django-disqus Documentation

Release 0.5

Arthur Koziel

Mar 30, 2017

Contents

Export comments and integrate DISQUS into your Django website.

Features:

- Export `django.contrib.comments` to DISQUS
- Dump comments from DISQUS into a local JSON file
- Exporting Comments as WXR
- Templatetags

django-disqus is available open-source under the BSD license. If you'd like to help out, you can fork the project at http://github.com/arthurk/django-disqus and report any bugs at http://github.com/arthurk/django-disqus/issues.

CHAPTER 1

Contents

Installation

The easiest way to get django-disqus is if you have pip installed:

```
pip install django-disqus
```

Without pip, it's still pretty easy. Download the django-disqus.tar.gz file from django-disqus' PyPI page, untar it and run:

```
python setup.py install
```

Configuring your Django installation

First, add `disqus` to your `INSTALLED_APPS`. You **don't** need to run `syncdb` as there are no models provided.

Next, add `DISQUS_API_KEY` and `DISQUS_WEBSITE_SHORTNAME` to your settings. You can get your API key here (you must be logged in on the DISQUS website). To see the shortname of your website, navigate to Settings->General on the DISQUS website.

Example settings.py:

```
INSTALLED_APPS = (
     ...
     'disqus',
 )

 DISQUS_API_KEY = 'FOOBARFOOBARFOOBARFOOBARFOOBARF'
 DISQUS_WEBSITE_SHORTNAME = 'foobar'
```

Finally, you need to change the domain of your Site to the domain you're actually going to use for your website. The easiest way to do this is to enable *django.contrib.admin* and just click on the *Site* object to modify it. If you don't have contrib.admin installed (or don't want to install it), you can run *python manage.py shell* and change the value in the cli:

```
>>> from django.contrib.sites.models import Site
>>> Site.objects.all()
[<Site: example.org>]
>>> s = Site.objects.all()[0]
>>> s.domain = 'arthurkoziel.com'
>>> s.name = 'arthurkoziel.com'
>>> s.save()
>>> Site.objects.all()
[<Site: arthurkoziel.com>]
```

Templatetags

Before you can use the template tags, you need to load them with `{% load disqus_tags %}`.

disqus_dev

Return the HTML/Javascript code to enable DISQUS comments on a local development server. This template tag will only return a value if the `settings.DEBUG` setting is set to `True`. If you don't include this, the comment form will not show up on a local development server.

Example:

```
{% load disqus_tags %}
{% disqus_dev %}
```

Result:

```
<script type="text/javascript">
  var disqus_developer = 1;
  var disqus_url = 'http://arthurkoziel.com/';
</script>
```

disqus_show_comments

Renders the `disqus/show_comments.html` template to display DISQUS comments, including any configuration variables set in this template block. The comments for the current Thread and the comment form are displayed to the user. See the embed code for more information.

Example:

```
{% load disqus_tags %}
{% disqus_show_comments %}
```

Result:

```
<div id="disqus_thread"></div>
<script type="text/javascript" src="http://disqus.com/forums/arthurkozielsblog/embed.
↪js"></script>
<noscript><p><a href="http://arthurkozielsblog.disqus.com/?url=ref">View the␣
↪discussion thread.</a></p></noscript>
<p><a href="http://disqus.com" class="dsq-brlink">blog comments powered by <span␣
↪class="logo-disqus">Disqus</span></a></p>
```

Options:

- `shortname`: DISQUS website shortname that should be used. The `settings.DISQUS_WEBSITE_SHORTNAME` setting takes precedence over this parameter. Example: `{% disqus_show_comments "foobar" %}`

disqus_recent_comments

Renders the `disqus/recent_comments.html` template to display a certail number of recent DISQUS comments on your site, including any configuration variables set in this template block. See the embed code for more information.

