Lab 9 Part 4

File/Image Uploads

Now that we have configured static assets such as images, we now want to look at user uploaded files, such as book covers, which are somewhat different. To start with, Django refers to the former as **static** whereas anything uploaded by a user, whether it be a file or an image, is referred to as **media**.

The process for adding this feature for files or images is similar, but for images the Python image processing library Pillow must be installed which includes additional features such as basic validation.

In Windows Command Line type the following command to install Pillow:

---

pip install pillow

Media Files

Fundamentally the difference between static and media files is that we can trust the former, but we can't trust the latter by default. There are always security concerns when dealing with user-uploaded content. Notably, it is important to validate all uploaded files to ensure they are what they say they are. There are a number of nasty ways a malicious actor can attack a website that blindly accepts user uploads.

To start let's add two new configurations to the **bookstore_project/settings.py** file.

By default, MEDIA_URL and MEDIA_ROOT are empty and not displayed so we need to configure them:

- MEDIA_ROOT is the absolute file system path to the directory for user-uploaded files
- MEDIA_URL is the URL we can use in our templates for the files

For convenience we will put both the static and media file configurations together so add both settings after STATICFILES_FINDERS near the bottom of the file. We'll use the common convention of calling both media. Don't forget to include the trailing slash /for MEDIA_URL!

Add the following two lines at the very end of **settings.py**:

```
134   MEDIA_URL = '/media/' # new
135   MEDIA_ROOT = str(BASE_DIR.joinpath('media')) # new
```

In VS Code create a new directory called **media** at the project level and a subdirectory called **covers** within it.

And finally, since user-uploaded content is assumed to exist in a production context, to see media items locally we need to update **bookstore_project/urls.py** to show the files locally. This involves importing both settings and static at the top and then adding an additional line at the bottom.

Open **booksproject/urls.py** and add in the following code:

```
16  from django.contrib import admin
17  from django.urls import path, include
18  from django.conf import settings
19  from django.conf.urls.static import static
20
21
22  urlpatterns = [
23      path('admin/', admin.site.urls),
24      path('books/', include('books.urls')),
25      path('accounts/', include('accounts.urls')),
26      path('accounts/', include('django.contrib.auth.urls')),
27      path('', include('pages.urls')),
28  ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Models

With our generic media configuration out of the way we can now turn to our models. To store these images, we will use Django's ImageField which comes with some basic image processing validation included.

Add the following new line of code to **models.py**

```
books > models.py > ...
1   import uuid
2   from django.db import models
3   from django.urls import reverse
4
5   # Create your models here.
6   class Book(models.Model):
7       id = models.UUIDField(primary_key=True,
8           default=uuid.uuid4,
9           editable=False)
10      title = models.CharField(max_length=200)
11      author = models.CharField(max_length=200)
12      price = models.DecimalField(max_digits=6, decimal_places=2)
13      date_publication = models.DateField(blank=False, null=False)
14      cover = models.ImageField(upload_to='covers/')
15
16      def __str__(self):
17          return self.title
18
19      def get_absolute_url(self):
20          return reverse('book_detail', args=[str(self.id)])
```

The name of the field is cover and we specify the location of the uploaded image will be in MEDIA_ROOT/covers (the MEDIA_ROOT part is implied based on our earlier **settings.py** config).

Since we have updated the model, it is time to create a migrations file. Run the following command:

---

python manage.py makemigrations books

When you run this command, you will see the following output:

```
(env) C:\Users\pmagee\djangoprojects\lab-9-upload-pmagee>python manage.py makemigrations books
It is impossible to add a non-nullable field 'cover' to book without specifying a default. This is
 because the database needs something to populate existing rows.
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with a null value for this col
umn)
 2) Quit and manually define a default value in models.py.
Select an option:
```

The reason for this output is that we are adding a new database field, but we already have three entries in our database for each book. Yet we failed to set a default value for cover. To fix this, type 2 to quit and we will add a blank field set to True for existing images.

books > models.py

