BLEXT Documentation

Release 1.0

seagullbird

Sep 04, 2017

User Documentation

BLEXT is a blog website where you can finish the whole "edit - store - publish" blogging process in one place. It was initially designed to provide blog writers **a better blogging experience**, where they can focus on their creation of their masterpieces and anything else (including publishing, loading & storing, theming and etc.) will be neatly done by BLEXT. It is highly recommended to use the supporting markdown desktop editor for a sublime blogging experience.

The code is open source, and available on github.

Notice: The project is still under constructing and this documentation is currently only available for co-workers. If you are interested in contributing, read `README` on github and feel free to contribute or make a contact by email!

CHAPTER 1

User's Guide

This guidance will help you go through BLEXT's features and things you should know quickly.

Blog Format

BLEXT use in-body header to indicate a blog's basic information including title, category and tags. In order to create a blog nicely, it is recommended to follow this format. (You can choose not to but the consequences are not foreseed.)

```
---
title: <title>
category: <category>
tags: [<tag1>(,tag2, ...)]
---
[<summary>
<!-- more -->]
<blog-text>
```

And notice:

1. Content between `---` is called the header in BLEXT and should not be ignored.
2. `summary` is also meant to be created in markdown, and should always be tailed by `<!-- more -->` at the next line.
3. `summary` can be ignored, and if so, ignore `<!-- more -->` as well.
4. There can be as many tags as you want but only one category is needed.

An simple example:

```
---
title: My First BLEXT Blog
category: First
tags: [tag1, tag2]
---
```

```
> This is my first BLEXT Blog!
<!-- more -->
## Hello World
My **first** BLEXT blog is now established.
```

CHAPTER 2

API v1.0

Before everything, I have to claim that this is only a testing API for development usage, and it is not guaranteed without any bugs and may well be updated soon.

Authentication

BLEXT adopts Basic access authentication for API authentication. Include your email and password (or token and an empty string) in http request headers and the server on receiving your request will automatically extract them and try to verify your infomation.

Notice that this version of API doesn't support anonymous users, which means you have to do anything below **after** receiveing a valid authentication. And any serving response is done based on a valid authentication.

Remember, since the http server won't remember any status, **each time an request is made there is an authentication required.** Therefore I strongly recommend you only use *Basic Authentication* once and use *Token Authentication* afterwards.

Basic Authentication

```
$ http --json --auth email:PASSWORD GET https://blext.herokuapp.com/api/v1.0/token
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 163
Content-Type: application/json
Date: Tue, 06 Dec 2016 07:43:46 GMT
Server: gunicorn/18.0
Via: 1.1 vegur

{
    "expiration": 3600,
    "token": "eyJhbGciOiJIUzI1NiIsImlhdCI6MTQ4MTAxMDIyNiwiZXhwIjoxNDgxMDEzODI2fQ.
↪eyJpZCI6MX0.OzzWcW3wvBOb6nTskRuUy-3nnB89bXgtiW8YaAKERiU"
}
```

Token Authentication

Once you've received a token you can use it as the authentication instead of email and password. Just replace `email` with your token and `PASSWORD` with an empty string:

```
$ http --json --auth token: GET/POST <some/other/apis/mentioned/below>
```

However, the token has an expiration time as you can also see from the first response, initialzed to 3600s. Which means you'll have to apply for another valid token after one hour.

Users

To get information based on a valid token or an email & PASSWORD pair.

Basic User information

```
$ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/user/
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 310
Content-Type: application/json
Date: Tue, 06 Dec 2016 08:18:52 GMT
Server: gunicorn/18.0
Via: 1.1 vegur

{
    "avatar_url": "http://...",
    "blog_count": 1,
    "blogs": "http://blext.herokuapp.com/api/v1.0/blogs/",
    "categories": "http://blext.herokuapp.com/api/v1.0/categories",
    "tags": "http://blext.herokuapp.com/api/v1.0/tags",
    "url": "http://blext.herokuapp.com/api/v1.0/user/",
    "username": "username"
}
```

The response in json format represents following information:

Name	Description
avatar_url	The URL for user's avatar.
blog_count	The number of blogs under the user's name.
blogs	The API URL with which you can find all the user's blogs.
categories	The API URL with which you can find all the user's categories.
tags	The API URL with which you can find all the user's tags.
url	The The API URL with which you can the user's information.
username	User's username.

User Categories

```
$ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/categories
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 60
Content-Type: application/json
Date: Tue, 06 Dec 2016 08:49:56 GMT
Server: gunicorn/18.0
Via: 1.1 vegur

{
    "categories": [
        {
            "name": "First"
        }
    ]
}
```

User Tags

```
$ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/tags
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 60
Content-Type: application/json
Date: Tue, 06 Dec 2016 08:49:56 GMT
Server: gunicorn/18.0
Via: 1.1 vegur

{
    "tags": [
        {
            "name": "First"
        }
    ]
}
```

Blogs

Currently based on authentication with blogs API you can: *Get All Blogs*, *Get a Single Blog*, *Get a Blog's Category*, *Get a Blog's Tags*, *Publish a New Blog* and *Update an Existing Blog*.

