django-rstblog Documentation

Release v0.1

luciano de falco alfano

Aug 24, 2020

Contents:

`django-rstblog` is a Django app to manage a blog, driven by articles written using reStructuredText, or Markdown or HTML.

This is its documentation.

CHAPTER 1

author manual

author manual table of contents

        * *offer_home*
        * *image*
        * *image_in_content*
    - *Author manual end*

Note: A note before to start.

Below *default* and *optional* items are indicative by now. In this version of `django-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 an optional field value at article update, set it explicitly. How? Continue reading :-)

Well, end of note, let's start the work.

This is the scene: you, the author, have availability of a working site that uses `diango-rstblog` to publish your articles. So, someone (a site master) gave you:

- the URL of the site[7];
- your *username*;
- your *password*.

You'll need them to upload the article: keep them safe and begin how to write an article, ... keep going ...

1.1 Article organization

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

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

[7] The site URL will be something of the type: `https://site-domain/blog`.

[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.

```
: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.

...
```

1.2 Article filename

Some speech 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. So, these:

- 159_full_text_search_python.rst
- 160_full_text_search_python.en.rst

could be two files about an article regarding how to do full text search in python. First is in *default* language, the second is in English language.

But whatever rule you'll adopt, it will be right: `django-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 master: 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.

`django-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 (v0.1 as we write) there aren't a lot of controls about syntax errors.

A single field has structure:

> **:fieldname:** *fieldvalue*

`django-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
```

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
```

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

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

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. And only values present in this database are accepted by `rstblog`.

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*.

1.4.1 Link to other files

Let us to give you 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 `django-rstblog` could upload even these files with the article.

1.4.2 Mathematical expressions

In case you need to write mathematical expressions, it's simpler to use Markdown as markup language. At the moment, `django-rstblog` is configured to render math to html from Markdown.

1.4.3 Loading the article content

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

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

`django-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.

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 `django-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:

[3] Or directly type it, if you remember its full path and name.
[4] Meaning: it is missing.

```
: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.

1.5.2 published

This is about considering published, or not, the article. Usually `django-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
```

[5] The `no` value is meaning. `django-rstblog` interprets any other value as `yes`.

1.5.3 offer_home

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

`django-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 image

If you load an image in `contents/media/images`, using this field you can link it from the summary in home page. In practice, you an use an image to characterize the article. A kind of *visual tag*.

Moreover, you can show this image in the main window of the article. You can control this behaviour using the next field: `image_in_content`.

Accepted values: the image filename as a string.

Default value: no.

Optional: yes.

Example:

```
:image: django-logo-negative.svg
```

1.5.5 image_in_content

If you indicate an image as an article characterizer, it is ever showed in the index page. And it is possible to show it even in the window of the article content.

This is the default behavior. If you wish, you can avoid to show this image in the window of the article content, setting to `no` the field `image_in_content`.

This is desiderable if you have an article showing the same image in its contents. In such a case django-rstblog would show this image twice. Not a beautiful behavior.

Accepted values: `yes` or `no`.

Default value: `yes`.

Optional: yes.

Example:

```
:image_in_content: no
```

[6] This value could be modified, but it is an operation to do during the application installation.

1.6 Author manual end

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

CHAPTER 2

Site manager manual

***site manager manual* table of contents**

Ok, so you are the manager of a website that uses `diango-rstblog` to publish articles from an author.

As a site administrator it is expected that you know how to carry out the operations necessary to ensure the operation of the site, from its installation, to routine activities during its operation.

We'll see this operations in the following chapters.

2.1 Installing `diango-rstblog`

2.1.1 Prerequisites

Hereafter we assume you already have a Django project currently running. Probably, even if not necessarly, it is installed using a virtualenv. If this is true, activate it before to do the operations here described.

And we assume that you are going to install in a computer having an Internet connection.

Please, be sure to use a Python version 3.6 or newer, and a Django version 2.0 or newer.

Through the Internet connection, the Python installer (pip) will upload `django-rstblog` and all of its dependencies:

- `docutils`;
- `django-concurrency`;
- `Markdown`;
- `Pygments`;
- `python-markdown-math`.

