django-revproxy Documentation

Release 0.9.15

Sergio Oliveira

Feb 20, 2019

Contents

1 Features **3**

2 Dependencies **5**

3 Install **7**

4 Contents: **9**

4.1 Introduction . . . 9
4.1.1 How does it work? . . . 9
4.2 Quickstart . . . 10
4.2.1 Installation . . . 10
4.2.2 Configuration . . . 10
4.3 Usage . . . 11
4.3.1 Proxy Views . . . 11
4.4 API . . . 14
4.4.1 revproxy.views . . . 14
4.4.2 revproxy.response . . . 15
4.4.3 revproxy.transformer . . . 15
4.4.4 revproxy.utils . . . 16
4.5 Changelog . . . 17
4.5.1 0.9.15 (2018-05-30) . . . 17
4.5.2 0.9.14 (2018-01-11) . . . 18
4.5.3 0.9.13 (2016-10-31) . . . 18
4.5.4 0.9.12 (2016-05-23) . . . 18
4.5.5 0.9.11 (2016-03-29) . . . 18
4.5.6 0.9.10 (2016-02-03) . . . 18
4.5.7 0.9.9 (2015-12-15) . . . 18
4.5.8 0.9.8 (2015-12-10) . . . 18
4.5.9 0.9.7 (2015-09-17) . . . 18
4.5.10 0.9.6 (2015-09-09) . . . 18
4.5.11 0.9.5 (2015-09-02) . . . 19
4.5.12 0.9.4 (2015-08-27) . . . 19
4.5.13 0.9.3 (2015-06-12) . . . 19
4.5.14 0.9.2 (2015-06-09) . . . 19
4.5.15 0.9.1 (2015-05-18) . . . 19
4.5.16 0.9.0 (2015-03-04) . . . 19

5 Indices and tables 21

Python Module Index 23

A simple reverse proxy using Django. It allows to use Django as a reverse Proxy to HTTP requests. It also allows to use Django as an authentication Proxy.

Documentation available at http://django-revproxy.readthedocs.org/

CHAPTER 1

Features

- Proxies all HTTP methods: HEAD, GET, POST, PUT, DELETE, OPTIONS, TRACE, CONNECT and PATCH
- Copy all http headers sent from the client to the proxied server
- Copy all http headers sent from the proxied server to the client (except hop-by-hop)
- Basic URL rewrite
- Sets the http header REQUEST_USER if the user is logged in Django
- Handles redirects
- Few external dependencies
- Apply XSLT transformation in the response (requires Diazo)

CHAPTER 2

Dependencies

- django >= 1.8
- urllib3 >= 1.12
- diazo >= 1.0.5 (optional)
- lxml >= 3.4, < 3.5 (optional, but diazo dependency)

CHAPTER 3

Install

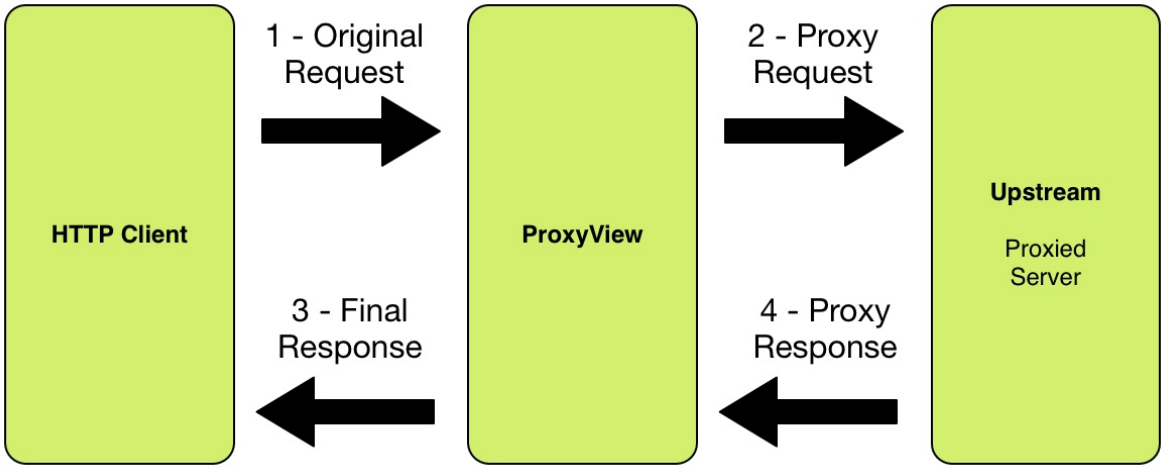
```
pip install django-revproxy
```

CHAPTER 4

Contents:

4.1 Introduction

4.1.1 How does it work?



At a high level, this is what happens behind the scenes in a request proxied by django-revproxy:

1. Django receives a request from the client and process it using a view that extends *revproxy.proxy.ProxyView*.
2. Revproxy will clone the client request.
3. If the user is authenticated in Django and *add_remote_user* attribute is set to *True* the HTTP header *REMOTE_USER* will be set with *request.user.username*.
4. The cloned request is sent to the upstream server (set in the view).
5. After receiving the response from upstream, the view will process it to make sure all headers are set properly. Some headers like *Location* are treated as special cases.

6. The response received from the upstream server is transformed into a *django.http.HttpResponse*. For binary files *StreamingHttpResponse* is used instead to reduce memory usage.
7. If the user has setted a set of diazo rules and a theme template, a diazo/XSLT transformation will be applied on the response body.
8. Finally, the response will then be returned to the user

4.2 Quickstart

4.2.1 Installation

```
$ pip install django-revproxy
```

If you want to use DiazoProxyView you will also need to install Diazo. In that case you can use the following handy shortcurt:

```
$ pip install django-revproxy[diazo]
```

4.2.2 Configuration

After installation, you'll need to configure your application to use django-revproxy. Start by adding revproxy to your `settings.py` file as follows:

```
#Add 'revproxy' to INSTALLED_APPS.
INSTALLED_APPS = (
    # ...
    'django.contrib.auth',
    'revproxy',
    # ...
)
```

Next, you'll need to create a View that extends `revproxy.views.ProxyView` and set the upstrem attribute:

```
from revproxy.views import ProxyView

class TestProxyView(ProxyView):
    upstream = 'http://example.com'
```

And now add your view in the `urls.py`:

```
from myapp.views import TestProxyView

urlpatterns = patterns('',
    url(r'^(?P<path>.*)$', TestProxyView.as_view()),
)
```

Alternatively you could just use the default ProxyView as follow:

```
from revproxy.views import ProxyView

urlpatterns = patterns('',
    url(r'^(?P<path>.*)$', ProxyView.as_view(upstream='http://example.com/')),
)
```

After starting your test server you should see the content of *http://example.com/* on *http://localhost:8000/*.

See also:

An example of a project can be found here: https://github.com/seocam/revproxy-test

The provided test project is a simple Django project that makes uses of revproxy. It basically possess a view.py that extends from ProxyView and sets the upstream address to 'httpbin.org'.

4.3 Usage

4.3.1 Proxy Views

This document covers the views provided by `revproxy.views` and all it's public attributes

class `revproxy.views.`**ProxyView**

Proxies requests to a given upstream server and returns a Django Response.

Example urls.py:

```
from revproxy.views import ProxyView

urlpatterns = patterns('',
    url(r'^(?P<path>.*)$', ProxyView.as_view(upstream='http://example.com/')),
)
```

Attributes

upstream

The URL of the proxied server. Requests will be made to this URL with `path` (extracted from `urls.py`) appended to it. This attribute is mandatory.

add_remote_user

Whether to add the `REMOTE_USER` to the request in case of an authenticated user. Defaults to `False`.

default_content_type

The *Content-Type* that will be added to the response in case the upstream server doesn't send it and if `mimetypes.guess_type` is not able to guess. Defaults to `'application/octet-stream'`.

retries

The max number of attempts for a request. This can also be an instance of `urllib3.Retry`. If set to None it will fail if the first attempt fails. The default value is None.

rewrite

A list of tuples in the style `(from, to)` where `from` must by a valid regex expression and `to` a valid URL. If `request.get_full_path` matches the `from` expression the request will be redirected to `to` with an status code `302`. Matches groups can be used to pass parts from the `from` URL to the `to` URL using numbered groups. By default no rewrite is set.

