Django-comments-dab Documentation

Release 2.8.0

Radico

Jun 25, 2022

Contents:

CHAPTER 1

django-comments-dab

CHAPTER 2

Introduction

dab stands for Django-Ajax-Bootstrap PS: Ajax and JQuery are not used anymore since v2.0.0 Vanilla JS and fetch API is used instead.

`django-comments-dab` is a commenting application for Django-powered websites.

It allows you to integrate commenting functionality with any model you have e.g. blogs, pictures, video etc. . .

List of actions that can be performed:

1. Post a new comment. (v2.0.0 authenticated and anonymous users)
2. Reply to an existing comment. (v2.0.0 authenticated and anonymous users)
3. Edit a comment. (authenticated user *comment owner*)
4. Delete a comment. (authenticated user *comment owner* and admins)
5. React to a comment. (authenticated users) Available reactions are LIKE and DISLIKE # open PR if you would like to have more reactions
6. Report (flag) a comment. (authenticated users)
7. Delete flagged comment. (admins and moderators)
8. Resolve or reject flag. This is used to revoke the flagged comment state (admins and moderators)
9. Follow and unfollow thread. (authenticated users)
10. Block users/emails (v2.7.0 admins and moderators)

- All actions are done by Fetch API since V2.0.0
- Bootstrap 4.1.1 is used in comment templates for responsive design.

CHAPTER 3

Installation

3.1 Requirements:

1. **django>=2.1**
2. **djangorestframework** # only for API Framework
3. **Bootstrap 4.1.1**

3.2 Installation:

Installation is available via `pip`

```
$ pip install django-comments-dab
```

or via source on github

```
$ git clone https://github.com/radi85/Comment.git
$ cd Comment
$ python setup.py install
```

3.3 Comment Settings and urls:

1. Add `comment` to installed_apps in the `settings.py` file. It should be added after `django.contrib.auth`.
2. `LOGIN_URL` shall be defined in the settings.

`settings.py` should look like this:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    ...
    'comment',
    ..
)

LOGIN_URL = 'login'  # or actual url
```

In `urls.py`:

```
urlpatterns = patterns(
    path('admin/', admin.site.urls),
    path('comment/', include('comment.urls')),
    ...
    path('api/', include('comment.api.urls')),  # only required for API Framework
    ...
)
```

3.4 Migrations:

Migrate comment app:

```
$ python manage.py migrate comment
```

CHAPTER 4

Setup

4.1 Step 1 - Connecting comment model with the target model

In models.py add the field `comments` as a `GenericRelation` field to the required model.

PS: Please note that the field name must be `comments` **NOT** `comment`.

E.g. `Post` model, as shown below:

```
from django.contrib.contenttypes.fields import GenericRelation

from comment.models import Comment

class Post(models.Model):
    author = models.ForeignKey(User)
    title = models.CharField(max_length=200)
    slug = models.SlugField(unique=True)
    body = models.TextField()
    # the field name should be comments
    comments = GenericRelation(Comment)
```

Important: Please add a `get_absolute_url` method to the model that you're associating with the `Comment` model. It is used at several places.

```
from django.urls import reverse

def get_absolute_url(self):
    return reverse('post_detail_url', kwargs={'slug': self.slug})
```

4.2 Step 2 - Adding template tags:

`render_comments` *tag uses 2 required and 1 optional arg*:

1. Instance of the targeted model. (**Required**)
2. Request object. (**Required**)
3. oauth. (optional - Default is false)

CHAPTER 5

Usage

5.1 1. Basics usage:

5.1.1 Rendering Comments

`include_bootstrap` tag is for bootstrap-4.1.1, if it's already used in the project, get rid of this tag.

In the template (e.g. post_detail.) add the following template tags where `obj` is the instance of post model.

```
{% load comment_tags %}  {# Loading the template tag #}
{% render_comments obj request %}  {# Render all the comments belong to the passed␣
↪object "obj" #}
{% include_bootstrap %} {# Include bootstrap 4.1.1 - remove this line if BS is␣
↪already used in your project #}
```

5.1.2 Display Comment Count

In a template where you may want to display count, the following tag can be used.

```
{% get_comments_count obj user %}
```

Here, `obj` refers to the post object instance

5.2 2. Advanced usage:

5.2.1 1. Pagination:

By default, the comments will be paginated, 10 comments per page. To disable the pagination, set `COMMENT_PER_PAGE=None` in your settings file. To change the default number, set `COMMENT_PER_PAGE=number`.

```
{% load comment_tags %}  {# Loading the template tag #}

{% render_comments obj request %}  {# Include comments belonging to a certain object
↪#}
{% include_bootstrap %} {# Include bootstrap 4.1.1 - remove this line if BS 4.1.1 is␣
↪already used in your project #}
```

5.2.2 2. Integrate user profile:

If you have a profile model for the user and you would like to show the profile image next to each comment, do the following steps:

- **Add `PROFILE_APP_NAME` and `PROFILE_MODEL_NAME` variables to your `settings.py` file.** e.g if user profile app is called `accounts` and profile model is called `UserProfile`

`settings.py`:

```
PROFILE_APP_NAME = 'accounts'
PROFILE_MODEL_NAME = 'UserProfile' # letter case insensitive
```

- Make sure that `get_absolute_url` method is defined in your profile model.

```
from django.urls import reverse

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    ...
    ...