2.1.2 Installing

First of all, install `diango-rstblog`:

```
pip install django-rstblog
```

Then:

1. In your project `setting.py` file:

    1.1. Add `rstblog` to your INSTALLED_APPS like[1] this:

[1] `django.contrib.sites` and `fullurl` are apps needed to simplify use of `django-rstblog` from the hosting django project. The first one is from Django, the second is the app `django-fullurl`.

```
INSTALLED_APPS = [
    ...
    'django.contrib.sites',       # django's sites framework
    'fullurl',                    # django-fullurl
    ...
    'rstblog',
]
```

1.2. check for presence of login parameters:

```
...
LOGIN_REDIRECT_URL = '/' # It means home view
LOGIN_URL = '/login/'
...
```

1.3. Add a RSTBLOG configuration section like this:

```
...
RSTBLOG = {
    'ARTICLES_DIR': os.path.join(BASE_DIR, "contents", "articles"),
    'START_CONTENT_SIGNAL': '.. hic sunt leones',     # BEWARE: string on␣
↪a single line, without other characters
    'languages': { 'en': 'englis',                   # 1st position is␣
↪default language (functioning on py 3.6+)
                   'it': 'italian', },
    'types': { 'article': 'article',                  # 1st position is␣
↪default type (ok on py 3.6+)
               'page': 'page', },
    'FIELDS': {'markup',
               'image',
               'image_in_content',
               'atype',
               'language',
               'title',
               'created',
               'modified',
               'slug',
               'category',
               'published',
               'offer_home',
               'summary',
               'authors',
               'translation_of', },
    'LIST_FIELDS': {'authors',},
    'DT_FIELDS': { 'created',
                   'modified', },
    'BOOL_FIELDS': { 'published',
                     'offer_home',
                     'image_in_content', },
    'HOME_ITEMS': 10,
}
...
```

1.4 check for presence of SITE_ID:

```
...
SITE_ID = 1
...
```

2. In your project urls.py file:

   2.1. include the `rstblog` URLconf:

```
from django.urls import include
...
path('blog/', include('rstblog.urls', namespace='rstblog')),
...
```

   2.2. check for presence of login url:

```
from django.contrib.auth     import views as auth_views
...
path('login/', auth_views.LoginView.as_view(), name='login'),
path('logout/', auth_views.LogoutView, {'next_page': settings.LOGIN_
↪REDIRECT_URL}, name='logout'),
...
```

3. About your project templates:

   3.1. they must have a `base.html` template with this blocks used from rstblog templates:

```
{% block title %}
{% block meta %}
{% block link %}
{% block content %}
```

   3.2. check for the presence of `templates/registration/login.html` used in login.

4. In your project directory (where live manage.py), create the directory `contents/articles`
5. Run `python manage.py migrate` to create the `django-rstblog` models.
6. Restart the http service and visit `https://your-domain/admin/`[2][3] to create at least a `Category` with value **uncategorized** to load articles[4].
7. Visit `https://your-domain/blog/` to show an empty list of articles.
8. Prepare an article on your PC as this one:

```
:markup:   restructuredtext
:title:    article
:language: en
:slug:     article
:category: uncategorized

.. hic sunt leones

=========
Article
=========

This is the article content.

And this is a secod paragraph of the article.
```

9. Visit `https:/your-domain/blog/load-article` to load the previous article.

[2] Or, you you are in a development environment, start the development server and visit `http://127.0.0.1:8000/admin/`.
[3] You'll need the Admin app enabled and to know an admin account.
[4] Classify the article using an appropriate `category` value is mandatory. An article with a `category` value not present in the database (or without this field at all) will not be uploaded.

10. Now, if you visit again `https://your-domain/blog/` you get a list with an article, and if you click on title, you'll show it (url: https://your-domain/blog/show/article)

2.2 Base architecture

Here we'll spend two words about how `django-rstblog` works.

It uploads articles in a directory as follows[5]:

```
/usr/share/nginx/html/project/site/contents/articles
```

Usually there is at least another directory, to upload other media files:

/usr/share/nginx/html/project/site/contents/media

When a user uploads an article, `django-rstblog` loads it in the aforesaid directory (`.../articles`), then it reads the lines that categorize the article (its fields) and it updates consequently the database tables. If we have a new article: it inserts new records, otherwise it updates the existings ones.