Example:

```
{% load disqus_tags %}
{% disqus_recent_comments shortname num_items excerpt_length hide_avatars avatar_size
↪%}

{% disqus_recent_comments shortname 5 50 0 24 %} - will show 5 comments, truncated to␣
↪50 symbols
with avatars 24x24px
{% disqus_recent_comments shortname 10 50 1 %} - show 10 comments, truncated to 50␣
↪symbols
without avatars
```

Options:

- `shortname`: DISQUS website shortname that should be used. The `settings.DISQUS_WEBSITE_SHORTNAME` setting takes precedence over this parameter.
- `num_items`: How many comments to show(default is to show 5 comments)
- `excerpt_length`: Truncate length of comment to a certain number(default is 200 symbols)
- `hide_avatars`: Whether to show avatars or not(default is to show)
- `avatar_size`: Size (in px) of avatar (default 32x32)

disqus_num_replies

Renders the `disqus/num_replies.html` template, including any configuration variables set in this template block. This code that transforms links which end with a `#disqus_thread` anchor into the thread's comment count.

Disqus recommends including a `data-disqus-identifier` parameter to the `<a>` tag for consistent lookup. Make sure you also use *set_disqus_identifier* on the page it links to, as well.

Example:

```
{% load disqus_tags %}
<a href="{{ object.get_absolute_url }}#disqus_thread" data-disqus-identifier="{{␣
↪object.id }}">View Comments</a>
{% disqus_num_replies %}
```

Template Tag output:

```
<script type="text/javascript">
...
</script>
```

The javascript will then transform the link to:

```
<a href="foobar/">2 Comments</a>
```

Options:

- `shortname`: DISQUS website shortname that should be used. The `settings.DISQUS_WEBSITE_SHORTNAME` setting takes precedence over this parameter. Example: `{% disqus_num_replies "foobar" %}`

set_disqus_developer

Adds `disqus_developer` variable to the context for the current block. The context variable is used in the *disqus_show_comments* and *disqus_num_replies* templatetags for signaling Disqus you are in testing mode. See JavaScript configuration vairables documentation for more information.

Example:

```
{% load disqus_tags %}
{% set_disqus_developer 1 %}
```

set_disqus_identifier

Adds `disqus_identifier` variable to the context for the current block. The context variable is used in the *disqus_show_comments* and *disqus_num_replies* templatetags to assign a unique value for this page. The value can be a static value or a variable. See JavaScript configuration vairables documentation for more information.

Example:

```
{% load disqus_tags %}
{% set_disqus_identifier object.id %}
```

You may also pass in multiple arguments, which will then be concatenated:

```
{% load disqus_tags %}
{% set_disqus_identifier "blogentry_" object.id %}
```

This results in `disqus_identifier` set to `blogentry_25`, if the object's id is 25.

set_disqus_url

Adds `disqus_url` variable to the context for the current block. The context variable is used in the *disqus_show_comments* and *disqus_num_replies* templatetags to assign a the URL for this page. This is very important if there are several ways to reach this page (mobile and desktop versions, for example). The value can be a static value or a variable. See JavaScript configuration vairables documentation for more information.

Example:

```
{% load disqus_tags %}
{% set_disqus_url object.get_absolute_url %}
```

set_disqus_title

Adds `disqus_title` variable to the context for the current block. The context variable is used in the *disqus_show_comments* and *disqus_num_replies* templatetags to assign a title for this page. If your `<title>` tag contains extra cruft, this is useful for setting a easier to read title. The value can be a static value or a variable. See JavaScript configuration vairables documentation for more information.

Example:

```
{% load disqus_tags %}
{% set_disqus_title object.headline %}
```

Commands

django-disqus provides the following management commands.

disqus_dumpdata

Outputs a list of comments in the JSON format.

If neither of the `--filter` or `--exclude` options are used, the output will include approved, deleted and spam comments. Each comment will have the data about its associated Author, Thread and Forum included.