    # this method must be defined for appropriate url mapping in comments section
    def get_absolute_url(self):
        return reverse('your_profile_url_name')
```

5.2.3 3. Enable flagging:

The comment can be reported by the users. This feature can be enabled by adding the `COMMENT_FLAGS_ALLOWED` to `settings.py` and its value must be greater than 0 (the default).

The comment that has been reported more than the `COMMENT_FLAGS_ALLOWED` value, will be hidden from the view. To keep displaying the flagged comments to all users add `COMMENT_SHOW_FLAGGED=True` to `settings.py`

The default report reasons are:

1. Spam | Exists only to promote a service.
2. Abusive | Intended at promoting hatred.
3. Something else. With a message info, this option will be always appended reasons list.

The reasons can be customized by adding `COMMENT_FLAG_REASONS` list of tuples to `settings.py`. E.g.

`settings.py`

```
COMMENT_FLAG_REASONS = [
    (1, _('Spam | Exists only to promote a service')),
    (2, _('Abusive | Intended at promoting hatred')),
    (3, _('Racist | Sick mentality')),
```

(continues on next page)

(continued from previous page)

```
    (4, _('Whatever | Your reason')),
    ...
]
```

The flag model has currently 4 states: *since v1.6.7*

1. UNFLAGGED
2. **FLAGGED** - this case only the comment will be hidden
3. REJECTED - flag reasons are rejected by the moderator
4. RESOLVED - the comment content has been changed and accepted by the moderator

Groups and Permissions:

For flagging purpose, the following groups and permissions will be created on the next migrate:

permissions:

1. delete_comment (default)
2. delete_flagged_comment

groups:

1. comment_admin => has both mentioned permissions (edit permission might be added in the future)
2. comment_moderator => has delete_flagged_comment permission

- Comment admin can delete any comment and change the state of flagged comment.
- Comment moderator can delete FLAGGED comment only and change their state.

PS: If the groups or the permissions don't exist, just run migrate. `./manage.py migrate`

5.2.4 4. Allow commenting by anonymous:

Commenting by anonymous is disabled by default. After enabling this feature, unauthenticated users will be able to post a comment by providing their email address. An email will be sent to confirmation. Only after confirming their email address, the comment will be saved in the DB associated with the anonymous user's email. comment only hits the database, after it is verified.

However, since these comment are created anonymously, they won't be editable nor deletable like a normal comments(`comment_admins` and `comment_moderators` can still delete them).

Before enabling this feature, make sure you set the `get_absolute_url` method on the model object with which the Comment model has been associated. For e.g, if the `Comment` model has been associated with the `Post` model, make sure you have something like this set inside your `models.py`

```
class Post(models.Model):
...
slug = models.SlugField(unique=True)
...

def get_absolute_url(self):
    return reverse('post:postdetail', kwargs={'slug': self.slug})
```

To enable this feature, the following settings variables need to be set alongside with django email settings:

```
COMMENT_ALLOW_ANONYMOUS = True
COMMENT_FROM_EMAIL = 'no-reply@email.com'    # used for sending confirmation emails,␣
↪if not set `EMAIL_HOST_USER` will be used.
```

Also, related to sending of email the following settings need to set.

```
EMAIL_HOST_USER = 'user@domain'
EMAIL_HOST_PASSWORD = 'password'
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'    # this backend won
↪'t send emails but will just print them to the console. For production use your own␣
↪backend.

# e.g for if you are using gmail address, you may set:
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
```

To further customize different attributes related to anonymous commenting, you may look into the Settings section for different configurations.

5.2.5 5. Enable gravatar:

To enable using gravatar for profile pics set `COMMENT_USE_GRAVATAR` in `settings.py` to `True`

5.2.6 6. Enable subscription:

To enable app subscription set `COMMENT_ALLOW_SUBSCRIPTION` in `settings.py` to `True`

This will enable the UI functionality and the API endpoint to follow and unfollow *thread*.

The thread can be a *parent* comment or the *content type* (i.e. Post, Picture, Video. . .) that uses the comment model.

Automatic Subscription:

1. Creating a comment in a thread will set the user automatically as a follower of the *thread*.
2. Replying to a comment will set the user as a follower of the *parent* comment

An email notification will be sent to the thread's followers up on adding a new comment to the thread.

PS: This feature needs the email settings to be configured similar to *4. Allow commenting by anonymous:*

5.2.7 7. Enable blocking system

Blocking functionality is added in version 2.7.0. It allows moderators to block users/emails from creating/editing or reacting to a comment.

To enable blocking system set `COMMENT_ALLOW_BLOCKING_USERS` in `settings` to `True`. This will grant access for the **admins** only to block users. However, in order to give the **moderators** this right, you need to add `COMMENT_ALLOW_MODERATOR_TO_BLOCK = True` to *settings*

5.2.8 8. Enable Markdown format

This functionality was added in version `2.8.0`. It allows comment content to be rendered using the power of `markdown` format.

To use this:

- Install additional dependency python-markdown may be installed using `python -m pip install django-comments-dab[markdown]`.
- To enable set `COMMENT_ALLOW_MARKDOWN` to `True` in your `settings` file.
- For advanced configuration, you may use *COMMENT_MARKDOWN_EXTENSIONS* and *COMMENT_MARKDOWN_EXTENSION_CONFIGS*.

CHAPTER 6

Web API

For Swagger page See openapi.html or swagger.html

django-comments-dab uses django-rest-framework to expose a Web API that provides developers with access to the same functionality offered through the web user interface.

The available actions with permitted user as follows:

1. Post a new comment. (authenticated and anonymous users)
2. Reply to an existing comment. (authenticated and anonymous users)
3. Edit a comment. (authenticated user *comment owner*)
4. Delete a comment you posted. (authenticated user *comment owner* and admins)
5. React to a comment. (authenticated users)
6. Report a comment. (authenticated users) Flagging system should be enabled.
7. Delete flagged comment. (admins and moderators)
8. Resolve or reject flag. This is used to revoke the flagged comment state (admins and moderators)
9. Retrieve a list of comments and associated replies to a given content type and object ID.
10. Confirm comment made by an anonymous users.
11. Subscribe a thread. (authenticated users)
12. Retrieve a list of subscribers to a given thread/content type. (admins and moderators)
13. Block users/emails. (admins and moderators)

These actions are explained below.

6.1 Setup:

To integrate the comment API in your content type (e.g Post model), in `serializers.py` for the Post model add comments field as shown below:

```
from rest_framework import serializers
from comment.models import Comment
from comment.api.serializers import CommentSerializer


class PostSerializer(serializers.ModelSerializer):

    comments = serializers.SerializerMethodField()

    class Meta:
        model = Post
        fields = (
            'id',
            ...
            ...
            'comments'
        )

    def get_comments(self, obj):
        comments_qs = Comment.objects.filter_parents_by_object(obj)
        return CommentSerializer(comments_qs, many=True).data
```

By default all fields in profile model will be nested inside the user object in JSON response. This can only happen if the profile attributes are defined in your `settings.py`. In case you would like to serialize particular fields in the profile model you should explicitly declare the `COMMENT_PROFILE_API_FIELDS` tuple inside your `settings.py`:

```
PROFILE_APP_NAME = 'accounts'
PROFILE_MODEL_NAME = 'userprofile'
# the field names below must be similar to your profile model fields
COMMENT_PROFILE_API_FIELDS = ('display_name', 'birth_date', 'image')
```

6.2 Comment API actions:

1- Retrieve the list of comments and associated replies to a given content type and object ID:

This action can be performed by providing the url with data queries related to the content type.

Get request accepts 3 params:

- `model_name`: the model name of the content type that has comments associated with it.
- `model_id`: the id of an object of that model.
- `app_name`: the name of the app that contains the model.

Endpoint call:

```
$ curl -H "Content-Type: application/json" '$BASE_URL/api/comments/?model_name=MODEL_
↪NAME&model_id=ID&app_name=APP_NAME''
```

2- Create a comment or reply to an existing comment:

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

- `type`: is the model name of the content type that have comments associated with it.
- `id`: is the id of an object of that model

- `parent_id`: is 0 or **NOT PROVIDED** for parent comments and for reply comments must be the id of parent comment

Example: posting a parent comment

```
$ curl -X POST -u USERNAME:PASSWORD -d "content=CONTENT" -H "Content-Type:␣
↪application/json" "$BASE_URL/api/comments/create/?model_name=MODEL_NAME&model_id=ID&
↪app_name=APP_NAME&parent_id=0"
```

PS: The parent_id param can be ignored as well to post a parent comment.

3- Update a comment:

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires the `comment.id` that you want to update:

```
$ curl -X PUT -u USERNAME:PASSWORD -d "content=CONTENT" -H "Content-Type: application/
↪json" "$BASE_URL/api/comments/ID/
```

4- Delete a comment:

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires the `comment.id` that you want to delete:

```
$ curl -X DELETE -u USERNAME:PASSWORD -H "Content-Type: application/json" "$BASE_URL/
↪api/comments/ID/
```

5- React to a comment:

`POST` is the allowed method to perform a reaction on a comment.

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires the `comment.id`. and, `reaction_type`: one of `like` or `dislike`

```
$ curl -X POST -u USERNAME:PASSWORD -H "Content-Type: application/json" "$BASE_URL/
↪api/comments/ID/react/REACTION_TYPE/
```

PS: This endpoint is for toggling the reaction as in the UI, clicking the **liked** button will remove the reaction => unlike the comment. This behaviour is performed when repeating the same post request.

6- Report a comment

Flagging system must be enabled by adding the attribute `COMMENT_FLAGS_ALLOWED` to a number(other than zero e.g. 10) in `settings.py`.

`POST` is the allowed method to report a comment.

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires the `comment.id`.

1. Set a flag:

```
payload = {
    "reason": REASON,  # number of the reason
    "info": str  # this is required if the reason is 100 ``Something else``
}
```

```
$ curl -X POST -u USERNAME:PASSWORD -H "Content-Type: application/json" -d '{"reason
↪":1, "info":""}' $BASE_URL/api/comments/ID/flag/
```

2. Un-flag a comment:

To un-flag a FLAGGED comment, set reason value to *0* or remove the payload from the request.

```
$ curl -X POST -u USERNAME:PASSWORD $BASE_URL/api/comments/ID/flag/
```

7- Change flagged comment state

`POST` is the allowed method to report a comment.

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires comment *admin* or *moderator* privilege.

```
payload = {
    "state": STATE  # accepted state is 3 (REJECTED) or 4 (RESOLVED) only
}
```

```
$ curl -X POST -u USERNAME:PASSWORD -H "Content-Type: application/json" -d '{"state
↪":3}' $BASE_URL/api/comments/ID/flag/state/change/
```

Repeating the same request and payload toggle the state to its original.

8- Confirm comment made by an anonymous users

`GET` is the allowed method to confirm an anonymous comment.

Get request accepts 3 params:

- `key`: is the encrypted key that contains the comment.

Example:

```
$ curl -X GET -H "Content-Type: application/json" $BASE_URL/api/comments/confirm/KEY/
```

Since the key generated for each comment is unique, it can only be used once to verify. Any tampering with the key will result in a BAD HTTP request(400).

9- Subscribe a thread

`POST` is the allowed method to toggle subscription.

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

Subscription variable `COMMENT_ALLOW_SUBSCRIPTION` must be enabled in `settings.py`.

```
$ curl -X POST -u USERNAME:PASSWORD -H "Content-Type: application/json" "$BASE_URL/
↪api/comments/toggle-subscription/?model_name=MODEL_NAME&model_id=ID&app_name=APP_
↪NAME"
```

10- Retrieve subscribers on a given thread/content type

`GET`.

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires comment *admin* or *moderator* privilege.

```
$ curl -X GET -u USERNAME:PASSWORD -H "Content-Type: application/json" $BASE_URL/api/
↪comments/subscribers/
```

11- Block users/emails

`POST` is the allowed method to toggle blocking.

Authorization must be provided as a TOKEN or USERNAME:PASSWORD.

This action requires comment *admin* or *moderator* privilege.

