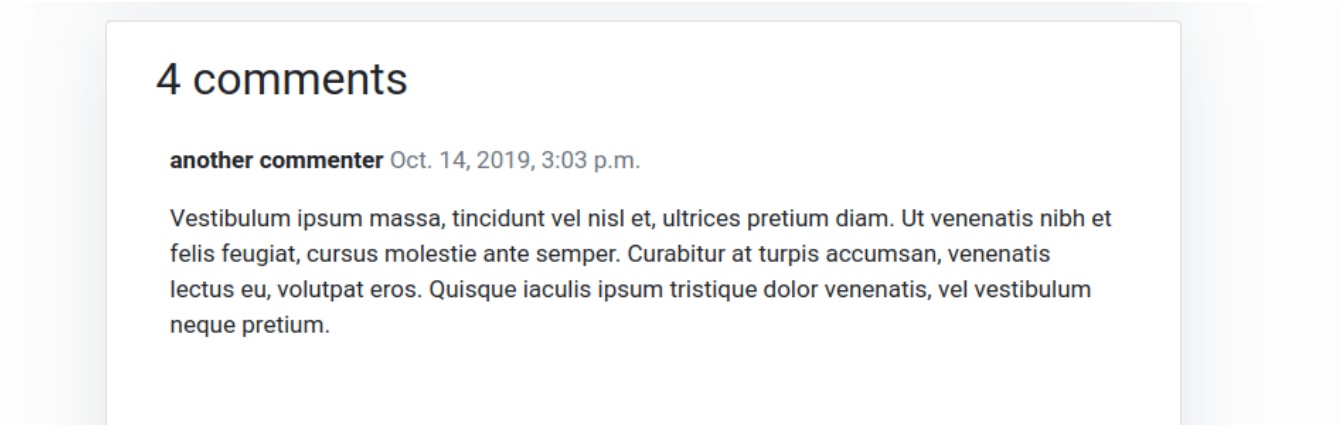
DJANGOCENTRAL

Creating Comments System With Django

In this tutorial, we will build a basic commenting system for a Django 2.X app, which lets readers add comments on posts.

Here is a preview of what we are going to build by the end of this tutorial.



Pre-Requirements

Before diving into the tutorial I am assuming that you already have a blog or some similar Django 2.X project up and running. I am using Bootstrap 4 for the styling part you can ignore it if you are using any other framework.

I am gonna use my previous blog project you can grab the repo from here, https://github.com/TheAbhijeet/Django_blog/releases/tag/1

Since Python 3 is the current version in active development and addressed as the future of Python, Django rolled out a significant update, and now all the releases after Django 2.0 are only compatible with Python 3.x. Therefore this tutorial is **strictly for Python 3.x**.

Roadmap To Build A Comment System

1. Create a model to save the comments.
2. Create a form to submit comments and validate the input data.
3. Add a view that processes the form and saves the new comment to the database.
4. Edit the post detail template to display the list of comments and the form to add a new comment.

Building Comment Model

Open The models.py file of blog application and below the Post model create the Comment model.

```
class Comment(models.Model):
    post = models.ForeignKey(Post,on_delete=models.CASCADE,related_name='c
    name = models.CharField(max_length=80)
    email = models.EmailField()
    body = models.TextField()
    created_on = models.DateTimeField(auto_now_add=True)
    active = models.BooleanField(default=False)

    class Meta:
        ordering = ['created_on']

    def __str__(self):
        return 'Comment {} by {}'.format(self.body, self.name)
```

In this comment model first, we have a Foreign key relation that establishes a many-to-one relationship with the `Post` model, since every comment will be made on a post and each post will have multiple comments.

The `related_name` attribute allows us to name the attribute that we use for the relation from the related object back to this one. After defining this, we can retrieve the post of a comment object using `comment.post` and retrieve all comments of a post using `post.comments.all()` . If you don't define the `related_name` attribute, Django will use the name of the model in lowercase, followed by `_set` (that is, comment_set) to name the manager of the related object back to this one.

As a traditional comment system, we are accepting the commenter's name, email and comment body as inputs. Then we have an `active` boolean field that is set to False to prevent spam we will manually allow all the comments posted.

The Meta class inside the model contains metadata. We tell Django to sort results in the `created_on` field in descending order by default when we query the database. We specify descending order using the negative prefix. By doing so, comments made recently will appear first.

The `__str__()` method is the default human-readable representation of the object. Django will use it in many places, such as the administration site.

Next, we need to synchronize this comment model into the database by running migrations to reflects the changes in the database.

```
(django) $ python manage.py makemigrations
(django) $ python manage.py migrate
```

We are done with the models, now let's include the Comment model in our Admin dashboard.

