

Django Practice

Creating a database of Music Albums

Setup Project

- Create a new folder within our development folder
 - `mkdir musicManager`
- Create a new virtual environment
 - `python3 -m venv venv`
 - `source venv/bin/activate`
- Install Django
 - `pip install Django`
- Create a new Django Project
 - `django-admin startproject musicMan`

```
adminuser@AdminVirt:~/dev$ mkdir musicManager
adminuser@AdminVirt:~/dev$ ls
forClass musicManager python-sms
adminuser@AdminVirt:~/dev$ cd musicManager/
adminuser@AdminVirt:~/dev/musicManager$ python3 -m venv venv
adminuser@AdminVirt:~/dev/musicManager$ ls
venv
adminuser@AdminVirt:~/dev/musicManager$ source venv/bin/activate
(venv) adminuser@AdminVirt:~/dev/musicManager$ pip install Django
Collecting Django
  Using cached Django-2.0.3-py3-none-any.whl
Collecting pytz (from Django)
  Using cached pytz-2018.3-py2.py3-none-any.whl
(venv) adminuser@AdminVirt:~/dev/musicManager$ django-admin startproject musicMan
(venv) adminuser@AdminVirt:~/dev/musicManager$ ls
musicMan venv
(venv) adminuser@AdminVirt:~/dev/musicManager$ cd musicMan/
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ code .
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$
```

Setup Project

- Open Visual Studio Code for the project
 - `cd musicMan`
 - `code`
- Run server to test
 - `python manage.py runserver`
 - Run local <http://127.0.0.1:8000/>
- End server via control-c

```
(venv) adminuser@AdminVirt:~/dev/musicManager$ cd musicMan/  
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ code .  
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py runserver  
Performing system checks...
```

System check identified no issues (0 silenced).

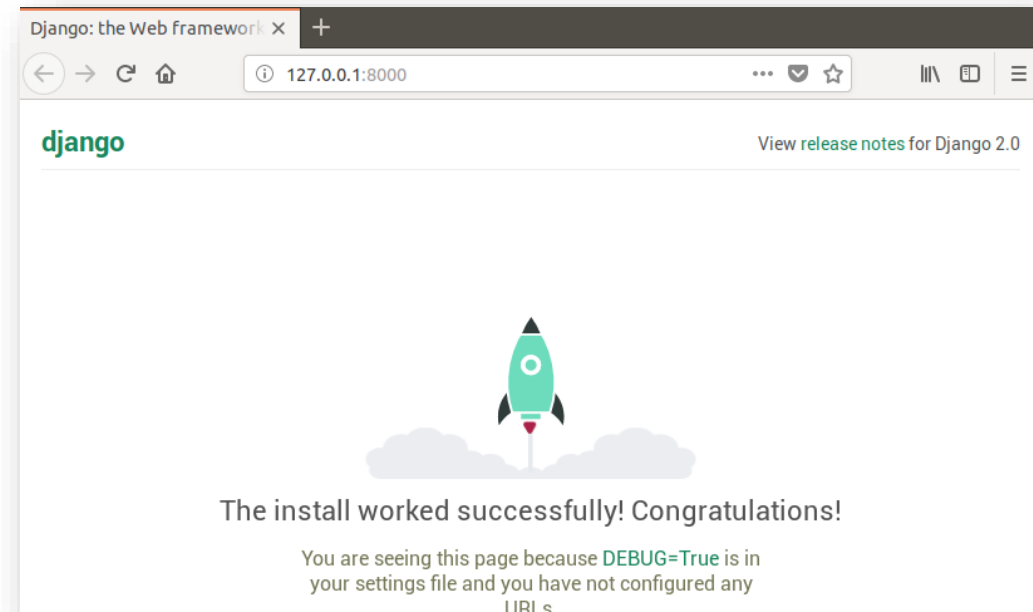
You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

March 21, 2018 - 02:38:37

Django version 2.0.3, using settings 'musicMan.settings'