So, user has only one basic operation available: `load-article`, available at URL: `https://your-domain/blog/load-article`

2.2.1 Available URLs

Here are the URLs available by `django-rstblog`, that is its functions:

- `blog/`, it shows the newer articles; allowed to all users, even anonymous;
- `blog/index/`, it shows all the articles; allowed to all users, even anonymous;
- `blog/index/<category>`, it shows all articles of indicated category; allowed to all users, even anonymous;
- `blog/load-article`, it (re)uploads an article; allowed to known users, *not* anonymous;
- **`blog/reset-article-table`, it rebuilds the DB contents from articles loaded in filesystem;** allowed to known users, *not* anonymous;
- `blog/show/<slug>`, it shows the article of indicated slug; allowed to all users, even anonymous;
- `blog/stats`, it shows the django-rstblog statistics; allowed to all users, even anonymous.

2.3 Administering

Note: DB administration is a site functionality, not a `django-rstblog` one.

`django-rstblog` declares its DB structures, so the Django's administration can manage them.

So this function isn't listed between the previous showed URLs, and the request URL is `https://your-domain/admin`, **not** `https://your-domain/blog/admin`.

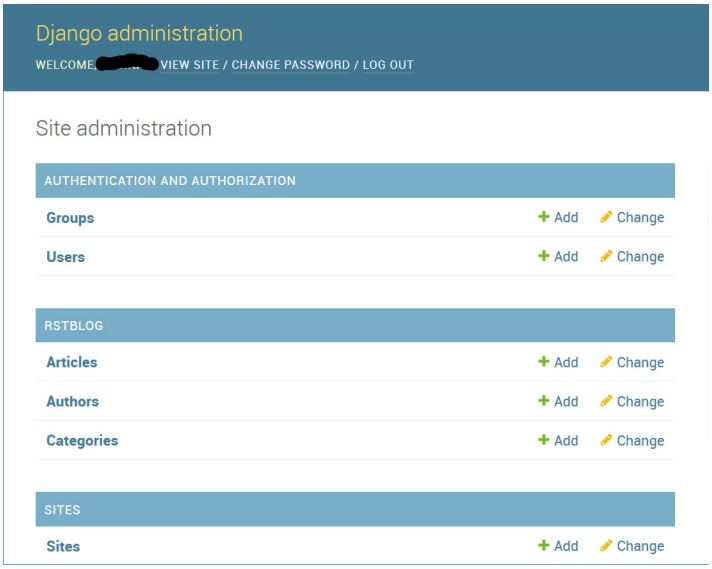
---

[5] It's possible to change it, we'll see how.

2.3.1 Author and Categories

To manage author(s) and categories recognized by `django-rstblog` it's necessary to store their values in the project database.

To do so, call `https://your-domain/admin/`. If you aren't logged in, the site will ask your username and password. Type them (this account must have adminstrative privileges) and you'll get the admin interface to database as follow:



To manage authors and categories you can click on these items showed in the *RSTBLOG* section of the previous image.

User interface to operate this tasks is straightforward, so we don't bother you showing it.

2.3.2 User

An author of articles needs an account to upload articles. So you may have to create one or more (in case of more authors publishing in the same blog) accounts.

You can create accounts using the item *Users* at the section *AUTHENTICATION AND AUTHORIZATION*.

This voice is useful even in case you need to change user's password: edit the user that have requested it.

2.4 Articles administration

2.4.1 Deleting an article

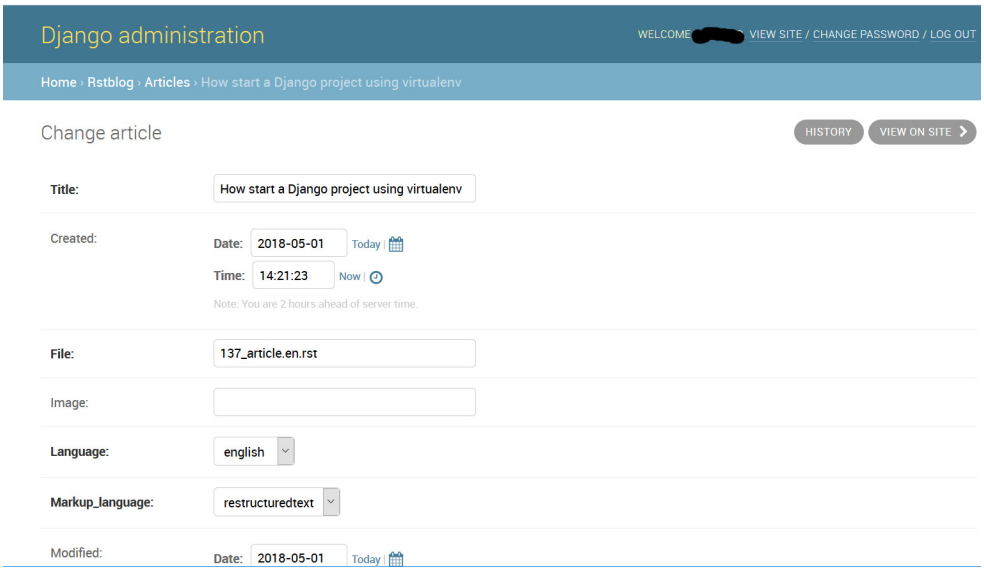
As you can see, delete an article now isn't a function of `django-rstblog`. So if a user wants absolutly to erase an article he/she must ask you to do it, as site master.