Example:

```
class CustomProxyView(ProxyView):
    upstream = 'http://www.example.com'
    rewrite = (
        (r'^/yellow/star/$', r'/black/hole/'),
        (r'^/red/?$', r'http://www.mozilla.org/'),

        # Example with numbered match groups
```

(continues on next page)

(continued from previous page)

```
        (r'^/foo/(.*)$', r'/bar\1'),
    )
```

strict_cookies
Whether to only accept RFC-compliant cookies. If set to `True`, any cookies received from the upstream server that do not conform to the RFC will be dropped.

Methods

`ProxyView.`**`get_request_headers`**`()`
Return request headers that will be sent to upstream.

The header REMOTE_USER is set to the current user if AuthenticationMiddleware is enabled and the view's add_remote_user property is True.

New in version 0.9.8.

Extend this method can be particularly useful to add or remove headers from your proxy request. See the example bellow:

```
class CustomProxyView(ProxyView):
    upstream = 'http://www.example.com'

    def get_request_headers(self):
        # Call super to get default headers
        headers = super(CustomProxyView, self).get_request_headers()
        # Add new header
        headers['DNT'] = 1
        return headers
```

class `revproxy.views.`**`DiazoProxyView`**
In addition to ProxyView behavior this view also performs Diazo transformations on the response before sending it back to the original client. Furthermore, it's possible to pass context data to the view thanks to ContextMixin behavior through `get_context_data()` method.

See also:

Diazo is an awesome tool developed by Plone Community to perform XSLT transformations in a simpler way. In order to use all Diazo power please refer to: http://diazo.org/

Example urls.py:

```
from revproxy.views import DiazoProxyView

proxy_view = DiazoProxyView.as_view(
    upstream='http://example.com/',
    html5=True,
    diazo_theme_template='base.html',
)

urlpatterns = patterns('',
    url(r'^(?P<path>.*)$', proxy_view),
)
```

Example base.html

```
<html>
    <head>...</head>
    <body>
```

(continues on next page)

(continued from previous page)

```
        ...
        <div id="content"></div>
        ...
    </body>
</html>
```

Example diazo.xml

```
<rules
    xmlns="http://namespaces.plone.org/diazo"
    xmlns:css="http://namespaces.plone.org/diazo/css"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <!-- Adds 'body' content from example.com into theme #content -->
    <before css:theme-children="#content" css:content-children="body" />
</rules>
```

Attributes

diazo_theme_template
: The Django template to be used as Diazo theme. If set to `None` Diazo will be disabled. By default `diazo.html` will be used.

diazo_rules
: The absolute path for the diazo rules file. By default it will look for the file `diazo.xml` on the Django application directory. If set to `None` Diazo will be disabled.

html5
: By default Diazo changes the doctype for html5 to html4. If this attribute is set to `True` the doctype will be kept. This attribute only works if Diazo transformations are enabled.

Methods

`DiazoProxyView.`**`get_context_data`**`(`***kwargs*`)`
: Extend this method if you need to send context variables to the template before it's used in the proxied response transformation. This method was inherited from ContextMixin.

  New in version 0.9.4.

  See the example bellow:

```
from revproxy.views import DiazoProxyView

class CustomProxyView(DiazoProxyView):
    upstream = 'http://example.com/'
    custom_attribute = 'hello'

    def get_context_data(self, **kwargs):
        context_data = super(CustomProxyView, self).get_context_data(**kwargs)
        context_data.update({'foo': 'bar'})
        return context_data


# urls.py
urlpatterns = patterns('',
    url(r'^(?P<path>.*)$', proxy_view),
)
```

  And than the data will be available in the template as follow:

```
<html>
  <head>...</head>
  <body>
    ...
    <div id="content">
      {{ view.custom_attribute }}
      {{ foo }}
    </div>
    ...
  </body>
</html>
```

4.4 API

4.4.1 revproxy.views

class revproxy.views.**DiazoProxyView**(**args*, ***kwargs*)
Bases: *revproxy.views.ProxyView*, django.views.generic.base.ContextMixin

diazo_rules

diazo_theme_template = 'diazo.html'

dispatch(*request*, *path*)

html5 = False

class revproxy.views.**ProxyView**(**args*, ***kwargs*)
Bases: django.views.generic.base.View

View responsable by excute proxy requests, process and return their responses.

add_remote_user = False

classmethod as_view(***initkwargs*)
Main entry point for a request-response process.

default_content_type = 'application/octet-stream'

dispatch(*request*, *path*)

get_encoded_query_params()
Return encoded query params to be used in proxied request

get_proxy_request_headers(*request*)
Get normalized headers for the upstream

Gets all headers from the original request and normalizes them. Normalization occurs by removing the prefix HTTP_ and replacing and _ by -. Example: HTTP_ACCEPT_ENCODING becames Accept-Encoding.