```
payload = {
    "comment_id": ID,
    "reason": str  # optional
}
```

```
$ curl -X POST -u USERNAME:PASSWORD -H "Content-Type: application/json" -d '{"comment_
↪id": ID}' $BASE_URL/api/comments/toggle-blocking/
```

CHAPTER 7

OpenAPI

{{openapi}}

CHAPTER 8

Style Customization

Some actual customizations has been done in the example project

8.1 1- Templates and default blocks:

BS classes, pagination and some other template values can be now customized from within your templates directory as follows:

1. Create `comment` folder inside templates directory.
2. Create a new template file `.html` give it the same name of the default template that needs to be overridden and put it in the right directory.

Templates tree:

```
comment
└── templates
    └── comment
        ├── anonymous
        │   ├── confirmation_request.html
        │   ├── confirmation_request.txt
        │   └── discarded.html
        ├── comments
        │   ├── apply_icon.html
        │   ├── base.html
        │   ├── cancel_icon.html
        │   ├── child_comment.html
        │   ├── comment_body.html
        │   ├── comment_content.html
        │   ├── comment_form.html
        │   ├── comment_modal.html
        │   ├── content.html
        │   ├── create_comment.html
        │   ├── delete_icon.html
```

(continues on next page)

(continued from previous page)

```
        ├── edit_icon.html
        ├── messages.html
        ├── pagination.html
        ├── parent_comment.html
        ├── reject_icon.html
        ├── resolve_icon.html
        ├── three_dots_menu.html
        ├── update_comment.html
        └── urlhash.html
    ├── email
        ├── email_template.html
        └── footer.html
    ├── flags
        ├── flag_icon.html
        ├── flag_modal.html
        └── flags.html
    ├── follow
        ├── follow.html
        ├── follow_icon.html
        └── follow_modal.html
    ├── notifications
        ├── notification.html
        └── notification.txt
    ├── reactions
        ├── dislike_icon.html
        ├── like_icon.html
        └── reactions.html
    ├── base.html
    ├── bootstrap.html
    └── static.html
```

for example to override the BS classes of *submit buttons* and pagination style do the following:

create `templates/comment/comments/create_comment.html`

```
{% extends "comment/comments/create_comment.html" %}

{% block submit_button_cls %}
btn btn-primary btn-block btn-sm
{% endblock submit_button_cls %}

{# override pagination style: #}
{% block pagination %}
{% include 'comment/comments/pagination.html' with active_btn='bg-danger'␣
↪text_style='text-dark' li_cls='page-item rounded mx-1' %}
{% endblock pagination %}
```

8.1.1 Templates and block tags names with default values:

This style customization is compatible with version **>= 1.6.5** Some block tags may not work on old versions.

base.html

```
{% extends "comment/comments/base.html" %}

{% block comment_section_cls %}my-5 mx-3{% endblock comment_section_cls %}
```

(continues on next page)

(continued from previous page)

```
{% block header_div_cls %}border-bottom mb-4{% endblock header_div_cls %}
{% block header_title_cls %}bb{% endblock header_title_cls %}
{% block follow_icon_wrapper_cls %}float-right{% endblock follow_icon_wrapper_cls %}

{% block pagination %}  {# override default pagination classes #}
{% include 'comment/comments/pagination.html' with active_btn='bg-success' text_style=
↪'text-success' li_cls='page-item rounded mx-1' %}
{% endblock pagination %}
```

create_comment.html

```
{% extends "comment/comments/create_comment.html" %}

{% block text_area_wrapper_cls %}col-sm-9 col-md-10 px-2 m-2 m-sm-0{% endblock text_
↪area_wrapper_cls %}
{% block submit_button_wrapper_cls %}col-sm-3 col-md-2 px-2 m-3 m-sm-0{% endblock␣
↪submit_button_wrapper_cls %}
{% block submit_button_cls %}btn btn-outline-success btn-block btn-sm{% endblock␣
↪submit_button_cls %}

{% block login_info %}
{# Please see 'comment/comments/create_comment.html' template for default block #}
{% endblock login_info %}

{% block oauth %}  {# override default oauth urls section #}
<a class="mx-1 my-0 h4 github-color" href="{% url 'social:begin' 'github' %}?next={
↪{request.path}}"><i class="fa fa-github-square"></i></a>
<a class="mx-1 my-0 h4 facebook-color" href="{% url 'social:begin' 'facebook' %}?next=
↪{{request.path}}"><i class="fa fa-facebook-square"></i></a>
<a class="mx-1 my-0 h4 twitter-color" href="{% url 'social:begin' 'twitter' %}?next={
↪{request.path}}"><i class="fa fa-twitter-square"></i></a>
<a class="mx-1 my-0 h4 google-color" href="{% url 'social:begin' 'google-oauth2' %}?
↪next={{request.path}}"><i class="fa fa-google-plus-square"></i></a>
{% endblock oauth %}
```

parent_comment.html

```
{% extends "comment/comments/parent_comment.html" %}

{% block parent_comment_wrapper_cls %}text-wrap{% endblock parent_comment_wrapper_cls
↪%}
{% block replies_wrapper_cls %}ml-5 my-4{% endblock replies_wrapper_cls %}
```

child_comment.html

```
{% extends "comment/comments/child_comment.html" %}

{% block child_comment_wrapper_cls %}text-wrap mb-4{% endblock child_comment_wrapper_
↪cls %}
```

comment_body.html

```
{% extends "comment/comments/comment_body.html" %}

{% block image_wrapper_cls %}col-2 col-md-1{% endblock image_wrapper_cls %}
```

(continues on next page)

(continued from previous page)

```
{% block image_cls %}w-100{% endblock image_cls %}

{% block three_dots_wrapper_cls %}col-1{% endblock three_dots_wrapper_cls %}
```

comment_content.html