Starting development server at http://127.0.0.1:8000/

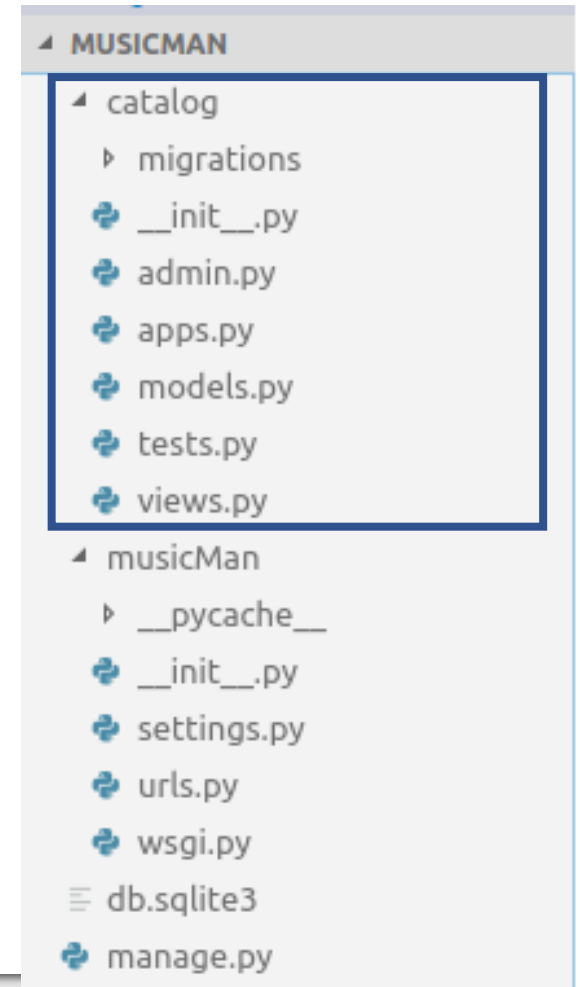
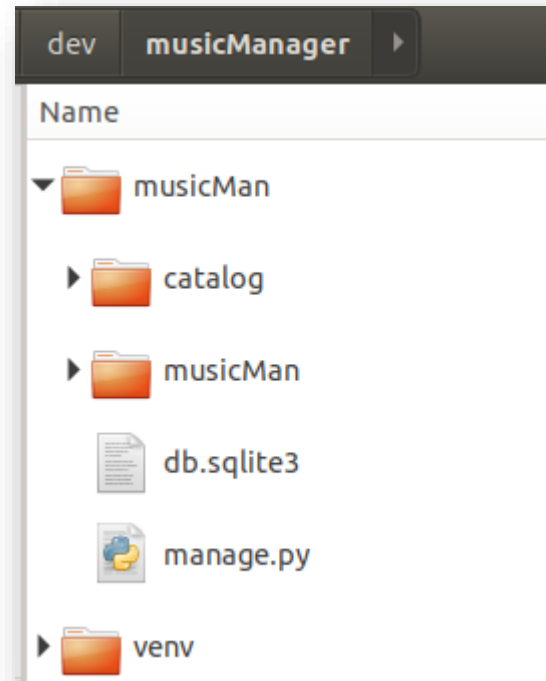
Quit the server with CONTROL-C.