```
14      cover = models.ImageField(upload_to='covers/', blank=True) # new
```

Now we can create a migrations file without errors.

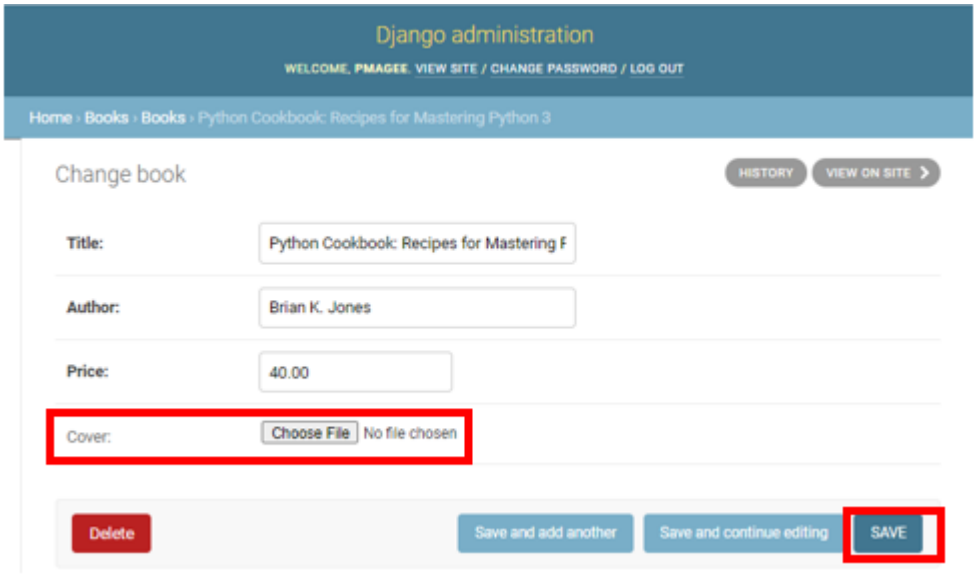
---

python manage.py makemigrations books

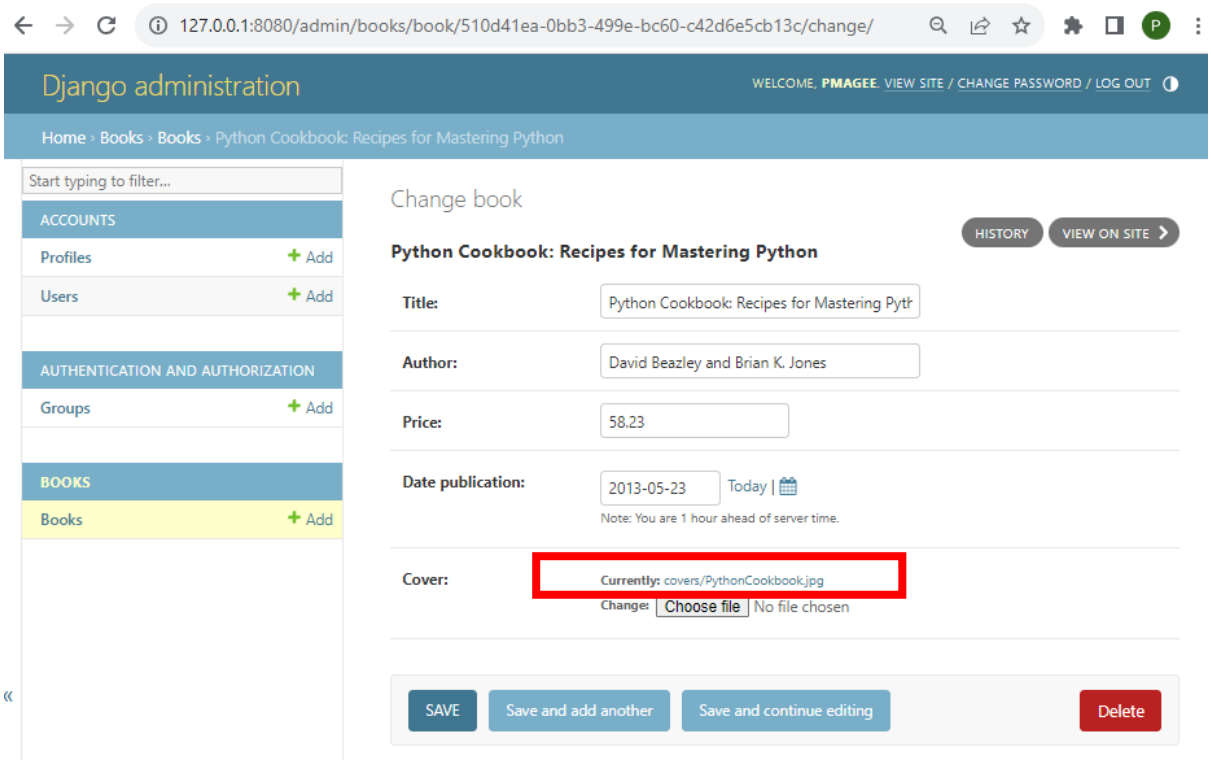
python manage.py migrate

Admin

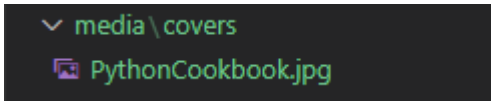
Run the server and log in to Django Admin and navigate over to the entry for the book “Python Cookbook.”. Select the “Choose File” option and navigate to the images folder that you downloaded from Moodle and select the appropriate image for the upload and then click the “Save” button in bottom right.



This will redirect back to the main Books section. Click on the link again for “Python Cookbook” and we can see it currently exists in our desired location of covers/.



In VS Code you should also see the image in the **media/covers** folder:



Template

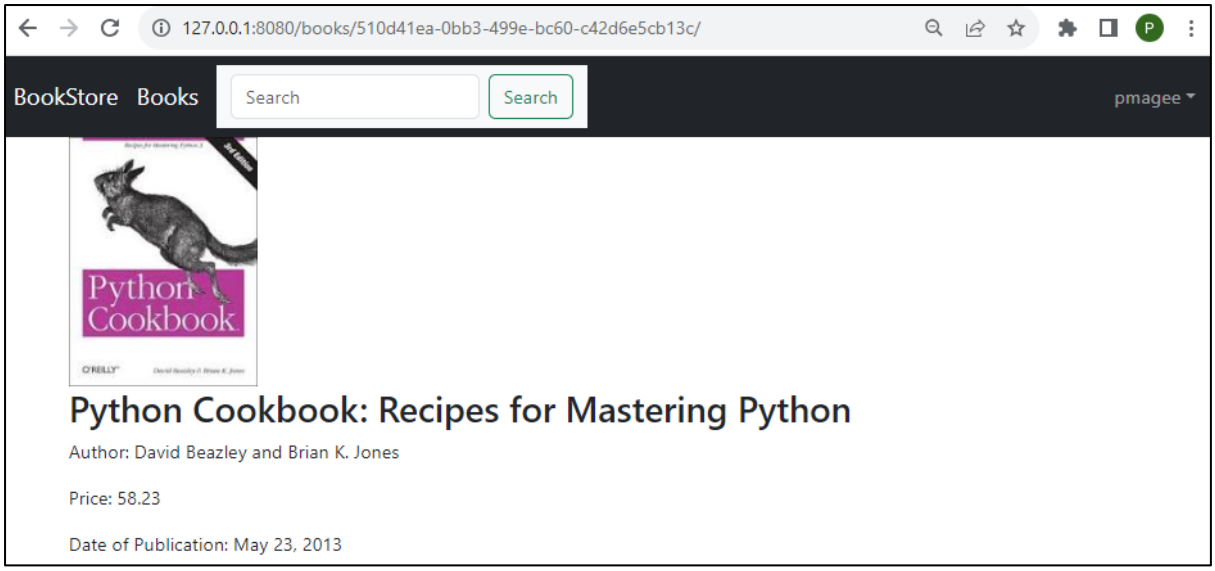
We need to update our template to display the book cover on the individual page. The route will be **book.cover.url** pointing to the location of the cover in our file system.

Add the following code to **book_detail.html** above the title.

templates > books > book_detail.html > ...

```
1   {% extends 'base.html' %}
2
3   {% block title %}{{ book.title }}{% endblock title %}
4
5   {% block content %}
6     <div class="book-detail">
7       {% if book.cover %}
8         <img class="bookcover" src="{{ book.cover.url}}" alt="{{ book.title }}">
9       {% endif %}
10      <h2><a href="">{{ book.title }}</a></h2>
11      <p>Author: {{ book.author }}</p>
12      <p>Price: {{ book.price }}</p>
13    </div>
14  {% endblock content %}
```

If you now visit the page for “Python Cookbook” you’ll see the cover image as shown below!



Go back into Django Admin and add the image covers for the other two books. Check that these images are displaying ok on the book_detail web pages.

Stop the server and run the following git commands to update the local and remote repositories:

git add -A

git commit -m “lab 9 part 4 commit”

git push -u origin main