chr Documentation

Release 3.0.8

plausibility

October 23, 2013

CONTENTS

chrso - the leanest (perhaps), meanest (if you look at it right) URL shortener written with Flask in the entire world (maybe...).

CHAPTER

ONE

CONTENTS

1.1 Usage

What good is a URL shortener if it's not shortening URLs? None, I say. **NONE!**

1.1.1 Installation

Installing chr is as easy as following these simple steps:

- Clone the latest repo: `git clone https://github.com/plausibility/chr.git` (then `cd chr`)
- Install the package: `python setup.py install`
- Copy and modify the example script: `cp example.py mychr.py && $EDITOR mychr.py`
- Run chrso: `gunicorn -b 127.0.0.1:5000 -p /tmp/chr.pid mychr:app`

Want to configure your instance (hint: you probably do)? `example.py` has all the essentials.

1.1.2 Running multiple instances

Assuming you're using something like gunicorn (as suggested), you can simply specify a different pid file, and that will handle itself.

If you want to isolate the databases (let's be honest, you probably do), you'll have to specifically set `chrso.redis_namespace` to something specific to each.

Keep this in mind if you ever want to hit up the database to remove or modify URLs, also.

```
# In chr1.py ('gunicorn ... chr1:app')
import chrso
chrso.redis_namespace = "chr1"
from chrso.main import app
other_app_setup()

# In chr2.py ('gunicorn ... chr2:app')
import chrso
chrso.redis_namespace = "chr2"
from chrso.main import app
other_app_setup()
```

1.1.3 Validation

When users submit a URL to be shortened, they're pushed against a rough URL-like regex provided by WTForms.

There's no baked in checks for legitimate domains, `200 OK` replies or anything like that [open an issue on the repo if you're interested and I'll work on it].

1.1.4 Logging

Logging is not implemented as of the `3.x` branch anymore - but there's not much that can happen, so don't fret. (Gunicorn probably provides it, not 100%)

1.1.5 Moderation

At the current time of writing [v3.0.8], there is currently no way to moderate the shortened URLs built into chr.

If you're serious about removing links, or you need to remove a link which has been reported to you, you'll have to use redis-cli.

Say we want to remove `url.example.org/foo` because it's scum:

```
$ redis-cli
# Pull the row id
redis 127.0.0.1:6379> hget chr:id_map foo
"1"
# Ensure it's the URL we're expecting
redis 127.0.0.1:6379> get chr:url:1:long
"https://mallory.example.com/some_scummy.exe"
# Remove it from the id_map
redis 127.0.0.1:6379> hdel chr:id_map 1
(integer) 1
redis 127.0.0.1:6379> exit
$
```

Simply removing a URL from the `id_map` will mean it's not accessible for users, but you will still have the long/short url, related info (IP, useragent) about the submitter, etc.

1.1.6 Clean deployment

If you want to deploy chr somewhere in production (which you.. would if you're reading this) you'll want to look at one of the standard deployment options for Flask.

It's also a nice idea to bind to a unix socket rather than a port. Just tidier:

```
gunicorn -b unix:/tmp/chr.sock -p /tmp/chr.pid example:app
```

nginx

```
upstream chrso {
    server unix:/tmp/chr.sock;
}

server {
    server_name chr.so;
```

```
    location / {
        proxy_pass http://chrso;
    }

    # Let nginx serve static files
    location /static/ {
        # Wherever you installed `chrso`
        root /path/to/chrso;
    }
}
```

lighttpd

TODO!

1.2 Automated Documentation

1.2.1 chrso

chrso - the leanest (perhaps), meanest (if you look at it right) URL shortener written with Flask in the entire world (maybe...).

1.2.2 chrso.main

Note: It's kind of expected for this to be empty - the main file is just Flask related CRUD.

1.2.3 chrso.url

chrso.url.**add**(*long_*, *statistics*, *burn*, *short=None*, *ua=None*, *ip=None*, *ptime=None*, *delete=None*)

Shorten a long URL and add it to our database.

Parameters

- **long** – the long URL we're wishing to shorten
- **statistics** – should we bother enabling statistics for the shortened URL? {0,1} or {True,False} please
- **burn** – should this be a "burn after reading" URL?
- **short** – an optional custom URL
- **ua** – the user-agent of the person the URL was shortened by.
- **ip** – the IP address the URL was shortened by
- **ptime** – the time of the shorten (time.time() is used if omitted)
- **delete** – the deletion key that can be used to remove the url by the user.