Adding Comments Model To The Administration Site

Open `admins.py` file and write the following code.

```
from django.contrib import admin
from .models import Post, Comment
@admin.register(Comment)
class CommentAdmin(admin.ModelAdmin):
    list_display = ('name', 'body', 'post', 'created_on', 'active')
    list_filter = ('active', 'created_on')
    search_fields = ('name', 'email', 'body')
    actions = ['approve_comments']

    def approve_comments(self, request, queryset):
        queryset.update(active=True)
```

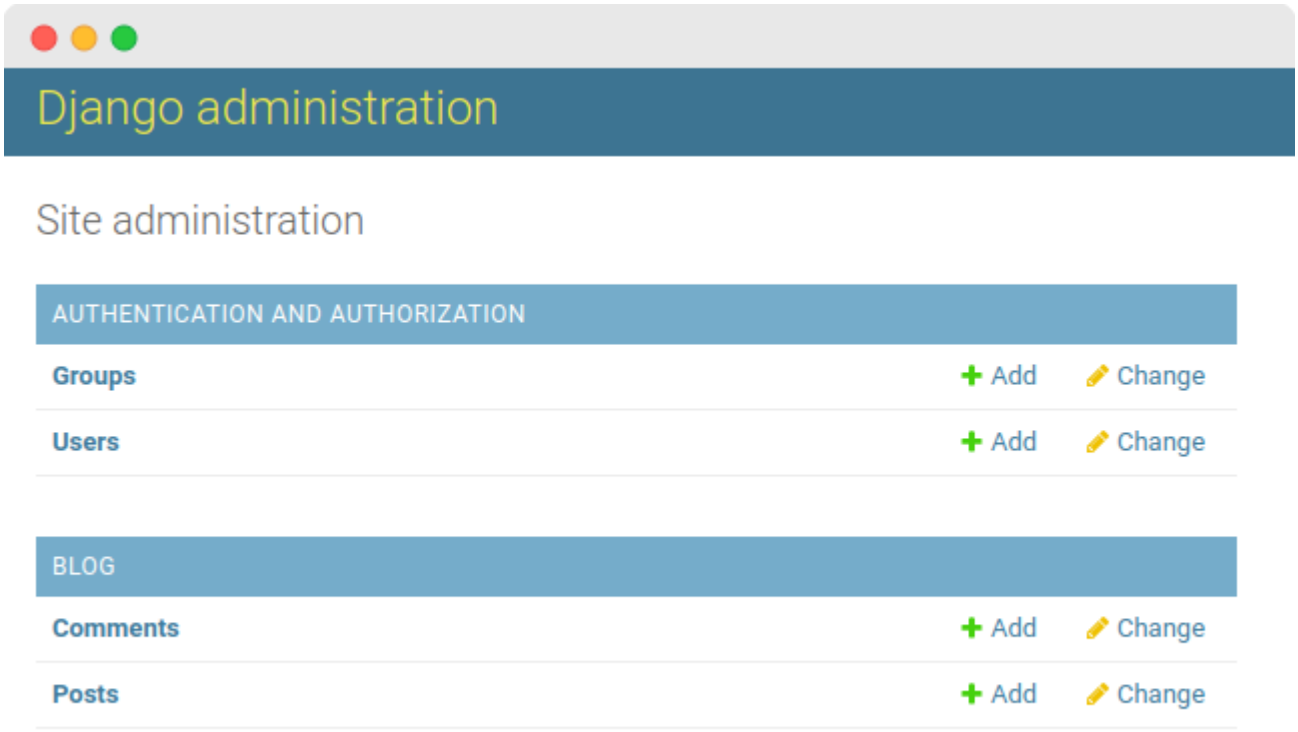
Going over the code `@admin.register(Comment)` registers the comment into the Admin area. Below the `CommentAdmin` class to customizes the representation of data on the screen.

The `list_display` attribute does what its name suggests display the properties mentioned in the tuple in the comments list for each comment.

The `list_filter` method will filter the comments based on the creation date and their active status and `search_fields` will simply search the database for the parameters provided in the tuple.

Finally, we have the actions method this will help us for approving many comment objects at once, the `approve_comments` method is a simple function that takes a queryset and updates the active boolean field to `True` .

Now create a superuser if you haven't already and log in to the dashboard you should see the comment model there.



Now click on comments and create your comments.

In case you struck an error like **no such table: blog_comment** you might wanna delete the SQLite file and run migrations again for a quick fix.

Creating forms from models

Django offers a very rich and secure API to handle forms. Since the form input will be saved in the database models we are gonna use the Django's **ModelForm**.

A common practice is to create a `forms.py` file inside your app directory for all the forms of an app. So create a `forms.py` file in your app and write the following code.

```
from .models import Comment
from django import forms


class CommentForm(forms.ModelForm):
    class Meta:
        model = Comment
        fields = ('name', 'email', 'body')
```

In the model form, we just need to provide the model name in the `Meta` class of the form Django will handle the form processing and validation on the basis of fields of the model.

By default, Django will generate a form dynamically from all fields of the model but we can explicitly define the fields we want the forms to have, that is what `fields` attribute is doing here.