How can you do it? With two steps:

- erase the record about the article;
- then delete the relative file in filesystem.

About the first step: be sure to delete the right record. Double check title **and filename** of the article.

If you have only the title and/or the slug, you can retrieve the filename reading it in the record of the article as in the image below (it is the field titled **File**):



About the second step. It's required because in case of *Articles table reconstruction* (see below) the presence of the articles file recreates its record . . .

2.4.2 Changing an article filename

This is another operation not allowed to the author. In case of a request from the author to change an article filename you must:

- change the filename in the record of the article (see the previous image);
- rename the file of the article in the server filesystem;
- tell the author to rename his/her filename in the authoring PC filesystem.

2.4.3 Articles table reconstruction

If needed it is possible force a complete *Articles table* reconstruction.

If you call URL `. . ./blog/reset-article-table`, `django-rstblog` will erase all records in *Articles table*, then it will rebuild it reading all files present in directory `. . ./contents/articles`.

2.5 Configuring

`django-rstblog` uses setup parameters from the django project's `settings.py`.

Notably it reads the dictionary named `RSTBLOG` with this structure:

```
RSTBLOG = {
    'ARTICLES_DIR': os.path.join(BASE_DIR, "contents", "articles"),
    'START_CONTENT_SIGNAL': '.. hic sunt leones',      # BEWARE: string on a single␣
↪line, without other characters
    'languages': { 'en': 'english',                    # 1st position is default␣
↪language (functioning on py 3.6+)
```

(continues on next page)

(continued from previous page)

```
                    'it': 'italian', },
    'types': { 'article': 'article',                    # 1st position is default type␣
↪(ok on py 3.6+)
               'page': 'page', },
    'FIELDS': {'markup',
               'image',
               'atype',
               'language',
               'title',
               'created',
               'modified',
               'slug',
               'category',
               'published',
               'offer_home',
               'summary',
               'authors',
               'translation_of', },
    'LIST_FIELDS': {'authors',},
    'DT_FIELDS': { 'created',
                   'modified', },
    'BOOL_FIELDS': { 'published',
                     'offer_home', },
    'HOME_ITEMS': 10,
}
```

Let's see the parameters in `RSTBLOG`.

2.5.1 ARTICLES_DIR

Directory containing the articles. Usually it is the directory: `project-base-dir/contents/articles`.

> **Warning:** If you wish to change this parameter, test the new value extensively. This is because it is possible have links in article relative from this directory to media directory.
>
> So maybe necessary move the two directories in pairs.

Type: string.

2.5.2 START_CONTENT_SIGNAL

This is the signal used from `django-rstblog` to discern the fields part of the article, from its contents.

You can change it, but keep it homogeneous: you cannot have some articles with one signal and other articles with another one.

Type: string.

2.5.3 languages

The list of the human languages used to write articles. It is a dictionary, and its first introduced key is the default language.

Authors must use *language key* to declare the used language in the article.

language value is displayed in html windows returned from `django-rstblog` to browser.

Note: Insert order in dictionary is assured using Python v.3.6+.

This is the reason that requires the use of Python v.3.6+.

Type: dictionary.

Example:

```
...
'languages': { 'en': 'english',     # 1st pos.is default language (functioning on py
↪3.6+)
               'it': 'italian', },
...
```

2.5.4 types

The list of types managed from `django.rstblog`. This is a **reserved** dictionary, please don't change it.