```
{% extends "comment/comments/comment_content.html" %}

{% block content_wrapper_cls %}{% if has_valid_profile %}col-9 col-md-10{% else %}co-
↪11 mx-3{% endif %}{% endblock content_wrapper_cls %}
{% block comment_content %}   {# override truncate words number - change the number␣
↪30 to your desired or 0 if you don't want to fold the comment #}
    {# new settings variable COMMENT_WRAP_CONTENT_WORDS is introduce for changing the␣
↪number of wrapped words. #}
    {# working with the settings var is more convenient than overriding the template
↪#}
    {% render_content comment words_number %}  {# words_number is the settings␣
↪variable COMMENT_WRAP_CONTENT_WORDS #}
{% endblock comment_content %}

{% block footer_wrapper_cls %}mt-2 text-muted{% endblock footer_wrapper_cls %}
{% block username_cls %}{% endblock username_cls %}
{% block reply_count_cls %}text-dark{% endblock reply_count_cls %}
{% block reply_link_cls %}btn btn-link ml-1{% endblock reply_link_cls %}
{% block follow_icon_wrapper_cls %}d-inline ml-3{% endblock follow_icon_wrapper_cls %}
```

content.html

```
{% extends "comment/comments/content.html" %}

{% block content_text_cls %}mb-0{% endblock content_text_cls %}
{% block read_more_cls %}btn btn-link btn-xs read-more{% endblock read_more_cls %}
```

edit_icon.html

```
{% extends "comment/comments/edit_icon.html" %}

{% block edit_link_cls %}btn btn-link{% endblock edit_link_cls %}
{% block edit_img_icon %}Here comes your favorite icon{% endblock edit_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block edit_icon_color %}#00bc8c{% endblock edit_icon_color %}
```

delete_icon.html

```
{% extends "comment/comments/delete_icon.html" %}

{% block delete_btn_cls %}btn btn-link{% endblock delete_btn_cls %}
{% block delete_img_icon %}Here comes your favorite icon{% endblock delete_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block delete_icon_color %}#E74C3C{% endblock delete_icon_color %}
```

apply_icon.html

```
{% extends "comment/comments/apply_icon.html" %}

{% block apply_btn_cls %}btn btn-link{% endblock apply_btn_cls %}
{% block apply_img_icon %}Here comes your favorite icon{% endblock apply_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block apply_icon_color %}#00bc8c{% endblock apply_icon_color %}
```

cancel_icon.html

```
{% extends "comment/comments/cancel_icon.html" %}

{% block cancel_btn_cls %}btn btn-link{% endblock cancel_btn_cls %}
{% block cancel_img_icon %}Here comes your favorite icon{% endblock cancel_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block cancel_icon_color %}#E74C3C{% endblock cancel_icon_color %}
```

flag_icon.html

```
{% extends "comment/flags/flag_icon.html" %}

{% block flag_img_icon %}
    {#
    IMPORTANT: please consider adding these classes to your icon element as they are␣
↪used in JS
    class="comment-flag-icon {% if user|has_flagged:comment %}user-has-flagged{% else
↪%}user-has-not-flagged{% endif %}"
    #}
    Here comes your favorite icon
{% endblock flag_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block flag_icon_color %}#427297{% endblock flag_icon_color %}
```

like_icon.html

```
{% extends "comment/actions/like_icon.html" %}

{% block like_img_icon %}
    {% load comment_tags %}
    {% has_reacted user=user comment=comment reaction="like" as has_user_liked %}
    {#
    IMPORTANT: please consider adding these classes to your icon element as they are␣
↪used in JS
    class="comment-reaction-icon reaction-like {% if has_user_liked %}user-has-reacted
↪{% else %}user-has-not-reacted{% endif %}"
    #}
    Here comes your favorite icon
{% endblock like_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block like_icon_color %}#427297{% endblock like_icon_color %}
```

dislike_icon.html

```
{% extends "comment/comments/reject_icon.html" %}

{% block dislike_img_icon %}
    {% load comment_tags %}
    {% has_reacted user=user comment=comment reaction="dislike" as has_user_disliked
↪%}
    {#
    IMPORTANT: please consider adding these classes to your icon element as they are␣
↪used in JS
    class="comment-reaction-icon reaction-dislike {% if has_user_disliked %}user-has-
↪reacted{% else %}user-has-not-reacted{% endif %}"
    #}
    Here comes your favorite icon
{% endblock dislike_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block dislike_icon_color %}#427297{% endblock dislike_icon_color %}
```

reject_icon.html

```
{% extends "comment/comments/reject_icon.html" %}

{% block reject_img_icon %}}
    {#
    IMPORTANT: please consider adding this class to your icon element as it is used␣
↪in JS
    class="{% if comment.has_rejected_state %}flag-rejected{% endif %}"
    #}
    Here comes your favorite icon
{% block reject_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block reject_icon_color %}#427297{% endblock reject_icon_color %}
```

resolve_icon.html

```
{% extends "comment/comments/resolve_icon.html" %}

{% block resolved_img_icon %}}
    {#
    IMPORTANT: please consider adding this class to your icon element as it is used␣
↪in JS
    class="{% if comment.has_resolved_state %}flag-resolved{% endif %}"
    #}
    Here comes your favorite icon
{% block resolved_img_icon %}

{# use this tag for overriding the default icon color, this tag won't have effect in␣
↪case of using the above one #}
{% block resolved_icon_color %}#427297{% endblock resolved_icon_color %}
```

follow_icon.html

```
{% extends "comment/follow/follow_icon.html" %}

{% block follow_img_icon %}
    {#
    IMPORTANT: please consider adding these classes to your icon element as they are␣
↪used in JS
    class="comment-follow-icon {% if user|has_followed:model_object %}user-has-
↪followed{% endif %}"
    #}
    Here comes your favorite icon
{% endblock follow_img_icon %}
```

comment_modal.html

```
{% extends "comment/comments/comment_modal.html" %}

{% block title %}
Confirm comment deletion
{% endblock title %}

{% block content %}
Are you sure you want to delete this comment
{% endblock content %}