Returns False if url add failed, (slug, delete) otherwise

chrso.url.**delete_key**(*ident*)

Grab the key usable for deleting a URL.

Parameters **ident** – an identifier for the URL we want info about.

chrso.url.**exists**(*ident*)
Find out whether the given ident (URL, usually) exists.

Parameters **ident** – an identifier for the URL we want info about.

chrso.url.**hit**(*ident*, *ua=None*, *ip=None*, *ptime=None*)
Add a 'hit' to the database for a given URL. If the given URL is a "burn after reading" url, it will be expunged by this function. If the given URL has statistics turned *off*, this won't do anything unless we're removing a burn after reading URL.

Parameters

- **ident** – an identifier for the URL we want to add a hit to.
- **ua** – the user-agent of the user
- **ip** – the IP of the user
- **ptime** – the time the hit occured (None means current time)

chrso.url.**hits**(*ident*)
Find all the corresponding hits for a given ident.

Parameters **ident** – an identifier for the URL we want to get hits for.

chrso.url.**long**(*ident*)
Get the long URL from a shortened ident.

Parameters **ident** – an identifier for the URL we want info about.

chrso.url.**partial_format**(*part*)
This is just a convenience function so rather than putting schema.thing.format(blah) everywhere, we just schema.thing(blah)

chrso.url.**remove**(*ident*)
Remove a shortened URL from our database.

Parameters **ident** – an identifier for the URL we want to remove.

chrso.url.**row_id**(*ident*)
Get the numerical row id for an ident.

Parameters **ident** – an identifier for the URL we want info about.

chrso.url.**should_burn**(*ident*)
Find out whether or not the given URL is a burn after reading URL.

Parameters **ident** – an identifier for the URL we want info about.

1.2.4 chrso.base62

Note: Taken from the public domain (namely: Stack Overflow), so this isn't my documentation.

Converts any integer into a base [BASE] number. I have chosen 62 as it is meant to represent the integers using all the alphanumeric characters, [no special characters] = {0..9}, {A..Z}, {a..z}

I plan on using this to shorten the representation of possibly long ids, a la url shorteners

saturate() takes the base 62 key, as a string, and turns it back into an integer dehydrate() takes an integer and turns it into the base 62 string

chrso.base62.**dehydrate**(*integer*)
Turn an integer [integer] into a base [BASE] number in string representation

Handles negatives by prefixing with a special character [NEGATIVE_BUFFER] and using the positive.

chrso.base62.**saturate**(*key*)
Turn the base [BASE] number [key] into an integer Handles negatives by returning the negative.

chrso.base62.**true_chr**(*integer*)
Turns an integer [integer] into digit in base [BASE] as a character representation.

chrso.base62.**true_ord**(*char*)
Turns a digit [char] in character representation from the number system with base [BASE] into an integer.

1.2.5 chrso.proxyfix

Borrowed this from werkzeug commits since it's not on PyPI yet: https://github.com/mitsuhiko/werkzeug/commit/cdf680222af293a2c11

class chrso.proxyfix.**ProxyFix**(*app*, *num_proxies=1*)
This middleware can be applied to add HTTP proxy support to an application that was not designed with HTTP proxies in mind. It sets *REMOTE_ADDR*, *HTTP_HOST* from *X-Forwarded* headers.

If you have more than one proxy server in front of your app, set *num_proxies* accordingly.

Do not use this middleware in non-proxy setups for security reasons.

The original values of *REMOTE_ADDR* and *HTTP_HOST* are stored in the WSGI environment as *werkzeug.proxy_fix.orig_remote_addr* and *werkzeug.proxy_fix.orig_http_host*.

Parameters

- **app** – the WSGI application
- **num_proxies** – the number of proxy servers in front of the app.

get_remote_addr(*forwarded_for*)
Selects the new remote addr from the given list of ips in X-Forwarded-For. By default it picks the one that the *num_proxies* proxy server provides. Before 0.9 it would always pick the first. New in version 0.8.

1.3 chr api documentation

Note: This entire thing is TODO! Poke plausibility on GitHub if it's not set up yet.

Note: As of the `3.x` tree, the settings documentation is gone! This is intentional, the doc page for `usage` and `example.py` in the repo have all the information you'll ever need.

PYTHON MODULE INDEX

C