Building Views

We will modify the post detail view for form processing using function based view.

```
from .models import Post
from .forms import CommentForm
from django.shortcuts import render, get_object_or_404

def post_detail(request, slug):
    template_name = 'post_detail.html'
    post = get_object_or_404(Post, slug=slug)
    comments = post.comments.filter(active=True)
    new_comment = None
```

```
    # Comment posted
    if request.method == 'POST':
        comment_form = CommentForm(data=request.POST)
        if comment_form.is_valid():

            # Create Comment object but don't save to database yet
            new_comment = comment_form.save(commit=False)
            # Assign the current post to the comment
            new_comment.post = post
            # Save the comment to the database
            new_comment.save()
    else:
        comment_form = CommentForm()

    return render(request, template_name, {'post': post,
                                           'comments': comments,
                                           'new_comment': new_comment,
                                           'comment_form': comment_form})
```

This post detail view will show the post and all its comments, let's break it down to see what's happening.

First, we assigned the HTML template to a variable name `template_name` for future reference and we are assigning the Post object inside the `post` variable.

This `comments = post.comments.filter(active=True)` queryset retrieves all the approved comments from the database.

Since this is the same page where users will create new comments we initialized the `new_comment` variable by setting it to none.

Next, we have a conditional statement if a `POST` request is made, the comment_form variable will hold the data of user input next Django will validate the data using the `is_valid()` method.

If the form is valid the following actions take place.

1. We create a new Comment object by calling the form's `save()` method and assign it to the `new_comment` variable, but with `commit=Flase` which will prevent it from saving into the database right away because we still have to link it the post object
2. We assigned the comment object to the current post
3. Finally, save the object into the database

Else if it is a `GET` request we initialize the form object and pass it to the template.

Adding URL patterns for Views

Open the `urls.py` file of your app and map the view.

```
path('<slug:slug>/', views.post_detail, name='post_detail')
```

Creating Templates For The Views

Let's see what we will do in the templates.

```
{% for comment in comments %}
    <div class="comments" style="padding: 10px;">
      <p class="font-weight-bold">
        {{ comment.name }}
        <span class=" text-muted font-weight-normal">
          {{ comment.created_on }}
        </span>
      </p>
      {{ comment.body | linebreaks }}
    </div>
    {% endfor %}
```

Here we are using Django's `{% for %}` template tag for looping over comments, then for each comment object, we are displaying the user's name,creation date and the comment body.

```
<div class="card-body">
    {% if new_comment %}
```

```
        <div class="alert alert-success" role="alert">
          Your comment is awaiting moderation
        </div>
        {% else %}
        <h3>Leave a comment</h3>
        <form method="post" style="margin-top: 1.3em;">
          {{ comment_form.as_p }}
          {% csrf_token %}
          <button type="submit" class="btn btn-primary  btn-lg">Submit</bu
        </form>
        {% endif %}
      </div>
```

When a user makes a new comment we show them a message saying, “Your comment is awaiting moderation” else we render the form.

So putting the entire template all together we have this,

```
{% extends 'base.html' %} {% block content %}

<div class="container">
  <div class="row">
    <div class="col-md-8 card mb-4  mt-3 left  top">
      <div class="card-body">
        <h1>{% block title %} {{ post.title }} {% endblock title %}</h1>
        <p class=" text-muted">{{ post.author }} | {{ post.created_on }}</
        <p class="card-text ">{{ post.content | safe }}</p>
      </div>
    </div>
    {% block sidebar %} {% include 'sidebar.html' %} {% endblock sidebar %

    <div class="col-md-8 card mb-4  mt-3 ">
      <div class="card-body">
```