New in version 0.9.1.

Parameters **request** – The original HTTPRequest instance

Returns Normalized headers for the upstream

get_quoted_path(*path*)
Return quoted path to be used in proxied request

get_request_headers()
Return request headers that will be sent to upstream.

The header REMOTE_USER is set to the current user if AuthenticationMiddleware is enabled and the view's add_remote_user property is True.

New in version 0.9.8.

get_upstream(*path*)

retries = None

rewrite = ()

strict_cookies = False

upstream

4.4.2 revproxy.response

revproxy.response.**DEFAULT_AMT = 65536**
Default number of bytes that are going to be read in a file lecture

revproxy.response.**get_django_response**(*proxy_response*, *strict_cookies=False*)
This method is used to create an appropriate response based on the Content-Length of the proxy_response. If the content is bigger than MIN_STREAMING_LENGTH, which is found on utils.py, than django.http.StreamingHttpResponse will be created, else a django.http.HTTPResponse will be created instead

Parameters

- **proxy_response** – An Instance of urllib3.response.HTTPResponse that will create an appropriate response
- **strict_cookies** – Whether to only accept RFC-compliant cookies

Returns Returns an appropriate response based on the proxy_response content-length

4.4.3 revproxy.transformer

revproxy.transformer.**DIAZO_OFF_REQUEST_HEADER = 'HTTP_X_DIAZO_OFF'**
String used to verify if request has a 'HTTP_X_DIAZO_OFF' header

revproxy.transformer.**DIAZO_OFF_RESPONSE_HEADER = 'X-Diazo-Off'**
String used to verify if request has a 'X-Diazo-Off' header

class revproxy.transformer.**DiazoTransformer**(*request*, *response*)
Bases: object

Class used to make a diazo transformation on a request or a response

reset_headers()
This method remove the header Content-Length entry from the response

set_html5_doctype()
Method used to transform a doctype in to a properly html5 doctype

should_transform()
Determine if we should transform the response

Returns A boolean value

transform(*rules*, *theme_template*, *is_html5*, *context_data=None*)
Method used to make a transformation on the content of the http response based on the rules and theme_templates passed as paremters

Parameters

- **rules** – A file with a set of diazo rules to make a transformation over the original response content
- **theme_template** – A file containing the template used to format the the original response content
- **is_html5** – A boolean parameter to identify a html5 doctype

Returns A response with a content transformed based on the rules and theme_template

revproxy.transformer.**HAS_DIAZO = False**
Variable used to identify if diazo is available

revproxy.transformer.**asbool**(*value*)
Function used to convert certain string values into an appropriated boolean value.If value is not a string the built-in python bool function will be used to convert the passed parameter

Parameters **value** – an object to be converted to a boolean value

Returns A boolean value

revproxy.transformer.**doctype_re = re.compile(b'^<!DOCTYPE\\s[^>]+>\\s*', re.MULTILINE)**
Regex used to find the doctype-header in a html content

4.4.4 revproxy.utils

revproxy.utils.**DEFAULT_CHARSET = 'latin-1'**
Variable that represent the default charset used

revproxy.utils.**HTML_CONTENT_TYPES = ('text/html', 'application/xhtml+xml')**
List containing string constants that represents possible html content type

revproxy.utils.**IGNORE_HEADERS = ('HTTP_ACCEPT_ENCODING', 'HTTP_HOST', 'HTTP_REMOTE_USER')**
List containing string constant that are used to represent headers that can be ignored in the required_header function

revproxy.utils.**MIN_STREAMING_LENGTH = 4096**
Variable used to represent a minimal content size required for response to be turned into stream

revproxy.utils.**cookie_from_string**(*cookie_string*, *strict_cookies=False*)
Parser for HTTP header set-cookie The return from this function will be used as parameters for django's response.set_cookie method. Because set_cookie doesn't have parameter comment, this cookie attribute will be ignored.