Example output:

```
[{
    "status": "approved",
    "has_been_moderated": false,
    "thread": {
        "category": "78805",
        "allow_comments": true,
        "forum": "71225",
        "title": "Passing MEDIA_URL in Django&#39;s 500 error view",
        "url": "http://arthurkoziel.com/2009/01/15/passing-mediaurl-djangos-500-error-
↪view/",
        "created_at": "2009-01-17T17:29",
        "slug": "passing_media_url_in_django39s_500_error_view_arthur_koziels_blog",
        "hidden": false,
        "identifier": [],
        "id": "102172011"
    },
    "forum": {
        "id": "71225",
        "created_at": "2009-01-17 05:48:00.863075",
        "shortname": "arthurkozielsblog",
        "name": "Arthur Koziel\u2019s Blog",
        "description": ""
    },
    "created_at": "2009-11-30T12:48",
    "is_anonymous": true,
    "points": 0,
    "message": "Thanks for the article!",
    "anonymous_author": {
        "url": "http://example.org/",
        "email_hash": "j198m7123m12837m12893m7128121u23",
        "name": "John",
        "email": "john@example.org"
    },
    "ip_address": "12.345.678.11",
    "id": "12345678",
    "parent_post": null
}]
```

Options:

- `--indent`: Specifies the indent level to use when pretty-printing output. Example: `./manage.py dumpdata --indent=4`
- `--filter`: Type of entries (approved, spam, killed) that should be returned. Types can be combined by separating them with a comma. Example: `./manage.py dumpdata --filter=spam,killed`
- `--exclude`: Type of entries (approved, spam, killed) that should be excluded. Types can be combined by separating them with a comma. Example: `./manage.py dumpdata --exclude=spam,killed`

disqus_export

Export comments from contrib.comments to DISQUS.

Before you run this command, make sure that `django.contrib.comments` and `django.contrib.sites` are listed in your project's `INSTALLED_APPS`. You also need to change the domain of your Site from `example.org` to your real domain.

The comment's associated content object must have the following two methods:

- `get_absolute_url`: Should return the URL of an object. For example: `/2009/10/10/foo`. This should not include the domain name
- `__unicode__`: Unicode representation of the object

The command will export all comments that have the `is_public` attribute set to `True` and `is_removed` set to `False`. To test which comments will be exported, you can pass the `--dry-run` option.

Options:

- `-d`/`--dry-run`: Does not export any comments, but merely outputs the comments which would have been exported. Example: `./manage.py disqus_export --dry-run`
- `-v`/`--verbosity`: Specify the amount of information that should be printed to the console. A verbosity of `0` will output nothing. The default verbosity is `1` and print the title of the comments that are exported. Example: `./manage.py disqus_export --verbosity=0`
- `-s`/`--state-file`: Specify the filepath where the export command should save its state (the id of the last exported comment) into. This makes it possible to resume interrupted exports.

Exporting Comments as WXR

The WXR feed is an XML document that contains the item upon which people commented as well as the comments. The Disqus WXR feed is a superset of the the typical RSS feed, and therefore works very much like a typical Django syndication feed.

Create a ContribCommentsWxrFeed Subclass

`ContribCommentsWxrFeed` exports `django.contrib.comments` for a set of items. This example exports comments to entries of a fictional blogging app.

```
from disqus.wxr_feed import ContribCommentsWxrFeed
from coolblog.models import Entry

class EntryWxrFeed(ContribCommentsWxrFeed):
    link = "/"

    def items(self):
        return Entry.objects.all()

    def item_pubdate(self, item):
        return item.pub_date

    def item_description(self, item):
        return item.content
```

All WxrFeed Attributes

For a full explanation of how you can define these attributes, see Django's syndication documentation.

title_template or item_title If `title_template` exists, the template is rendered with `obj` and `site` in the context, otherwise `item_title` is used.

This attribute becomes the `<title>` element.

description_template or item_description If `description_template` exists, the template is rendered with `obj` and `site` in the context, otherwise `item_description` is used.

This attribute becomes the `<content:encoded>` element.

item_pubdate When the item was published.

The attribute becomes the `<wp:post_date_gmt>` element.

item_guid A unique identifier for this item. By default it is the item's content type name and the item's id, separated by an underscore (_). This allows for exporting comments on several different things without id collisions.

This attribute becomes the `<dsq:thread_identifier>` element.

item_comment_status Can people comment on this item? One of either `open` or `closed`.

This attribute becomes the `<wp:comment_status>` element.

item_comments Return a list of comments for the given item. All comment attributes are mapped based on the attributes below.

comment_id The unique identifier for this comment.

This attribute becomes the `<wp:comment_id>` element.

comment_user_id The unique identifier for the commenting user.