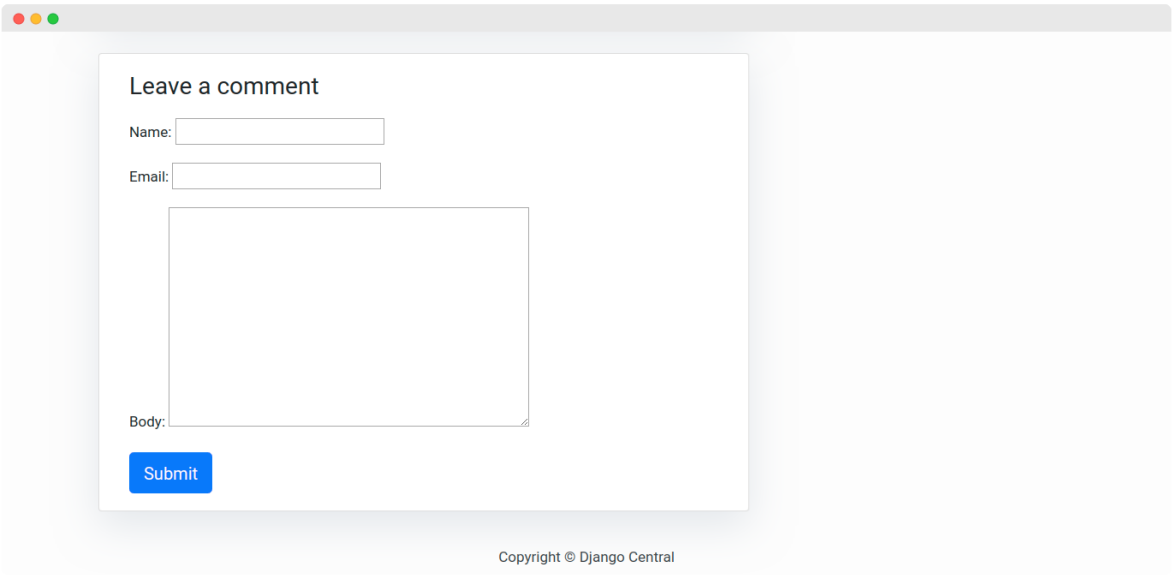
```
        <!-- comments -->
        <h2>{{ comments.count }} comments</h2>

        {% for comment in comments %}
        <div class="comments" style="padding: 10px;">
          <p class="font-weight-bold">
            {{ comment.name }}
            <span class=" text-muted font-weight-normal">
              {{ comment.created_on }}
            </span>
          </p>
          {{ comment.body | linebreaks }}
        </div>
        {% endfor %}
      </div>
    </div>
    <div class="col-md-8 card mb-4  mt-3 ">
      <div class="card-body">
        {% if new_comment %}
        <div class="alert alert-success" role="alert">
          Your comment is awaiting moderation
        </div>
        {% else %}
        <h3>Leave a comment</h3>
        <form method="post" style="margin-top: 1.3em;">
          {{ comment_form.as_p }}
          {% csrf_token %}
          <button type="submit" class="btn btn-primary  btn-lg">Submit</bu
        </form>
        {% endif %}
      </div>
    </div>
  </div>
</div>
{% endblock content %}
```

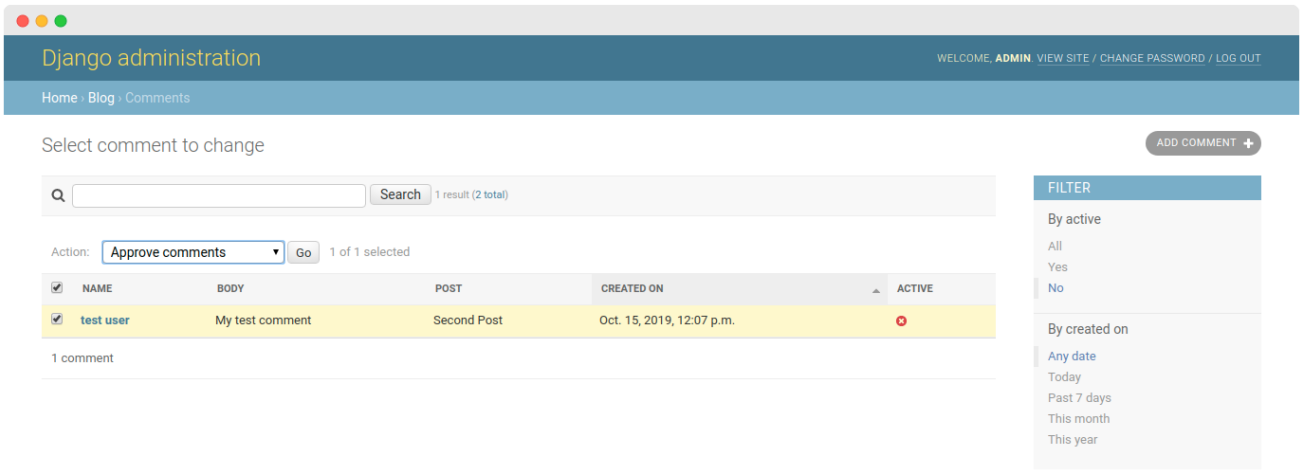
If you are working on a completely different project with some other CSS framework then ignore the styling.

Testing The Comment System

Save all the files and run the server and visit `http://127.0.0.1:8000/` and visit a post detail page.



Now create your comment and hit the submit button you should see the message. Now login to the Admin dashboard and approve the comment.



Making Comment Form Crispy

Although our form works as expected, yet we can make the form look better without much changing the template using the Django crispy form library. It's a very popular library for form managing you can check it out here – https://github.com/django-crispy-forms/django-crispy-forms

Install it using

```
pip install django-crispy-forms
```

Add it to the installed apps list.

```
INSTALLED_APPS = [
    ...
```

```
    'crispy_forms',
]
```

If you using Bootstrap 4 for styling add this in settings.py file.

```
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Now in your template, you just need to load the crispy tag and use the crispy tag beside the form, as follows.

