djsubdomains Documentation

Release 2.1.0

ted kaemming

May 04, 2016

Contents

1 Installation **3**

2 Quick Start **5**
2.1 Example Configuration 5

3 Basic Usage **7**
3.1 Using Subdomains in Views 7
3.2 Resolving Named URLs by Subdomain 7
3.3 Resolving Named URLs in Templates 8

4 API Reference **9**
4.1 subdomains.middleware 9
4.2 subdomains.templatetags.subdomainurls 9
4.3 subdomains.utils 10

5 Indices and tables **11**

Python Module Index **13**

Subdomain helpers for the Django framework, including subdomain-based URL routing and reversing.

CHAPTER 1

Installation

This application is available via the Python Package Index and can be installed with any Python package manager, such as `pip` or `easy_install` by running:

```
pip install djsubdomains
```

or:

```
easy_install djsubdomains
```

It is highly recommended to use package version numbers when using this project as a dependency to ensure API consistency.

To install the latest version from the repository source, clone the repository and then run `make install` in the repository directory.

CHAPTER 2

Quick Start

To set up subdomain URL routing and reversing in a Django project:

1. Add *subdomains.middleware.SubdomainURLRoutingMiddleware* to your MIDDLEWARE_CLASSES in your Django settings file. If you are using django.middleware.common.CommonMiddleware, the subdomain middleware should come before CommonMiddleware.
2. Configure your SUBDOMAIN_URLCONFS dictionary in your Django settings file.
3. Configure your BASE_DOMAIN dictionary in your Django settings file.
4. If you want to use the subdomain-based {% url %} template tag, add subdomains to your INSTALLED_APPS.

2.1 Example Configuration

```
# This is the urlconf that will be used for any subdomain that is not
# listed in ``SUBDOMAIN_URLCONFS``, or if the HTTP ``Host`` header does not
# contain the correct domain.
# If you're planning on using wildcard subdomains, this should correspond
# to the urlconf that will be used for the wildcard subdomain. For example,
# 'accountname.mysite.com' will load the ROOT_URLCONF, since it is not
# defined in ``SUBDOMAIN_URLCONFS``.
ROOT_URLCONF = 'myproject.urls.account'

# A dictionary of urlconf module paths, keyed by their subdomain.
SUBDOMAIN_URLCONFS = {
    None: 'myproject.urls.frontend',  # no subdomain, e.g. ``example.com``
    'www': 'myproject.urls.frontend',
    'api': 'myproject.urls.api',
}
```

CHAPTER 3

Basic Usage

3.1 Using Subdomains in Views

On each request, a `subdomain` attribute will be added to the `request` object. You can use this attribute to effect view logic, like in this example:

```
def user_profile(request):
    try:
        # Retrieve the user account associated with the current subdomain.
        user = User.objects.get(username=request.subdomain)
    except User.DoesNotExist:
        # No user matches the current subdomain, so return a generic 404.
        raise Http404
```

3.2 Resolving Named URLs by Subdomain

Included is a *`subdomains.utils.reverse()`* function that responds similarly to `django.core.urlresolvers.reverse()`, but accepts optional `subdomain` and `scheme` arguments and does not allow a `urlconf` parameter.

If no `subdomain` argument is provided, the URL will be resolved relative to the `SUBDOMAIN_URLCONFS[None]` or `ROOT_URLCONF`, in order. The protocol scheme is the value of `settings.DEFAULT_URL_SCHEME`, or if unset, `http`:

```
>>> from subdomains.utils import reverse
>>> reverse('home')
'http://example.com/'
>>> reverse('user-profile', kwargs={'username': 'ted'})
'http://example.com/users/ted/'
>>> reverse('home', scheme='https')
'https://example.com/'
```

For subdomains, the URL will be resolved relative to the `SUBDOMAIN_URLCONFS[subdomain]` value if it exists, otherwise falling back to the `ROOT_URLCONF`:

```
>>> from subdomains.utils import reverse
>>> reverse('home', subdomain='api')
'http://api.example.com/'
>>> reverse('home', subdomain='wildcard')
'http://wildcard.example.com/'
```

```
>>> reverse('login', subdomain='wildcard')
'http://wildcard.example.com/login/'
```

If a URL cannot be resolved, a `django.core.urlresolvers.NoReverseMatch` will be raised.