Get All Blogs

```
$ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/blogs/
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 701
Content-Type: application/json
Date: Tue, 06 Dec 2016 09:02:21 GMT
Server: gunicorn/18.0
```

```
Via: 1.1 vegur

{
    "blogs": [
        {
            "author": "http://blext.herokuapp.com/api/v1.0/user/",
            "body": "<blog body>",
            "category": "http://blext.herokuapp.com/api/v1.0/category/1",
            "draft": false,
            "id": 1,
            "summary_text": "<summary text>",
            "tags": "http://blext.herokuapp.com/api/v1.0/tags/1",
            "timestamp": "Sun, 27 Nov 2016 03:12:45 GMT",
            "title": "<title>",
            "url": "http://blext.herokuapp.com/api/v1.0/blogs/1"
        },

        {
            "author": "http://blext.herokuapp.com/api/v1.0/user/",
            "body": "<blog body>",
            "category": "http://blext.herokuapp.com/api/v1.0/category/2",
            "draft": false,
            "id": 2,
            "summary_text": "<summary text>",
            "tags": "http://blext.herokuapp.com/api/v1.0/tags/2",
            "timestamp": "Sun, 27 Nov 2016 03:12:45 GMT",
            "title": "<title>",
            "url": "http://blext.herokuapp.com/api/v1.0/blogs/2"
        }
    ],
    "count": 2,
    "next": null,
    "prev": null
}
```

Each blog included in the `blogs` list contains the information below.

Name	Type	Description
author	string	The API URL with which you can get the author's information.
body	string	The blog's body, in pure text.
category	string	The API URL with which you can get the blog's category.
draft	boolean	Whether this blog is a draft.
id	int	The blog's id, with which you can get an particular blog.
summary_text	string	The blog's summary, in pure text.
tags	string	The API URL with which you can get the blog's tags.
timestamp	string	The time this blog was initially built.
title	string	The blog's title.
url	string	The API URL with which you can get this blog.

Note that not all the blogs are served at one time if the total amount exceeds a particular number. Instead, blogs are served in pagination. Refer to the URL provided in `prev` and `next` for a new page of blogs if any.

Get a Single Blog

Getting a single blog is as easy as getting them all, except that a blog id must be followed.

```
    $ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/blogs/1
    HTTP/1.1 200 OK
    Connection: keep-alive
    Content-Length: 583
    Content-Type: application/json
    Date: Tue, 06 Dec 2016 09:19:30 GMT
    Server: gunicorn/18.0
    Via: 1.1 vegur

    {
    "author": "http://blext.herokuapp.com/api/v1.0/user/",
    "body": "<blog body>",
    "category": "http://blext.herokuapp.com/api/v1.0/category/1",
    "draft": false,
    "id": 1,
    "summary_text": "<summary text>",
    "tags": "http://blext.herokuapp.com/api/v1.0/tags/1",
    "timestamp": "Sun, 27 Nov 2016 03:12:45 GMT",
    "title": "<title>",
    "url": "http://blext.herokuapp.com/api/v1.0/blogs/1"
}
```

Get a Blog's Category

With a blog id followed.

```
$ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/category/1
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 22
Content-Type: application/json
Date: Tue, 06 Dec 2016 09:22:02 GMT
Server: gunicorn/18.0
Via: 1.1 vegur

{
    "name": "First"
}
```

Get a Blog's Tags

With a blog id followed.

```
$ http --json --auth token: GET https://blext.herokuapp.com/api/v1.0/tags/1
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 87
Content-Type: application/json
Date: Tue, 06 Dec 2016 09:23:19 GMT
Server: gunicorn/18.0
Via: 1.1 vegur

{
    "tags": [
        {
            "name": "first"
        },
        {
            "name": "my"
        }
    ]
}
```

Publish a New Blog

A `POST` method is needed to publish a new blog. On publishing, make sure your data includes both *blog body* and *draft* value indicating whether you want to publish this blog as a draft or not.

```
$ http --json --auth token: POST https://blext.herokuapp.com/api/v1.0/blogs/ \
> "body=<body>" \
> "draft=false"
```

If your *blog body* is properly composed according to the *blog format*, the response will contain the Location of this new blog with a status code 201. Otherwise, an error response will be sent.

Update an Existing Blog

Updating an existing blog is pretty much the same as creating a new one. Except that if an existing blog is to be updated, a blog id should be provided. Besides, you should use `PUT` instead of `POST` to update an existing blog.

```
$ http --json --auth token: PUT https://blext.herokuapp.com/api/v1.0/blogs/1 \
> "body=<body>" \
> "draft=false"
```

And everything else including the response is the same as creating a new blog.