```
{% load crispy_forms_tags %}
...
<form method="post" style="margin-top: 1.3em;">
          {{ comment_form | crispy }}
          {% csrf_token %}
          <button type="submit" class="btn btn-primary  btn-lg">Submit</bu
 </form>
```

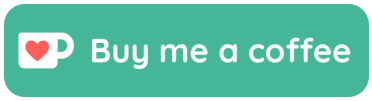
Save the files and run the server.

If you are stuck at any step, refer to this GitHub repo

Support Django Central

If you appreciate my work, or if it has helped you along your journey. It would mean a lot to me if you could write a message on my wall and

share a cup of coffee (or tea) with me.



95 thoughts on “Creating Comments System With Django”

Menja

November 2, 2019 at 9:02 am

Your tutorials are amazing, so THANK YOU for them!

I have an issue when I try to show the data from the post-detail and the comments.
When I click to see/read details on an article and its related comments, I only see the comments, the details of the article doesn’t show, but the page do not fail.

– I have tried just to call the model ‘LifePost’ and then it shows data in the template but not the related comments (only the fields for it).

– I have tried to call the method that use get_object_or_404 to call the model ‘LifePost’ and then it shows all the comments to the related article but not the article data (only the fields for it)

I don't get any error message and it doesn't fail, it shows the right comments related to the specific article. It just doesn't show the article content unless I call the model directly (and then I don't get the comments).
What am I doing wrong? ☺

Reply

Abhijeet

November 3, 2019 at 7:01 am

I recommend you to look at the Github repo.

Reply

Shahzod

March 22, 2021 at 7:46 am

aa good

Reply

Zianezo

November 6, 2019 at 1:54 pm

Thank you for the information and thank you for sharing it with us, Best Regards.

Reply

Dave.

November 22, 2019 at 8:21 am

Well done. Thanks for the information. Love the comments so far. Cheers. Dave.

Reply

Jiang

November 23, 2019 at 7:19 am

Thanks for your good work! It really helps me to learn the basics of Django! Have a nice day!

Reply

rakib

December 1, 2019 at 12:19 am

Very interesting post.this is my first-time visit here. I found so many interesting stuff in your blog especially its discussion..thanks for the post!

Reply

Fatima

December 5, 2019 at 9:21 am

I m so glad to visit this blog.This blog is really so amazing.Thanks for sharing with us.

Reply

UserName

December 17, 2019 at 10:10 pm

Help PLS, how to display the number of comments in index.html next to each post?

Reply

Thomas

December 18, 2019 at 4:19 pm

Hi, Admin!

Please help me figure it out. I made a blog on your guide, thank you for the material. Made comments on your article. I have a question. Please tell me how can I get the number of comments on a post in index.html? I insert {{comments.count}} into index.hmtl but nothing works for me. Help. Thanks in advance

Reply

Abhijeet

December 21, 2019 at 4:22 am

{{ post.comments.count}} will return the number of comments for a particular post.

Reply

Hoa L

December 28, 2020 at 6:48 pm

How about updating “post” model to include a comment counter. The post.comment_counter will be updated when a new comment is submitted. Do you think the performance will be better?

Reply

Abhijeet

December 29, 2020 at 5:18 am

Using count() seems a better option.

Reply

sepi

December 20, 2019 at 1:28 am

Thanks for the amazing tutorial.
When I try to save a comment it throws back this error: NOT NULL constraint failed, could you help me with this?

Reply

aryne

December 21, 2019 at 2:37 pm

Hi, Its is really a nice blog worth to read and share.

Reply

Thomas

December 22, 2019 at 12:32 pm

how to display the latest comments in index.html?

Reply

Abhijeet

December 23, 2019 at 10:27 am

In `post_detail` view change the comment query too `comments = post.comments.filter(active=True).order_by("-created_on")` this will fetch the latest comments first.

Reply

Thomas

December 23, 2019 at 6:34 pm

I want to display ALL the latest comments on all posts, for example the last 5 comments on the main page index.html?

Reply

Abhijeet

December 24, 2019 at 6:12 am

`comments = post.comments.filter(active=True).order_by("-created_on")[0:5]` will fetch the latest 5 comments.

Reply

Thomas

December 24, 2019 at 7:05 pm

thanks, how to display in the template itself?

Thomas

January 1, 2020 at 2:54 pm

Admin, help me please, I want to display the latest comments on posts on the main page. How can I do it ?

Reply

Vadym

April 23, 2020 at 5:12 pm

As well as you display the latest posts

Reply

Don

January 6, 2020 at 1:50 am

Thank you for the amazing tutorial, a quick question here:
When editing a comment, after pressing edit button/link, the page naturally refreshes itself, going to the top of the page instead of, what I hope for, staying right where it was… This isn't nice or friendly at all… Now I've added an id="{{ comment.pk }}" for each comment div based on their primary key. I'm probably v close to the solution, yet I'm not sure how to reference a div id in the view function…
