names of accepted authors.

If you are sure it is necessary to add a new Author, not yet present in database, you can force its creation using the *insert_author* control field (see below).

Optional: yes.

Default: null.

Example:

> **authors** Lawrence of Arabia

1.3.8 category

This is the master of categorizations. It catalogs our article assigning it to a main type.

Accepted values. Again, it depends on the configuration of your blog. It is responsability of site manager to insert the accepted categories in the blog database. Usually only values present in this database are accepted by `rstblog`.

If you are absolutly sure it is necessary to add a new category, not yet present in database, you can force its creation using the *insert_category* control field (see below).

Optional: no.

Example:

```
:category: latin literature history
```

1.4 Content

What to say about content?

Here the author develops his true work: to write the articles contents.

You are free to choose the format type you like throught *markdown*, *reST* and *html*.

Let us to give you just some advices about other files you could refer from your article.

First of all: the external hyperlinks. These are html pages available thanks to other sites. And all three quoted formats allow to refer them. As an example, this is an external hyperlink to wikipedia main page using reST:

```
`wikipedia <https://en.wikipedia.org/wiki/Main_Page>`_
```

It shows word `wikipedia` and it jumps to its main page if you click on the word.

Then, what about hyperlink to other article in the site? In this case, use the (relative) article URL. Remember: it uses `/blog/show` as prefix, and slug as article identifier. So to hyperlink to your article *Speaking about Cicero* you can use (for example):

```
...
you can read our wonderful `article about Cicero </blog/show/speaking-about-cicero>`_
...
```

Note that it isn't necessary to report the site domain (`my.blog.org`), and we use the article slug.

And, last but not least, how hyperlink to other files (not articles) present in our site? Here we need some technical clarifications to keep in touch.

In our site, files that aren't articles can live on these directories:

- `pages` that hosts the site pages that aren't articles;
- `media` that hosts other type of files, such as images, audio, video, pdf, and so on.

Usually `media` has one subdirectory for every kind of hosted file. I.e.:

- `media/images` to keep images;
- `media/pdfs` to store pdf files, and so on.

As you can argue, if you would hyperlink to `mylife.pdf` file, you can use:

```
...
`here </media/pdfs/mylife.pdf>`_ you can know something more about my life.
...
```

By now, these files must be uploaded to your site using some other kind of software; maybe ftp, or remote copy. This means that you must be a true site administrator to handle this files. If this is a problem for you: stay tuned ... In the future it's possible `rstblog` could upload even these files with the article.

A last note. When you would publish your work, you need to call:

```
https://my.blog.org/blog/load-article
```

`rstblog` will ask you for your username and password. When you'll give them to it, it will ask for the article filename to load. Here you can browse to the article file[3] and submit it, loading the request file.

[3] Or directly type it, if you remember its full path and name.

1.5 Advanced attributes

Hereafter more fields, useful in case of more advanced functions.

1.5.1 translation_of

Surprise: a field name not quoted in the *article about Marco Tullius Cicero*! What is this? You can send to `rstblog` even articles that are translations of article already known by `rstblog`. If is this the case, in this field you write the title of the *original* (translated) article.

If this field is missing, the article is an *original* article, meaning it is a principal article whatever its language.

Accepted values. A title of an article **present** in the blog database.

Default value: Null[4].

Optional: yes.

Example. If you write a translation of *article about Marco Tullius Cicero*, it could be as follow:

```
:markup: restructuredtext
:language: it
:title: Parlando di Cicerone
:created: 2018-08-05 10:00:00
:modified: 2018-08-05 10:00:00
:slug: parlando-di-cicerone
:summary:  Il profilo di Marco Tullio Cicerone: notizie e vita.
:authors:   Wikipedia
:category: latin literature history
:translation_of: Speaking about Cicero

.. hic sunt leones

Parlando di Cicerone
====================

*Marco Tullio Cicerone* è stato uno statista Romano, oratore, avvocato
e filosofo, che ha servito come console nell'anno 63 AC.
Veniva da una agiata famiglia cittadina dell'ordine Romano degli Equestri,
ed è considerato uno dei più grandi oratori e scrittori di Roma.

...
```

As you can see, in the fields area of this translation, we changed:

- the language indicator, to reflect the new language used in the translation;
- the title (remember: two equal titles aren't possible in the same blog);
- the slug (like above: no equal slugs in the blog, and we would match as near possible the title);
- the summary (maybe it would be read from Italians . . .).

And we added:

- the **translation_of** field, with a value of `Speaking about Cicero`, the title of translated article.

[4] Meaning: it is missing.

1.5.2 published

This is about considering published, or not, the article. Usually `rstblog` regards an article as published by default, unless the article author sets this filed to `no`[5]. An **unpublished** article:

- doesn't compare in indexes;
- doesn't compare in sitemap.xml;
- isn't shown, even if you request it using directly the correct slug in URL.

But it's counted in statistics.

Accepted values: `yes` or `no`.

Default value: `yes`.

Optional: yes.

Example:

```
:published: yes
```

1.5.3 offer_home

`offer_home` is about to show the article in the blog home index.

`rstblog` shows in its home some, usually 20[6], newer articles, checking their creation dates.