Setup Project

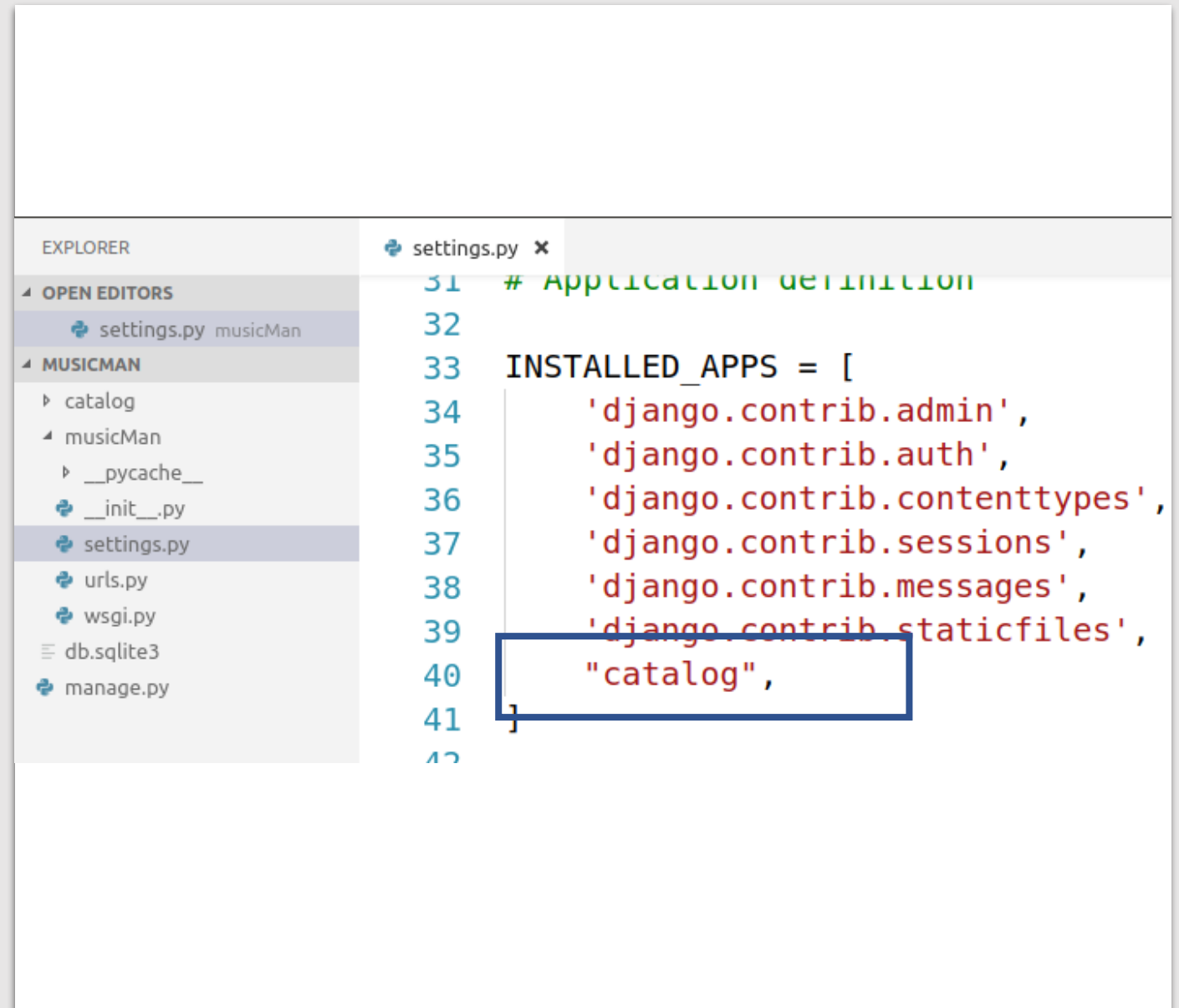
- Create application within project
 - A project can have multiple applications
 - In this example, our project “musicMan” will have an application called “catalog” (NB... catalogue is deliberately misspelt for convenience)
 - `python manage.py startapp catalog`
- Side Note (from documentation):
 - Projects vs. apps
 - What’s the difference between a project and an app?
 - An app is a Web application that does something – e.g., a Weblog system, a database of public records or a simple poll app.
 - A project is a collection of configuration and apps for a particular website.
 - A project can contain multiple apps. An app can be in multiple projects

```
[21/Mar/2018 02:40:14] GET /favicon.ico HTTP/1.1 404 1974
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py startapp catalog
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ ls
catalog  db.sqlite3  manage.py  musicMan
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$
```



Setup Project

- Add created application to the list of installed apps in “musicManager/musicMan/settings.py” file



```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     "catalog",
41 ]
```

Routes and Views

Views

- Create a simple HTTP response within the catalog view
- Create a urls.py file to specify routes for the catalog application

The image shows two screenshots of a VS Code editor. The top screenshot shows the `views.py` file in the `catalog` app, with imports for `render` and `HttpResponse`, and a simple `index` view function. The bottom screenshot shows the `urls.py` file in the `catalog` app, with imports for `path` and `index`, and a `urlpatterns` list containing a route for the `index` view.

