DJ4E (https://www.dj4e.com)

Lessons (https://www.dj4e.com/lessons) Discussions (https://www.dj4e.com/discussions) My Progress (https://www.dj4e.com/assignments)

# Classified Ad Web Site - Milestone 2

Instructor 

In this assignment, you will expand your classified ads web site to add functionality equivalent to:

https://chucklist.dj4e.com/m2

The primary additions from the previous milestone are to add an image to each ad and add comments for each ad.

You will build this application by borrowing parts and pieces from the code that runs

https://samples.dj4e.com/

and combining them into a single application.

**Important Note:** If you find you have a problem saving files in the PythonAnywhere system using their browser-based editor, you might need to turn off your ad blocker for this site - weird but true.

## Adding Pictures to the Ads Application

In this section, you will pull bits and pieces of the `pics` sample application into your `ads` application to add support for an optional single picture per ad. If you already added pictures to your application in a previous assignment, you can just skip to the instructions to add comments to your application.

(1) Add this to your `ads/model.py` , taking inspiration from `dj4e-samples/pics/models.py`

```
class Ad(models.Model) :

    ...
    # Picture
    picture = models.BinaryField(null=True, editable=True)
    content_type = models.CharField(max_length=256, null=True, help_text='The MIMEType of the file')
    ...
```

Do not include the entire `Pic` model. Of course do the migrations once you have modified the model.

(2) Copy in the `pics/forms.py` as well as `pics/humanize.py` .

(3) Take a look at `pics/views.py` and adapt the patterns in `PicCreateView` and `PicUpdateView` and replace the code for `AdCreateView` and `AdUpdateView` in `ads/views.py`. These new views don't inherit from owner.py because they manage the `owner` column in the `get()` and `post()` methods.

(4) Alter your `templates/ads/ad_form.html` by looking through `pics/templates/pics/form.html`. Make sure to add the JavaScript bits at the end and add `enctype="multipart/form-data"` and the `id` attribute to the `form` tag.

(5) Alter the `templates/ads/ad_detail.html` template by looking through `pics/templates/pics/detail.html` and to add code to include the image in the output if there is an image associated with the ad. Make sure not to lose the `price` field in your UI. If you don't see the `price` field in your UI it is likely a mistake in your `forms.py`.

(6) Add an `ad_picture` route to your `urls.py` based on the `pics_picture` route from `pics/urls.py`:

```
path('ad_picture/<int:pk>', views.stream_file, name='ad_picture'),
```

(5) Add the `stream_file()` view from `pics/views.py` and adapt appropriately.

Test to make sure you can upload, view, and update pictures with your ads.

# Adding Comments to the Ads Application

In this section, you will pull bits and pieces of the `forum` sample application into your `ads` application to add support for comments for each ad.

(1) Update your `models.py` adding the comment feature from the `forums/models.py`

```
class Ad(models.Model) :
    ...
    owner = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    comments = models.ManyToManyField(settings.AUTH_USER_MODEL,
        through='Comment', related_name='comments_owned')
    ...

class Comment(models.Model) :
    text = models.TextField(
        validators=[MinLengthValidator(3, "Comment must be greater than 3 characters")]
    )

    ad = models.ForeignKey(Ad, on_delete=models.CASCADE)
    owner = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    # Shows up in the admin list
    def __str__(self):
        if len(self.text) < 15 : return self.text
        return self.text[:11] + ' ...'
```

Do not add the Forum model - simply connect the `Comment` model to the `Ad` model. Of course do the migrations once you have modified the model successfully.

(2) Pull the `CommentForm` class from `forums/forms.py` into your `forms.py` .

(3) Adapt the `get()` method from `ForumDetailView` to your `AdDetailView` to retrieve the list of comments and create the `CommentForm` and pass them into your `templates/ads/ad_detail.html` template through the context.

(4) Adapt the `templates/ads/ad_detail.html` template to show comments and show a delete icon when a comment belongs to the current logged in user.

(5) Also add the ability to add a comment to an ad in `ad_detail.html` when the user is logged in by looking at the techniques in `forums/templates/forums/detail.html` .

(6) Add a route in `urls.py` for the `ad_comment_create` and `ad_comment_delete` routes from `forums/urls.py` . Make sure to use the same URL patterns as shown here:

```
urlpatterns = [
    ...
    path('ad/<int:pk>/comment',
        views.CommentCreateView.as_view(), name='ad_comment_create'),
    path('comment/<int:pk>/delete',
        views.CommentDeleteView.as_view(success_url=reverse_lazy('ads:all')), name='ad_comment_delete'),
]
```

(7) Adapt the comment related views from `forums/views.py` and put them into your `views.py`.



(8) You will have to adapt the `forums/templates/forums/comment_delete.html` template to work in your ads application.



# Manual Testing



It is always a good idea to manually test your application before submitting it for grading. Here are a set of manual test steps:

- Make two accounts - If you have not already done so
- Log in to your application on the first account
- Create an ad with a picture
- In the all ads list make sure that the edit / delete button shows correctly
- View its details click on the picture to see that it fulls the screen
- Update the ad, check that the details are correct
- Delete the ad - just to make sure it works - the autograder gets grumpy if it cannot delete an ad
- Create two more ads
- Make two comments on an ad - make sure you can see the delete button on your comments
- Delete one of your comments
- Log in on the second account - make sure you **do not** see edit / delete buttons on the existing ads
- Go into one the ad you commented on above - make sure to see the comment from the first user and make sure there is **no** delete button
- Make two new comments - make sure you *do* see the delete button for your comments but not for the other user's comments
- Delete one of your comments and make sure it goes away

# Do Some or All of the Challenges

You will have to finish these by the next assignment - so you might as well work on them now. And they are fun.

(1) Make yourself a gravatar at https://en.gravatar.com/ - it is super easy and you will see your avatar when you log in in your application and elsewhere with gravatar enabled apps. The gravatar can be anything you like - it does not have to be a picture of you. The gravatar is associated with an email address so make sure to give an email address to the user you create with `createsuperuser`.

(2) Change your `home/static/favicon.ico` to a favicon of your own making. I made my favicon at https://favicon.io/favicon-generator/ - it might not change instantly after you update the favicon because they are cached extensively. Probably the best way to test is to go right to the favicon url after you update the file and press 'Refresh' and/or switch browsers. Sometimes the browser caching is "too effective" on a favicon so to force a real reload to check if the new favicon is really being served you can add a GET parameter to the

URL to force it to be re-retrieved:

```
https://chucklist.dj4e.com/favicon.ico?x=42
```

Change the `x` value to something else if you want to test over and over.

(3) Make social login work. Take a look at github_settings-dist.py, copy it into `mysite/mysite/github_settings.py` and go through the process on github to get your client ID and secret. The documentation is in comments of the file. Also take a look at dj4e-samples/urls.py and make sure that the "Switch to social login" code is correct and at the end of your `mysite/mysite/github_settings.py` .

You can register two applications with github - one on localhost and one on PythonAnywhere. If you are using github login on localhost - make sure that you register `http://127.0.0.1:8000/` instead of `http://localhost:8000/` and use that in your browser to test your site. If you use localhost, you probably will get the `The redirect_uri MUST match the registered callback URL for this application.` error message when you use social login.