Thank you sooo much!!

Reply

Abhijeet

January 7, 2020 at 8:33 am

You need to Ajaxify the forms for preventing browsers from reloading the page.

Reply

Manooj Kumar

February 17, 2020 at 9:30 am

First of all,I'm genuinely grateful for the particulars you shared with us which is very informative regarding the Global interaction among students and developers on Python.The tutorials were well organized

& met all the necessary courses for industry requirements of students.Second of all,the discussion part was interesting.Best Regards !

Reply

Mihion

February 22, 2020 at 12:21 pm

@admin.register(Comment) <——-Missing her but found it on github…
class CommentAdmin(admin.ModelAdmin):
list_display = ('name', 'body', 'post', 'created_on', 'active')

Thanks for a great tutorial!

Reply

Abhijeet

April 21, 2020 at 2:43 pm

Thanks!

Reply

user20

February 23, 2020 at 5:42 am

hey there, thanks alot for your material, I was wondering if do you have any idea how could I check on the comment? if it has a forbidden word in it? and turn it to (#$%@)?

Reply

Abhijeet

April 21, 2020 at 2:39 pm

You need a profanity algorithm in place for that.

Reply

Daniel Dibaba

November 27, 2020 at 7:30 pm

How can we add the reply or threaded comment?

Reply

Jeff

March 13, 2020 at 10:03 pm

Hi
This post will really help me with a project I'm working on – thank you for sharing.

What I am trying to do with my project is make it so that users can post comments against pictures. I have a Pictures model already in place, so would I simply replace 'Post' with 'Pictures' and 'post' with 'images'?

Also, I get the following warning message when I run python manage.py runserver:

WARNINGS:
?: (urls.W002) Your URL pattern '/' [name='image_detail'] has a route beginning with a '/'. Remove this slash as it is unnecessary. If this pattern is targeted in an include(), ensure the include() pattern has a trailing '/'.

Can I disregard this?

Thank you in advance.

Reply

Abhijeet

April 21, 2020 at 2:38 pm

In URL you have remove the trailing '/' also have you properly configured the media URLs?

Reply

mhw

March 15, 2020 at 7:23 pm

Thanks (again) for an amazing tutorial!
The comments system works perfectly, except when I try to use the crispy-form part where I get the error "LookupError: No installed app with label 'admin'" .

– I have used 'pip install django-crispy-forms'
– I have added 'crispy_forms' in INSTALLED_APPS and CRISPY_TEMPLATE_PACK = 'bootstrap4' in the settings file
– I have added {% load crispy_forms_tags %} in the detail html-file
– I have looked in your git-repository and placed all the tags at the right places

Do you have an idea why this error occurs?

Reply

Abhijeet

April 21, 2020 at 2:36 pm

Do you have ‘django.contrib.admin’ in installed apps?

Reply

Renata

March 17, 2020 at 8:38 pm

Nice work

Reply

vishal singh

April 18, 2020 at 7:23 pm

great man loved it

Reply

Vadym

April 21, 2020 at 3:24 pm

cool

Reply

RSean

April 25, 2020 at 12:51 am

This looks really good…very much appreciated!!! I'd like to request a tutorial demonstrating this comment system that we all are using to comment on this tutorial or possibly some direction as to how you created it? Thank you in advance.

Reply

Epaphradito Lugayavu

April 26, 2020 at 8:14 am

This is so wonerful.
Thanks. I have looking for this.

Reply

Divyansh Upadhyay

April 27, 2020 at 1:43 pm

I am getting error 405

Post method not allowed how can I solve it?

Reply

chrisnorman

April 27, 2020 at 10:52 pm

Thnks for this brief tutorial. really appreciate the code.

although i am having an error on the webpage

and it says something like this.

post_detail() missing 1 required positional argument: 'slug'

i believe because the function has the slug argument, but is never indicated in the url of the page.

please help me check it out.

Thanks in advance.

Reply

Abhijeet

April 28, 2020 at 3:30 am

Try passing id, or refer to the blog article https://djangocentral.com/building-a-blog-application-with-django/

Reply

Jnr

April 28, 2020 at 1:31 am

Please, help me out, I have carried out all the guidlines to create the comments but getting this error
“TypeError at /post_detail/
post_detail() missing 1 required positional argument: ‘slug'”

Reply

Abhijeet

April 28, 2020 at 3:28 am

You are not passing slug to the detail view either try sending id or refer to the blog article.

Reply

KhoiN

April 28, 2020 at 5:20 pm

Thanks for the tutorial. The comments system works perfectly. However, when I click the submit button, the comment form doesn't exist anymore. It's a little bit inconvenience that I have to reload the page to submit a new comment. How can I submit more than one comment without reloading the page?

Reply

Abhijeet

April 29, 2020 at 3:07 am

You need to make ajax call for that.

Reply

Nuser

May 1, 2020 at 6:06 pm

Thank you a lot for that tutorials! And i have a question,so in the result i have only “comments” “leave a comment” forms and “submit” button, there is no “name”,”email” and “body” forms. How can i solve this problem?