3.3 Resolving Named URLs in Templates

The `subdomainurls` template tag library contains a `url` tag that takes an optional `subdomain` argument as it's first positional argument, or as named argument. The following are all valid invocations of the tag:

```
{% load subdomainurls %}
{% url 'home' %}
{% url 'home' 'subdomain' %}
{% url 'home' subdomain='subdomain' %}
{% url 'user-profile' username='ted' %}
{% url 'user-profile' subdomain='subdomain' username='ted' %}
```

If `request` is in the template context when rendering and no subdomain is provided, the URL will be attempt to be resolved by relative to the current subdomain. If no request is available, the URL will be resolved using the same rules as a call to *`subdomains.utils.reverse()`* without a `subdomain` argument value. An easy way to ensure this functionality is available is to add `django.core.context_processors.request()` is in your `settings.TEMPLATE_CONTEXT_PROCESSORS` list.

Note: For implementation simplicity, this template tag only supports the Django 1.5 `{% url %}` syntax with variable URL names. For more information, please see the reference documentation for *`url()`*.

CHAPTER 4

API Reference

4.1 subdomains.middleware

class subdomains.middleware.**SubdomainMiddleware**
A middleware class that adds a subdomain attribute to the current request.

get_domain_for_request(*request*)
Returns the domain that will be used to identify the subdomain part for this request.

process_request(*request*)
Adds a subdomain attribute to the request parameter.

class subdomains.middleware.**SubdomainURLRoutingMiddleware**
A middleware class that allows for subdomain-based URL routing.

process_request(*request*)
Sets the current request's urlconf attribute to the urlconf associated with the subdomain, if it is listed in settings.SUBDOMAIN_URLCONFS.

process_response(*request*, *response*)
Forces the HTTP Vary header onto requests to avoid having responses cached across subdomains.

4.2 subdomains.templatetags.subdomainurls

subdomains.templatetags.subdomainurls.**url**(*context*, *view*, *subdomain=<object object>*, **args*, ***kwargs*)
Resolves a URL in a template, using subdomain-based URL resolution.

If no subdomain is provided and a request is in the template context when rendering, the URL will be resolved relative to the current request's subdomain. If no request is provided, the URL will be resolved relative to current domain with the settings.ROOT_URLCONF.

Usage:

```
{% load subdomainurls %}
{% url 'view-name' subdomain='subdomain' %}
```

4.3 subdomains.utils

subdomains.utils.**urljoin**(*domain*, *path=None*, *scheme=None*)
Joins a domain, path and scheme part together, returning a full URL.

Parameters

- **domain** – the domain, e.g. `example.com`
- **path** – the path part of the URL, e.g. `/example/`
- **scheme** – the scheme part of the URL, e.g. `http`, defaulting to the value of `settings.DEFAULT_URL_SCHEME`

Returns a full URL

subdomains.utils.**reverse**(*viewname*, *subdomain=None*, *scheme=None*, *args=None*, *kwargs=None*, *current_app=None*)
Reverses a URL from the given parameters, in a similar fashion to `django.core.urlresolvers.reverse()`.

Parameters

- **viewname** – the name of URL
- **subdomain** – the subdomain to use for URL reversing
- **scheme** – the scheme to use when generating the full URL
- **args** – positional arguments used for URL reversing
- **kwargs** – named arguments used for URL reversing
- **current_app** – hint for the currently executing application

subdomains.utils.**insecure_reverse** = <functools.partial object>
reverse() bound to insecure (non-HTTPS) URLs scheme

subdomains.utils.**secure_reverse** = <functools.partial object>
reverse() bound to secure (HTTPS) URLs scheme

subdomains.utils.**relative_reverse** = <functools.partial object>
reverse() bound to be relative to the current scheme

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

S

subdomains.middleware, 9
subdomains.templatetags.subdomainurls, 9
subdomains.utils, 10

G

get_domain_for_request() (subdomains.middleware.SubdomainMiddleware method), 9

I

insecure_reverse (in module subdomains.utils), 10

P

process_request() (subdomains.middleware.SubdomainMiddleware method), 9
process_request() (subdomains.middleware.SubdomainURLRoutingMiddleware method), 9
process_response() (subdomains.middleware.SubdomainURLRoutingMiddleware method), 9

R

relative_reverse (in module subdomains.utils), 10
reverse() (in module subdomains.utils), 10

S

secure_reverse (in module subdomains.utils), 10
SubdomainMiddleware (class in subdomains.middleware), 9
subdomains.middleware (module), 9
subdomains.templatetags.subdomainurls (module), 9
subdomains.utils (module), 10
SubdomainURLRoutingMiddleware (class in subdomains.middleware), 9

U

url() (in module subdomains.templatetags.subdomainurls), 9
urljoin() (in module subdomains.utils), 10