{% block close_btn_cls %}
btn btn-secondary
{% endblock close_btn_cls %}


{% block del_btn_cls %}
btn btn-danger
{% endblock del_btn_cls %}
```

flag_modal.html

```
{% extends "comment/flags/flag_modal.html" %}

{% block title %}
{% trans "Please select a reason for flagging" %}
{% endblock title %}

{% block flag_link_cls %}{% endblock flag_link_cls %}
```

follow_modal.html

```
{% extends "comment/follow/follow_modal.html" %}

{% block title %}
{% trans "Please insert your email to follow this thread" %}
{% endblock title %}

{% block email_input %}
    <div class="row">
        <div class="col-3">
            <label for="email">Email:</label>
        </div>
        <div class="col-9">
```

(continues on next page)

(continued from previous page)

```
            <input id="email" class="form-control mr-2 w-100" type="email" name="email
↪" required>
            <div class="error text-danger small mt-1"></div>
        </div>
    </div>
{% endblock email_input %}

{% block follow_btn_extra_cls %}{% endblock follow_btn_extra_cls %}
```

8.1.2 Email templates:

Responsive email templates are used from https://github.com/leemunroe/responsive-html-email-template

This can be overridden by creating `base.html` in *templates/comment/email/* directory.

Both `confirmation_request.html` and `notification.html` extends the base email template and they have the following blocks for partial customization:

```
{% extends "comment/notifications/notification.html" %}

{% block content %}
{# your custom email message/template here #}
{% endblock content %}

{% block footer %}
{# your footer here #}
{% endblock footer %}
```

PS: The footer template is disabled by default.

8.2 2- CSS file:

To customize the default style of comments app , you can create a `comment.css` file inside `static/css` directory.

The new created file will override the original file used in the app.

CHAPTER 9

Example

Using local virtual env

```
$ git clone https://github.com/Radi85/Comment.git  # or clone your forked repo
$ cd Comment
$ python3 -m venv local_env  # or any name. local_env is in .gitignore
$ export DEBUG=True
$ source local_env/bin/activate
$ pip install -r test/example/requirements.txt
$ python manage.py migrate
$ python manage.py create_initial_data
$ python manage.py runserver
```

Or run with docker

```
$ git clone https://github.com/Radi85/Comment.git  # or clone your forked repo
$ cd Comment
$ docker-compose up
```

Login with:

username: `test`

password: `test`

CHAPTER 10

Settings

django-comments-dab has a few configuration options that allow you to customize it.

10.1 PROFILE_APP_NAME

The django app that contains the model that has user profiles. This will be used to display profile pictures alongside the comments. Defaults to `None`.

10.2 PROFILE_MODEL_NAME

The model that contains the user profiles. This will be used in display profile pictures alongside the comments. Defaults to `None`.

10.3 COMMENT_PROFILE_API_FIELDS

This will only be useful if `PROFILE_APP_NAME` and `PROFILE_MODEL_NAME` are defined in your `settings.py`. By default all fields in profile model will be nested inside the user object in JSON response. In case you would like to serialize particular fields in the profile model you should explicitly declare the `COMMENT_PROFILE_API_FIELDS` tuple inside your `settings.py`:

```
PROFILE_APP_NAME = 'accounts'
PROFILE_MODEL_NAME = 'userprofile'
# the field names below must be similar to your profile model fields
COMMENT_PROFILE_API_FIELDS = ('display_name', 'birth_date', 'image')
```

10.4 COMMENT_USER_API_FIELDS

The fields returned for the `user` serializer by the REST API. Defaults to `['id', 'username', 'email']`.

10.5 COMMENT_FLAGS_ALLOWED

Number of flags allowed before a comment is termed as flagged for review. Defaults to `10`. To disable the flagging feature set this as `None` or `0`.

10.6 COMMENT_SHOW_FLAGGED

Should flagged comment be shown or not? Defaults to `False`.

10.7 COMMENT_FLAG_REASONS

The reasons for which a comment can be flagged. Users will have a choose one of these before they flag a comment. This a list of tuples. Defaults to:

```
from django.utils.translation import gettext_lazy as _

[
    (1, _('Spam | Exists only to promote a service')),
    (2, _('Abusive | Intended at promoting hatred')),
]
```

10.8 COMMENT_URL_PREFIX

The prefix to be used when assigning a `urlhash` to a comment. Defaults to `'comment-'`.

10.9 COMMENT_URL_SUFFIX

The prefix to be used when assigning a `urlhash` to a comment. Defaults to `''`.

10.10 COMMENT_URL_ID_LENGTH

The length of the unique id generated for `urlhash` to a comment. Defaults to `8`.

10.11 COMMENT_PER_PAGE

No. of comments to be displayed per page. Defaults to `10`. To disable pagination, set it to `None`.

10.12 COMMENT_ORDER_BY

Order parent comments in a specific order. Defaults to `['-posted']`.

Note: Allowed order should contain a combination of any of the following values without repeating themselves.

Value	Comment Ordered By
`posted`	Date posted, ascendingly
`-posted`	Date posted, descending
`reaction__likes`	Like count, ascendingly
`-reaction__likes`	Like count, descendingly
`reaction__dislikes`	Dislike count, ascendingly
`-reaction__dislikes`	Dislike count, descendingly
`?`	Random

10.13 COMMENT_ALLOW_ANONYMOUS

Should the anonymous commenting featured be allowed? Defaults to `False`.

10.14 COMMENT_FROM_EMAIL

The email address to be used for sending email for comment confirmation. Defaults to the value of `EMAIL_HOST_USER`.

10.15 COMMENT_CONTACT_EMAIL

Used for contact address in confirmation emails. For e.g. `contact@domain`. Defaults to the value of `COMMENT_FROM_EMAIL`.

10.16 COMMENT_SEND_HTML_EMAIL

Should the email to be sent for confirmation contain `html` part as well? Defaults to `True`.

10.17 COMMENT_ANONYMOUS_USERNAME

Username to be shown beside anonymous comment. Defaults to `Anonymous User`.

10.18 COMMENT_USE_EMAIL_FIRST_PART_AS_USERNAME

Whether to use the first part of the email address as username for anonymous comments? For e.g. for `user@domain`, `user` will be used. Defaults to `False`.

10.19 COMMENT_USE_GRAVATAR

Whether to use gravatar for displaying profile pictures alongside comments. Defaults to `False`.

10.20 COMMENT_ALLOW_SUBSCRIPTION

Allow threads subscription feature. Defaults to `False`.

10.21 COMMENT_WRAP_CONTENT_WORDS

Number of comment content to be show and the rest of words to be wrapped. Default is `30`. Changing it to `0` or `None` no words will be wrapped (Full content is shown/rendered).

10.22 COMMENT_DEFAULT_PROFILE_PIC_LOC

Provides an alternate location for profile picture that can be used other than default image. Defaults to '/static/img/default.png'

10.23 COMMENT_ALLOW_BLOCKING_USERS

Enable blocking system. This gives only **admins** the right. Default to `False`

10.24 COMMENT_ALLOW_MODERATOR_TO_BLOCK

Allow **moderators** to perform blocking action when *COMMENT_ALLOW_BLOCKING_USERS* is enabled. Default to `False`

10.25 COMMENT_RESPONSE_FOR_BLOCKED_USER

The response message for blocking reason. Default to `You cannot perform this action at the moment! Contact the admin for more details`

10.26 COMMENT_ALLOW_MARKDOWN

Enable rendering comment content in markdown format. Defaults to `False`.

Note: When `markdown` format is being used to render content, no content wrapping is done. Passing a value for wrapping to the `render_content` template tag in such situations will raise a `RuntimeWarning`.

10.27 COMMENT_MARKDOWN_EXTENSIONS

The list of extensions to be used for the rendering the `markdown`. Defaults to `['markdown.extensions.fenced_code']`. See python markdown's documentation for more information on this.

Note: Both `COMMENT_MARKDOWN_EXTENSIONS` and `COMMENT_MARKDOWN_EXTENSION_CONFIG` will only be used when `COMMENT_ALLOW_MARKDOWN` is set to `True`.

10.28 COMMENT_MARKDOWN_EXTENSION_CONFIGS

The configuration used for markdown-extensions. Defaults to `{}`. See python markdown's documentation for more information.

CHAPTER 11

Internationalization

django-comments-dab is i18n ready. Please, consider extending support for your language if it's not listed below. At the moment it's available only in:

- English, **en** (default language)
- Hindi, **hi**
- To enable internationalization in your django applications, in your `settings` file, set

```
USE_I18N = True
USE_L18N = True
COMMENT_ALLOW_TRANSLATION = True
```

- To generate the translation files, run:

```
python manage.py compilemessages
```

11.1 Adding Support for Translation

We would love to expand this projects to support as many languages as possible. In case your native language is not yet supported, it would be really nice if you could take the time to add translations. To add translation support for your native language:

- Please follow the guidelines for *setting up the project*.
- Find out the ISO_639-1 code for your language.
- Now that you have setup the project, from the root directory of your django project, run:

```
python manage.py makemessages -l my_language_code
# for generating translations corresponding to javascript code
python manage.py makemessages -l my_language_code -d djangojs
```

- This will create two files with the extension `.po` inside the `locale/{language_code}/LC_MESSAGES/` directory.
- After adding translation to both the files, please run the following command to verify everything is working:

```
python manage.py compilemessages -l my_language_code
```

- If you don't see an error in the last command, your translations have been added in the correct format.
- Please consider performing a couple of operations with the UI on the `example` app. These guidelines help you in doing so.
  - Change the language for the `example` app.

```
# in the file test/settings.py
LANGUAGE_CODE = '{my_language_code}'
```

  - Now, you may test the app by running:

```
python manage.py runserver # or docker-compose up(if you are using docker)
```

  - Open the URL `localhost:8000` inside your browser.
  - If everything looks fine, you are good to go to the final step.
- You are now ready push the changes(the two `.po` files) to your forked repository and make a pull-request from there.

Thanks for taking the time to add translations, you are awesome!.

CHAPTER 12

Contributing

Django Comments Dab is developed and maintained by developers in an Open Source manner. Any support is welcome. You could help by writing documentation, pull-requests, report issues and/or translations.

12.1 Starting points

An issue with a good first label might be a good place to start with.

You can also try to take up an issue tagged with an upcoming release.

12.2 PR and commit messages

Pull Requests

Refer to #125

In order to keep the default branch clean and up to date with the current release, all PR shall be merged with `develop` branch.

Commit messages

Use one commit per issue and try to keep the first line within 50 characters.

Need more? use the commit body.

The commit message should reference the issue number in the header (first line) like so:

feat(*#NUMBER*): YOUR COMMIT

headers:

- feat(*#NUMBER*) for adding new feature
- fix(*#NUMBER*) for fixing a bug
- ref(*#NUMBER*) for code refactoring and enhancement

- test(*#NUMBER*) for adding, fixing or adjusting tests
- doc(*#NUMBER*) for documentation
- chore(*#NUMBER*) Changes to the build process, new releases and work doesn't relate to any of the previous header

The issue number can be skipped if not available..

12.3 Development

To start development on this project, fork this repository and follow the guidelines given below.