Parameters

- **cookie_string** – A string representing a valid cookie
- **strict_cookies** – Whether to only accept RFC-compliant cookies

Returns A dictionary containing the cookie_string attributes

revproxy.utils.**encode_items**(*items*)
Function that encode all elements in the list of items passed as a parameter

Parameters **items** – A list of tuple

Returns A list of tuple with all items encoded in 'utf-8'

revproxy.utils.**get_charset**(*content_type*)
Function used to retrieve the charset from a content-type.If there is no charset in the content type then the charset defined on DEFAULT_CHARSET will be returned

Parameters **content_type** – A string containing a Content-Type header

Returns A string containing the charset

revproxy.utils.**is_html_content_type**(*content_type*)
Function used to verify if the parameter is a proper html content type

Parameters **content_type** – String variable that represent a content-type

Returns A boolean value stating if the content_type is a valid html content type

revproxy.utils.**normalize_request_headers**(*request*)
Function used to transform header, replacing 'HTTP_' to '' and replace '_' to '-'

Parameters **request** – A HttpRequest that will be transformed

Returns A dictionary with the normalized headers

revproxy.utils.**required_header**(*header*)
Function that verify if the header parameter is a essential header

Parameters **header** – A string represented a header

Returns A boolean value that represent if the header is required

revproxy.utils.**set_response_headers**(*response*, *response_headers*)

revproxy.utils.**should_stream**(*proxy_response*)
Function to verify if the proxy_response must be converted into a stream.This will be done by checking the proxy_response content-length and verify if its length is bigger than one stipulated by MIN_STREAMING_LENGTH.

Parameters **proxy_response** – An Instance of urllib3.response.HTTPResponse

Returns A boolean stating if the proxy_response should be treated as a stream

revproxy.utils.**unquote**(*value*)
Remove wrapping quotes from a string.

Parameters **value** – A string that might be wrapped in double quotes, such as a HTTP cookie value.

Returns Beginning and ending quotes removed and escaped quotes (") unescaped

4.5 Changelog

4.5.1 0.9.15 (2018-05-30)

- Fix issues with latest urllib3. Fixes #75.
- Fix issues with parsing cookies. Fixes #84.
- Drop Python 3.3, 3.4, and PyPy support.
- Add Python 3.6 support.

4.5.2 0.9.14 (2018-01-11)

- Move construction of proxied path to method [@dimrozakis]
- User.get_username() rather than User.name to support custom User models [@acordiner]

4.5.3 0.9.13 (2016-10-31)

- Added support to Django 1.10 (support to 1.7 was dropped)

4.5.4 0.9.12 (2016-05-23)

- Fixed error 500 caused by content with wrong encoding [@lucaskanashiro, @macartur]

4.5.5 0.9.11 (2016-03-29)

- Updated urllib3 to 1.12 (at least)

4.5.6 0.9.10 (2016-02-03)

- Fixed Python 3 compatibility issue (see #59 and #61). Thanks @stefanklug and @macro1!

4.5.7 0.9.9 (2015-12-15)

- Reorder header prior to httplib request. *Host* should be always the first

request header.

4.5.8 0.9.8 (2015-12-10)

- Added support to Django 1.9 (dropped support to Django 1.6)
- Added *get_request_headers* to make easier to set and override request headers

4.5.9 0.9.7 (2015-09-17)

- Bug fixed: property preventing to set upstream and diazo_rules (#53, #54) [@vdemin]
- Security issue fixed: when colon is present at URL path urljoin ignores the upstream and the request is redirected to the path itself allowing content injection

4.5.10 0.9.6 (2015-09-09)

- Fixed connections pools
- Use wsgiref to check for hop-by-hop headers [#50]
- Refactored tests
- Fixed security issue that allowed remote-user header injection

4.5.11 0.9.5 (2015-09-02)

- Added extras_require to make easier diazo installation

4.5.12 0.9.4 (2015-08-27)

- Alow to send context dict to transformation template. [@chaws, @macartur]

4.5.13 0.9.3 (2015-06-12)

- Use StringIO intead of BytesIO on theme compilation (transformation)

4.5.14 0.9.2 (2015-06-09)

Thanks @rafamanzo for the reports.