Value: it must be:

```
...
'types': { 'article': 'article',                # 1st position is default type (ok
↪on py 3.6+)
           'page': 'page', },
```

2.5.5 FIELDS

The list of fields managed from `django.rstblog`. This is a **reserved** set, please don't change it.

Value: it must be:

```
...
'FIELDS': {'markup',
           'image',
           'atype',
           'language',
           'title',
           'created',
           'modified',
           'slug',
           'category',
           'published',
           'offer_home',
           'summary',
           'authors',
           'translation_of', },
```

2.5.6 LIST_FIELDS

The list of fields managed from `django.rstblog` that are fields. This is a **reserved** set, please don't change it.

Value: it must be:

```
...
'LIST_FIELDS': {'authors',},
```

2.5.7 DT_FIELDS

As above about the datetime fields. Again: this is **reserved**, don't alter it.

Value: it must be:

```
...
'DT_FIELDS': { 'created',
               'modified', },
```

2.5.8 BOOL_FIELDS

As above about the boleean fields. Again: this is **reserved**, don't alter it.

Value: it must be:

```
...
'BOOL_FIELDS': { 'published',
                 'offer_home', },
```

2.5.9 HOME_ITEMS

How many items `django.rstblog` shows in its blog home page, keeping the newer articles. You can change it.

Type: integer.

Example:

```
...
'HOME_ITEMS': 10,
```

2.6 Adding a banner in home page

It is possible to add a banner in the home page of the blog site.

To achieve this effect, it is sufficient to load a page titled as `banner`.

It's more complicated to remove the banner. You must delete the `banner` page record from the blog database and remove the relative file from the `contents/pages` directory.

It's strongly suggested to name the banner file with a name such as `banner.html` to avoid to forget it.

CHAPTER 3

General introduction

The basic idea is to adopt a *hybrid* publication model, halfway between a static site (pure html) and a dynamic one (all inside a DB, as Wordpress).

In practice, the author writes his article locally, at his/her PC, then

- he puts a series of lines of text at the top of the article; they serve to categorize it, indicating the language used, the title, and other attributes ...
- and a line of text, of fixed format, which separates the attributes from the article content.

Finally he calls an address (URL) of the site that allows him to upload the article. If the user is not logged in to the site, this address asks for username and password.

When the article is uploaded to the site, `django-rstblog` uses its attributes to classify it in the database. The content of the article is not loaded in the DB; when necessary, it is resumed from the file uploaded on the site.

If the author wants to modify the content of the article (or its attributes), he edits the file on his PC, then upload it again.

CHAPTER 4

Why use `django-rstblog`?

What are the reasons that led us to this design choice? The following:

- we can always count on a local backup of all the contents of the site;
- we can work without an Internet connection, and connect only when we want to upload;
- the program is extremely light, it runs smoothly on servers with limited CPU capacity as with little RAM and HDU space (as long as accesses are contained, and we haven't this problem :-);
- we do not renounce the flexibility and speed of research that a DB allows;
- if we have a few articles[1] the DB can be implemented with the support library of Python (`sqlite3`), without using big programs (in the sense that they commit a lot of resources) as MySQL, or PostgreSQL, ...

[1] Not so few: with hundreds articles, everything reacts well.

CHAPTER 5

Features

The features that the app currently implements are:

- the index of articles, indicating the number of consultations of each article and the main attributes;
- display of an article;
- upload of an article;
- complete reconstruction of the DB starting from the files of the articles uploaded to the site;
- administration of the DB contents using the Django's admin interface; use this interface to:
  - manage a list of authors of the articles;
  - manage a list or arguments to classify the articles (an article must belong to an argument);
- articles may have translations, they can be present in more than one language;
- indication of site statistics; in the sense of how many articles are loaded, how many languages are used, how many articles are present in each classification topic and language.

Note that, at least by now, `django-rstblog` is capable to manage sites with a single blog. It isn't developed to manage multi-blogs sites.

CHAPTER 6

Cons

What are the cons to the use of this environment? You must have a good knowledge of Python/Django:

- to customize the app to your needs;
- to install it in a django project and in a production server.

CHAPTER 7

License

CHAPTER 8

References

This project is hosted on GitHub Here you will find the complete environment needed to develop the `django-rstblog` app. It means: not only the app, but even a minimal django project that hosts it.

If you wish to see a website implemented using this app, you can navigate to the author's website.

And the full documentation is hosted on Read the Docs.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search