This attribute becomes the `<dsq:id>` element.

comment_avatar The url to the commenting user's avatar

This attribute becomes the `<dsq:avatar>` element.

comment_user_name The name of the commenting user.

This attribute becomes the `<wp:comment_author>` element.

comment_user_email The email of the commenting user.

This attribute becomes the `<wp:comment_author_email>` element.

comment_user_url The commenting user's URL.

This attribute becomes the `<wp:comment_author_url>` element.

comment_ip_address The commenting user's IP address.

This attribute becomes the `<wp:comment_author_IP>` element.

comment_submit_date The date and time when the comment was submitted.

This attribute becomes the `<wp:comment_date_gmt>` element.

comment_comment The text of the content

This attribute becomes the `<wp:comment_content>` element.

comment_is_approved The site moderators have approved this comment for public display. `1` for yes, and `0` for no.

This attribute becomes the `<wp:comment_approved>` element.

comment_parent The id of the comment in which this comment is responding.

This attribute becomes the `<wp:comment_parent>` element.

Release Notes

django-disqus 0.5 (08-MAR-2015)

- Python 3.4 support
- Django 1.7 support
- Improved unit tests

Thanks a lot to Alexey Kalinin, and also everyone else who submitted PR's regarding the Django 1.7 support.

django-disqus 0.4.1 (19-APR-2011)

- Fix installation on Windows (Bug #11)

django-disqus 0.4 (21-MAR-2011)

- Fix for unicode titles in urlencode (http://bugs.python.org/issue1349732). Thanks Adriano Petrich.
- **New templatetags to set context variables:**
  - set_disqus_developer
  - set_disqus_identifier
  - set_disqus_url
  - set_disqus_title

- Export comments as WXR feed

A huge thanks to Corey Oordt, who implemented the new templatetags and the WXR feed.

django-disqus 0.3.4 (30-OCT-2010)

- Update the *disqus_num_replies* template tag to use the new JS code. This will make the site load faster, as loading isn't blocked by the call to document.write. Thanks to Nick Fitzgerald.

django-disqus 0.3.3 (23-SEP-2010)

- Update the *disqus_show_comments* template tag to use the new loader method. Thanks David Cramer for the patch.

django-disqus 0.3.2 (16-MAY-2010)

- Added a *-s/–state-file* option to the *disqus_export* command. The state file saves the id of the last exported comment. This makes it possible to resume interrupted exports. Thanks Horst Gutmann for the patch.

django-disqus 0.3.1 (01-MAY-2010)

This is a bugfix release. The following changes were made:

- Fixed a bug where the disqus_export command raised an error if non-ascii characters were used in the author name.
- Added "async" attribute to DISQUS JavaScript tag. This loads the comments faster on browsers that support the html5 async tag (e.g. firefox).

django-disqus 0.3 (09-MAR-2010)

This release updates django-disqus to use the new DISQUS v1.1 API and cleans up the templatetags and management commands.

Management Commands

The following management commands were renamed:

- `disqus-dumpdata` to `disqus_dumpdata`
- `disqus-export` to `disqus_export`

The old names weren't valid Python module identifiers. This lead to problems when trying to import them.

The `disqus_dumpdata` command has two new options:

- `filter`: Type of entries (e.g. spam or killed) that should be returned
- `exclude`: Type of entries that should be excluded from the response

For further information take a look at the documentation for the *disqus_dumpdata* command.

The `disqus-export-threadedcomments` command was removed from django-disqus because the upcoming `django-threadedcomments` release will rely on the comment extension hooks provided in Django 1.1. This means that the `disqus_export` command will work just fine when exporting threadedcomments.

Templatetags

The `disqus_recent_comments` templatetag was removed. If you want to use this or any other widget, go to the *Tools* section on the DISQUS website. There you can configure the widget and get the Javascript code that is necessary to display it on your website.

The parameters of the `disqus_show_comments` tag have changed. Previously you could pass the title, url, snippet and shortname. As of this release, it's only possible to pass the shortname. If you want to change the Javascript variables that the DISQUS comment form uses, take a look at the Configure and override comment system behaviors page in the DISQUS wiki.