Reply

Abhijeet

May 2, 2020 at 5:02 am

Probably you have missed something in the templates.

Reply

Anas Ghrab

May 1, 2020 at 9:14 pm

The path has to be corrected :

path(“/”, views.post_detail, name=”post_detail”),

Reply

Abhijeet

May 2, 2020 at 5:00 am

Thanks for pointing it out!

Reply

Damian Marley

May 3, 2020 at 11:13 pm

Thanks for the information shared. This content is very explanatory and breaks the concept to smaller pieces… But am having a serious challenge because I already created my blog app before watching this tutorial to implement comment system in my blog. In my project, I used Classed Based Views and not Function Based Views.. I already established my DetailView route and it's up and running… But I can't actually use this code to create a comment system… Please how can I solve this problem? Thanks in advance.

Reply

Abhijeet

May 4, 2020 at 2:27 am

Handle the post request from `def post(self, request, *args, **kwargs)` method.

Reply

Theo

June 10, 2020 at 10:24 pm

Hi, thank you for the detailed explanation!

I also have the same issue as mentioned above. I'm new to Django so that's probably why I didn't fully understand your answer. What do you mean by 'handle the post request from def post(self, request, *args, **kwargs)'?

Should I change 'def post_detail(request, slug):' to 'def post(self, request, *args, **kwargs)'? Are there any other changes I am supposed to make to make sure it is handled properly? Do I leave my class based view for post_detail unchanged?

Thanks again for explaining!

Reply

Ekundayo folorunsho

June 3, 2020 at 2:11 pm

Please sir kindly help me with this am having an error when I paste the model class form which is comment I try to save it so that I can migrate but the error was
Post =models.ForeignKey(Post,on_delete=models.CASCADE
…..,……..
so the error am having at the downside is that
NameError : name Post is not define please help me how can I solve this problem I need this comment form

Reply

Kya Karna hai

September 3, 2020 at 1:57 pm

I am also facing the same issue!!

Reply

Amit

September 14, 2020 at 10:46 am

Please check your model file, if there is class Post.

If not please add one and then try again.

Reply

Devansh

September 15, 2020 at 3:53 pm

same issue

Reply

Sourabh Sarkar

November 2, 2020 at 4:40 pm

put ‘post’ instead of post

Reply

jeff

June 12, 2020 at 4:49 am

Is not working for me. i did something wrong may be in urls.py, views.py and changing my post name.
ok i have post is Destination and post detail is destview in destview i like to show the comments and make a comments form.
the problem is it did not show the comment and the form in destview.
when i click the Destination it go to home/destview/1 i can see the post and some text of leave comments, and submit button that's all
i can't see any comment and forms.

your help is appreciated
new to programming here

Reply

Bruce

June 12, 2020 at 4:33 pm

Thanks for a very nice share. I just have one point not really happy that, after submit comment, if i refresh the page, a duplicated comment also created. I did try redirect to the home page, but it doesnt make sense.
I also try to display only first 5 comments in total of 10 like this: comments = post.comments.filter(active=True).order_by('-created_on')[0:5]. But dont know how to show the total number of comments & read the remaining in post_detail.html.

BTW, how can you post the command format on this form, it looks so professional

Reply

Sylvain

June 14, 2020 at 7:12 pm

Hey, Thank you so much for this great tutorial. I set up the comment system exactly as you showed but when I try adding a comment, it gives me the error below:
Traceback (most recent call last):
File "/Users/……../Projects/lib/python3.8/site-packages/django/db/backends/utils.py", line 86, in _execute
return self.cursor.execute(sql, params)
File "/Users/……../Projects/lib/python3.8/site-packages/django/db/backends/sqlite3/base.py", line 396, in execute
return Database.Cursor.execute(self, query, params)

The above exception (NOT NULL constraint failed: my_site_comment.article_id) was the direct cause of the following exception:
File "/Users/…../Projects/lib/python3.8/site-packages/django/core/handlers/exception.py", line 34, in inner

response = get_response(request)

……..

Exception Type: IntegrityError at /article_detail/political-and-economic-implications-covid-19/5/
Exception Value: NOT NULL constraint failed: my_site_comment.article_id

Reply

Hanii

June 20, 2020 at 10:02 pm

Hello thanks for the comment system.I used this system but my forms not showing on template. P.S used Crispy forms.Thanks For the help.

Reply

Orchid Chetia Phukan

June 21, 2020 at 3:06 pm

Awesome tutorial, Sir i would be thankful if you help me with on How to make every comment visible in the post detail .

Reply

Justin S

June 26, 2020 at 8:33 pm

Love the tutorial, any ideas on how to do this with class based views? I dont really want to change everything to FBVs!