If you if you want an article not to be counted between the articles to consider in home, you can set this field to `no`.

Accepted values: `yes` or `no`.

Default value: `yes`.

Optional: yes.

Example:

```
:offer_home: yes
```

1.5.4 insert_author

This is a *control field* . It isn't stored in database. It's value is used from application logic to decide if an author not already existing in database has to be added to the database.

Accepted values: `yes` or `no`.

Default value: `no`.

Optional: yes.

Example:

```
:insert_author: yes
```

[5] The `no` value is meaning. `rstblog` interprets any other value as `yes`.
[6] This value could be modified, but it is an operation to do during the application installation.

1.5.5 insert_category

Another *control field* . This value is used from application logic to decide if a category not already existing in database has to be added to it.

Accepted values: `yes` or `no`.

Default value: `no`.

Optional: yes.

Example:

```
:insert_category: yes
```

1.6 Author manual end

That's all folk about author manual. Thank you to read it. We hope you enjoy it.

CHAPTER 2

manager manual

manager manual

CHAPTER 3

programmer manual

programmer manual table of contents

3.1 load article

This is necessary in two different context:

- while loading a single article
- while rebuilding the table of articles starting from list of article files hosted in server file system

3.1.1 loading a single article

Here we need:

- upload_file from client to server file system
- create (or update) article record in DB from uploaded file

3.1.2 rebuilding the table of articles

We:

- list all the article files in server file system
- cicling on previous list, for every original file:
  - create article record in DB

- cicling again on previous list, for every translation file:
  - create article record in DB

So we can see: there is a function *create article record in DB* used in both the context.

CHAPTER 4

General introduction

This project is developed using Django, a web framework based on the Python language. This fact is important to you *only if you wish to modify how the project work*. In other words if you need to change the programming code of the project.

Central to this work is also the reStructuredText markup language (for short: reST from now on). This is a method to sign a text achieving formatting effects, as writing charactes in bold, or creating a table, or an html link to another document.

And this fact is important to you as a user of this project. Why? we'll see.

Let's assume you have an `rstblog` installation functioning and responding to address `https://my.blog.org` (original, isn't it?). And, of corse, you know a username (& password) authorized to publish an article on it.

Well, now you have a fantastic idea you need to share with mankind.

First think first: you write a vibrant article using your favourite text editor in your computer, formatting it using *reST*[1]. Note: by now your work is saved in a file on your PC.

When you are satisfied of your article, using your web browser, you navigate to `https://my.blog.org/blog/load-article`.

`rstblog` will request your username and password. If you feed them, it will request to you what file you need to load. Browse to it and click the send button. `rstblog` will respond you: *article xxx loaded*[2]

We are done: homepage at `https://my.blog.org` will show the title and summary of your new article. And if you click on title, you'll see your article in all its glory.

Well, there is some little trick to use to obtain the right result. But I swear: it's all very simple. Otherwise I would not be able to use it.

Now. I imagine you are asking yourself: "for God's sake: why I need use a text editor and upload a file to my site, if wordpress or drupal allow me to write it directly in the browser window?".

Due to some reasons.

[1] To be honest, you can also use markdown, or html too. But we are a little fanatic about reST; so: this is a link to a primer to reST.

[2] Or, in case of error, it will prompt you.

First of all: never appened while you are writing, something goes wrong (power failure at your adsl router, a timeout to your server, ...) and do you risk to loose your work? It appened to me. From that moment on, I got used to write long articles in files *before* to feed drupal with them.

But ... this habit lead to different use of the tools. Long articles initially written by files. The short ones written directly using the user interface of the blog, without a local file in your computer.

And when you need to make a little correction to an article, for sure you'll do it using the blog user interface. So your local file in a moment will no longer be aligned with the online version of your article.

So you ask to me: "where is the problem?". The problem is the fact that your articles are stored partially only in the database of your blog software. Partially because probably images, and other files that your refer to into your articles are in the web server file system. So you need to backup all these informations and wish you will not have to change the blog software, or you might encounter some (big) problems to transfer them from one kind of system to the other.

Again: *but* ... if there is a think I know working in IT for over 30 years is that *for sure* your data will have to change system type. Every type of system type :-). Even not only the name of software (e.g. from Drupal to Wordpress), but from type of software to another (e.g. from specialized Blog to general purpouse CMS), or even more complex scenarios.

In these cases, you have more chance of success if your data are in some form of *source* format. The simpler it is, the better it is.

So, returning to us, I decided to experiment to use a simple markup language, as reSt, or markdown, to write a local copy of the articles. And, while I'm living a full copy of them on my PC (this is a local backup, from start!), I upload them to a web server able to host them and catalog their contents by some simple fields written in the article file text.

So, in case of restore in other server I can load my local copy. And in case of a radical software change, I can think to write some (hopefully) simple interface to load my files in a future shocking AI.

If this long introduction did not make you escape, and you are still interested to know better how use `rstblog`, I can propose you these chapters:

- an article *author manual*;
- an **empty** site *manager manual*;
- an **empty** *programmer manual*.

CHAPTER 5

References

This project is open source hosted on GitHub.

And here there is the author's website. Its contents are mainly written in italian language.

This is the launch article about this project.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search