- Append a backslash on upstream when needed
- Validate upstream URL to make sure it has a scheme
- Added branch test coverage

4.5.15 0.9.1 (2015-05-18)

- More permissive URL scheme (#41).
- Refactored code to allow setting custom headers by extending method (#40) [@marciomazza]

4.5.16 0.9.0 (2015-03-04)

- urllib2 replaced by urllib3 (#10)
- No Diazo transformation if header X-Diazo-Off is set to true - either request or response (#15)
- Removed double memory usage when reading response body (#16)
- Fixed bug caused by many set-cookies coming from upstream (#23) - by @thiagovsk and @macartur
- Added stream support for serving big files with an acceptable memory footprint (#17 and #24). Thanks to @lucasmoura, @macartur, @carloshfoliveira and @thiagovsk.
- Moved Diazo functionalities to DiazoProxyView.
- Logging improved (#21).
- Added options for default_content_type and retries [@gldnspud].
- Sphinx docs (#25).
- 100% test coverage.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

revproxy.response, 15
revproxy.transformer, 15
revproxy.utils, 16
revproxy.views, 14

Index

A

add_remote_user (revproxy.views.ProxyView attribute), 11, 14
as_view() (revproxy.views.ProxyView class method), 14
asbool() (in module revproxy.transformer), 16

C

cookie_from_string() (in module revproxy.utils), 16

D

DEFAULT_AMT (in module revproxy.response), 15
DEFAULT_CHARSET (in module revproxy.utils), 16
default_content_type (revproxy.views.ProxyView attribute), 11, 14
DIAZO_OFF_REQUEST_HEADER (in module revproxy.transformer), 15
DIAZO_OFF_RESPONSE_HEADER (in module revproxy.transformer), 15
diazo_rules (revproxy.views.DiazoProxyView attribute), 13, 14
diazo_theme_template (revproxy.views.DiazoProxyView attribute), 13, 14
DiazoProxyView (class in revproxy.views), 14
DiazoTransformer (class in revproxy.transformer), 15
dispatch() (revproxy.views.DiazoProxyView method), 14
dispatch() (revproxy.views.ProxyView method), 14
doctype_re (in module revproxy.transformer), 16

E

encode_items() (in module revproxy.utils), 16

G

get_charset() (in module revproxy.utils), 17
get_context_data() (revproxy.views.revproxy.views.DiazoProxyView.DiazoProxyView method), 13
get_django_response() (in module revproxy.response), 15
get_encoded_query_params() (revproxy.views.ProxyView method), 14
get_proxy_request_headers() (revproxy.views.ProxyView method), 14
get_quoted_path() (revproxy.views.ProxyView method), 14
get_request_headers() (revproxy.views.ProxyView method), 14
get_request_headers() (revproxy.views.revproxy.views.ProxyView.ProxyVie method), 12
get_upstream() (revproxy.views.ProxyView method), 15

H

HAS_DIAZO (in module revproxy.transformer), 16
html5 (revproxy.views.DiazoProxyView attribute), 13, 14
HTML_CONTENT_TYPES (in module revproxy.utils), 16

I

IGNORE_HEADERS (in module revproxy.utils), 16
is_html_content_type() (in module revproxy.utils), 17

M

MIN_STREAMING_LENGTH (in module revproxy.utils), 16

N

normalize_request_headers() (in module revproxy.utils), 17

P

ProxyView (class in revproxy.views), 14

R

required_header() (in module revproxy.utils), 17
reset_headers() (revproxy.transformer.DiazoTransformer method), 15
retries (revproxy.views.ProxyView attribute), 11, 15
revproxy.response (module), 15
revproxy.transformer (module), 15
revproxy.utils (module), 16

revproxy.views (module), 14
revproxy.views.DiazoProxyView (built-in class), 12
revproxy.views.ProxyView (built-in class), 11
rewrite (revproxy.views.ProxyView attribute), 11, 15

S

set_html5_doctype() (revproxy.transformer.DiazoTransformer method), 15
set_response_headers() (in module revproxy.utils), 17
should_stream() (in module revproxy.utils), 17
should_transform() (revproxy.transformer.DiazoTransformer method), 15
strict_cookies (revproxy.views.ProxyView attribute), 12, 15

T

transform() (revproxy.transformer.DiazoTransformer method), 15

U

unquote() (in module revproxy.utils), 17
upstream (revproxy.views.ProxyView attribute), 11, 15