Reply

Gazi

July 1, 2020 at 10:19 am

Can You implement this function-base-view in class-based-view.

Reply

David

July 1, 2020 at 10:32 am

First of all, Thanks for this perfect tutorial. I would like to extend this comment system by implementing comment replies. What's the best way to handle this?

Reply

suman

July 8, 2020 at 8:47 am

I cann’t input the hiddenfield (primary key) and cann’t delete or update the comment

Reply

Nick

July 15, 2020 at 10:33 am

Nice article ! Thank you

Reply

Gerard Sánchez Vidal

July 23, 2020 at 9:14 pm

Hello, is this still working? Thank you, was very instructive anyways

Reply

mat shell

August 28, 2020 at 5:50 am

Hello,

It works very well,

How can I add to that replyr?

Thank you

Reply

BoJio

August 28, 2020 at 7:38 am

After submit the comment , the warning “Your comment is awaiting moderation” appeared.

At this moment i press “F5” to refresh , it will come out duplicate or multiples comment awaiting for approval.

May i know how to stop the duplicated or multiples comment awaiting for approval?

Reply

Jonas Slater

September 8, 2020 at 12:44 pm

This is awesome thank you, how would I go about adding the option to delete comments?

Reply

Devansh

September 15, 2020 at 3:52 pm

I am getting an error that says,
post = models.ForeignKey(Post,on_delete=models.CASCADE,related_name='comments')
NameError: name 'Post' is not defined

Reply

Abhijeet

September 24, 2020 at 5:19 am

I think you don’t have the Post model please check out the github repo.

Reply

Jyoti Rani

September 24, 2020 at 1:30 pm

Thank you for this amazing tutorial

Reply

Reza Faizi

September 30, 2020 at 5:51 am

Awesome tutorial. But how to configure replies in every comment, could you help me ?(in model and view) thank you so much.

Reply

darkhorse jo

October 22, 2020 at 4:27 pm

I noticed that in your blog, (this present one) you could reply comment, how can one achieve that?

Thanks again for the tutorial

Reply

Ranjan Nayak

October 30, 2020 at 1:25 am

Thank you so much. you helped me lots. Using your article I have created my first Django blog. Thank you again.

Reply

Clinton

November 4, 2020 at 4:28 pm

Your tutorial is great, please show us how to implement the comment form to be on the same page with blog detail templates.

Reply

Sigit Wijanarko

November 6, 2020 at 11:05 am

This is really great and awesome, I learned a lot. Thank you very much.

Reply

djangolearner

November 15, 2020 at 2:49 pm

how can we add like and dislike button? and awesome tutorial. thank you.

Reply

sripathi

December 24, 2020 at 7:36 am

Can you Please share me the Post model?

Reply

Abhijeet

December 25, 2020 at 5:16 am

Please look at the Github repo.

Reply

ali

December 31, 2020 at 7:42 am

Hello sir. I have a problem in migrations when I write the command python manage.py makemigrations it shows “no changes detected”. Please help me solve this problem.
Thank you!!

Reply

Djangolearner

December 31, 2020 at 2:55 pm

Thank you for this tutorial sir. But when i run after writting the codes for the comment section there is no name and email address box for the comment submit. The submit button is there but not the box where we are supposed to leave the comment. Kindly solve this

problem for me sir.
P.s- i have downloaded the github code for this comment form.
Thank you!!

Reply

izzun

January 10, 2021 at 4:39 am

thank you for tutorial sir. i am still confusing, how to use cbv comment system in django. can you give some reference for this problem.

Reply

Michael Romanov

January 17, 2021 at 3:07 pm

izzun: I have the same question. Trying to implement comment system with class-based views.

Dear author, is it possible to create a manual for constructing a comment system with CBV?

Reply

Anthonix

January 19, 2021 at 9:51 pm

Really helpful post am glad I came here

Reply

priya

January 20, 2021 at 6:53 pm

Nice but you make a post for reply also pls

Reply

mohiiip

February 8, 2021 at 7:49 am

I can not see my form table , why??

Reply

Tony Jays

March 7, 2021 at 3:36 pm

Job well done bro. Thanks a lot for this. It was quite helpful

Reply

Naman

March 10, 2021 at 7:04 am

In post admin table form the content section is not working saying 127.0.0.1 refused to connect. I checked every possible thing but not able to fix it so please provide the solution.

Reply

Paguro

March 11, 2021 at 10:23 am

well done!

Reply

Omar Ahmed

March 23, 2021 at 5:27 pm

very helpful article

thanks so much

Reply

Fala

April 5, 2021 at 8:20 am

How work with ajax?

Reply

Leave a Comment

Name *

Email *

Post Comment

About

Django Central is an educational site providing content on Python programming and web development to all the programmers and budding programmers across the internet. The main goal of this site is to provide quality tips, tricks, hacks, and other Programming resources that allows beginners to improve their skills.

Categories

Django

Programs

Python

Tools

Web Development

Pages

About Us

Contact Us

Disclaimer

Privacy Policy