```
# clone the forked repository
$ git clone YOUR_FORKED_REPO_URL

# create a virtual environment
$ python3 -m venv local_env
# activate the virtual environment
$ source local_env/bin/activate
# install dependencies
(local_env) $ pip install -e . -r test/example/requirements.txt

(local_env) $ export DEBUG="True"
# migrate the changes to database
(local_env) $ python manage.py migrate
# prepare initial data
(local_env) $ python manage.py create_initial_data
# start the development server
(local_env) $ python manage.py runserver
```

Or run with docker

```
$ git clone YOUR_FORKED_REPO_URL
$ cd Comment
$ docker-compose up
```

Login with:

username: `test`

password: `test`

12.4 Testing

To run tests against a particular `python` and `django` version installed inside your virtual environment, you may use:

```
(local_env) $ python manage.py test --settings=test.settings.test
```

To run tests against all supported `python` and `django` versions, you may run:

```
# install dependency
(local_env) $ pip install tox
# run tests
(local_env) $ tox
```

12.5 Translations

To add translations in your native language, please take a look at the instructions for translators.

CHAPTER 13

Release Notes

13.1 Django-comments-dab v2.6

13.1.1 v2.6.0

Features

- #43 - Add thread (content-type and parent comment) subscription.
- #43 - Send email notification to thread's subscribers on creating new comment.
- #83 - Add ordering option for comments.
- #117 - Add support for customization of user fields in the api.
- #128 - Allow custom user model fields.
- #142 - Extend ui customization.
- #158 - Allow rendering new line in comment.

Bug fixes

- #135 - Fix modal X button issue.
- #139 - Fix url resolving issue.
- #157 - Fix pluralization issue.

Codebase enhancement

- #112 - Reduce testing time in pipeline.
- #122 - Code clean up.

- #129 - Update development guideline.
- #132 and #137 - Tests enhancement and refactoring.
- #144 - Move common logic from views to mixin.
- #150 - Unify django views responses.
- #165 - Solve CI issue with django master.

13.1.2 v2.6.1

Features

- #163 - Add option for default profile pic location.

Bug fixes

- #168 - Fix redirect path after login (Pass *request* object in template context).
- #175 - Fix creating replies when subscription is disabled.

Codebase enhancement

- #147 - Add missing step to setup documentation.
- #173 - Rename default django branch to main.

13.2 Django-comments-dab v2.7

13.2.1 v2.7.0

Features

- #143 - Allow blocking users and emails.
- #156 - Improve UI for commenting anonymously.
- #203 - Enhance API docs by adding openapi and swagger page to RTD.
- #215 - Reduce number of queries by prefetching foreign key objects on comment.
- #218 - Add support for django `3.2`.

Bug fixes

- #202 - Fix response for state change on unflagged comments.
- #210 - Reduce chances of XSS injections.
- #213 - Fix namespace pollution caused by star imports.
- #216 - Give warnings when deprecated template tag include_static is used.

Codebase enhancement

- #205 - Enhance Coverage - Add tests for permissions.
- #206 - Move metadata from setup.py to setup.cfg.
- #200 and #209 - Tests enhancement and refactoring.
- #223 - Migrate to github actions for CI.
- #225 - Run tests for all supported python and django versions.

13.2.2 v2.7.1

Bug fixes

- #230 - Fix closing of anonymous create comment modal.

CHAPTER 14

Changelog

14.1 2.8.0

- Using exists instead of count is faster.
- Confirm support for django 4.0.
- Confirm support for python 3.10
- Move validation of orders to system checks.
- Add support for rendering content in markdown format.

14.2 2.7.1

- Fix closing of anonymous create comment modal.

14.3 2.7.0

- Add support for django `3.2`.
- Allow blocking users/emails from adding/reacting with comments.
- Improve commenting anonymously UI.
- Enhance API docs by adding openapi and swagger page to RTD.
- Reduce number of queries by prefetching foreign key objects on comment.
- Reduce chances of XSS injections.
- Fix response for state change on unflagged comments.

14.4 2.6.1

- Fix redirect path after login (Pass *request* object in template context).
- Fix creating replies when subscription is disabled.
- Add missing step to setup documentation.
- Add option for default profile pic location.

14.5 2.6.0

- Support rendering new lines in the comment content.
- Fix pluralization issue for the translation.
- Add support for custom fields in user model.
- Add ordering option for comments.
- extend UI customization.
- Fix bugs.
- Add subscription feature.
- Send email notifications.

14.6 2.5.1

- Fix version/installation issue.
- Fix class names conflict.

14.7 2.5.0

- Add django 3.1 compatibility.
- Add gravatar support.
- Add i18n support.
- Include static files implicitly. `include_static` template tag is deprecated.
- Bugs fixes.

14.8 2.0.0

- Allow commenting by unauthenticated users (Anonymous comment).
- Add permalink to comments.
- Remove JQuery from dependencies and replace it with Vanilla JS.
- Update mixin and add content type and parent id validators.

- Bug fixes.

14.9 1.6.7

- Add states to flag model
- Add functionality to allow comment admin or moderator to change flag state
- Extend the API to cover all GUI actions

14.10 1.6.5

- Add groups and permissions
- Update styling
- Make the style more customizable

14.11 1.6.1

- Fix bugs

14.12 1.6.0

- Add flagging system - Report a comment

14.13 1.5.0

- Add reactions - (LIKE and DISLIKE)
- Restrict the requests to AJAX calls only

14.14 1.4.0

- Remove unnecessary dependencies.
- Add unittests for all components.
- Add compatibility checking with django versions >= 2.1

14.15 1.3.0

- For more compatibility with ContentType (models), slug option has been deprecated.
- Now retrieving and creating comment is based on provided ContentType and its id only.

14.16 1.2.4

- Integrate profile fields with user serializer

14.17 1.2.3

- Change the retrieved comments list in the API from all comments to list of comments and associated replies to a given content type and object ID

14.18 1.2.2

- Update pagination on comment action

14.19 1.2.1

- Fix static files bug

14.20 1.2.0

- Serialize comments
- Add web API feature

14.21 1.1.0

- Add pagination feature

14.22 1.0.1

- Move profile_model_name and profile_app_name to setting file
- Fix a bug due to letter case in ContentType class

14.23 1.0.0

First release

CHAPTER 15

License

CHAPTER 16

Help

Need for help? please contact me: mus.radi85@gmail.com