Top Screenshot: `views.py`

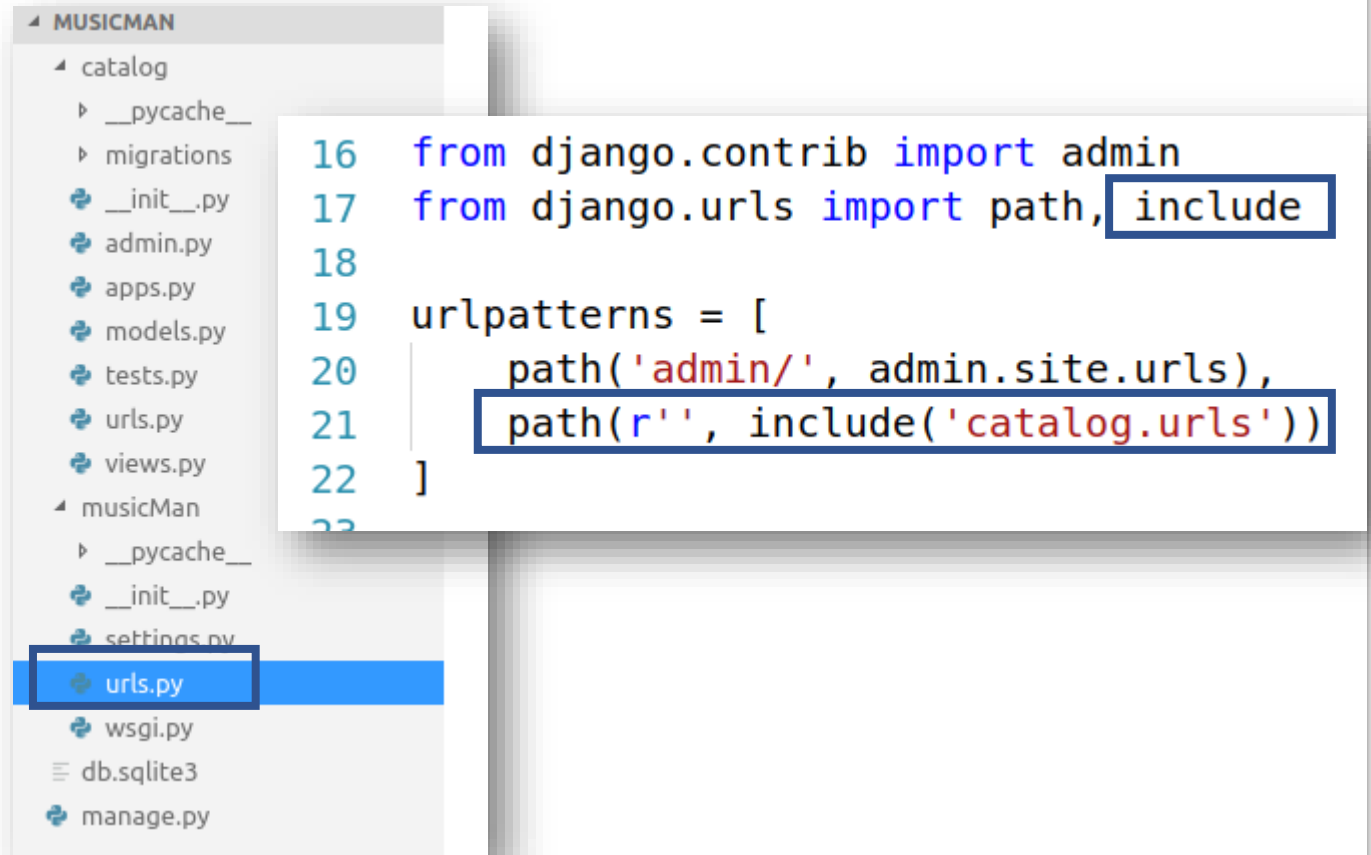
```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5 def index(request):
6     return HttpResponse("Within the Catalog")
7
```

Bottom Screenshot: `urls.py`

```
1 from django.urls import path
2 from .views import index
3
4 urlpatterns = [
5     path('', index),
6 ]
```

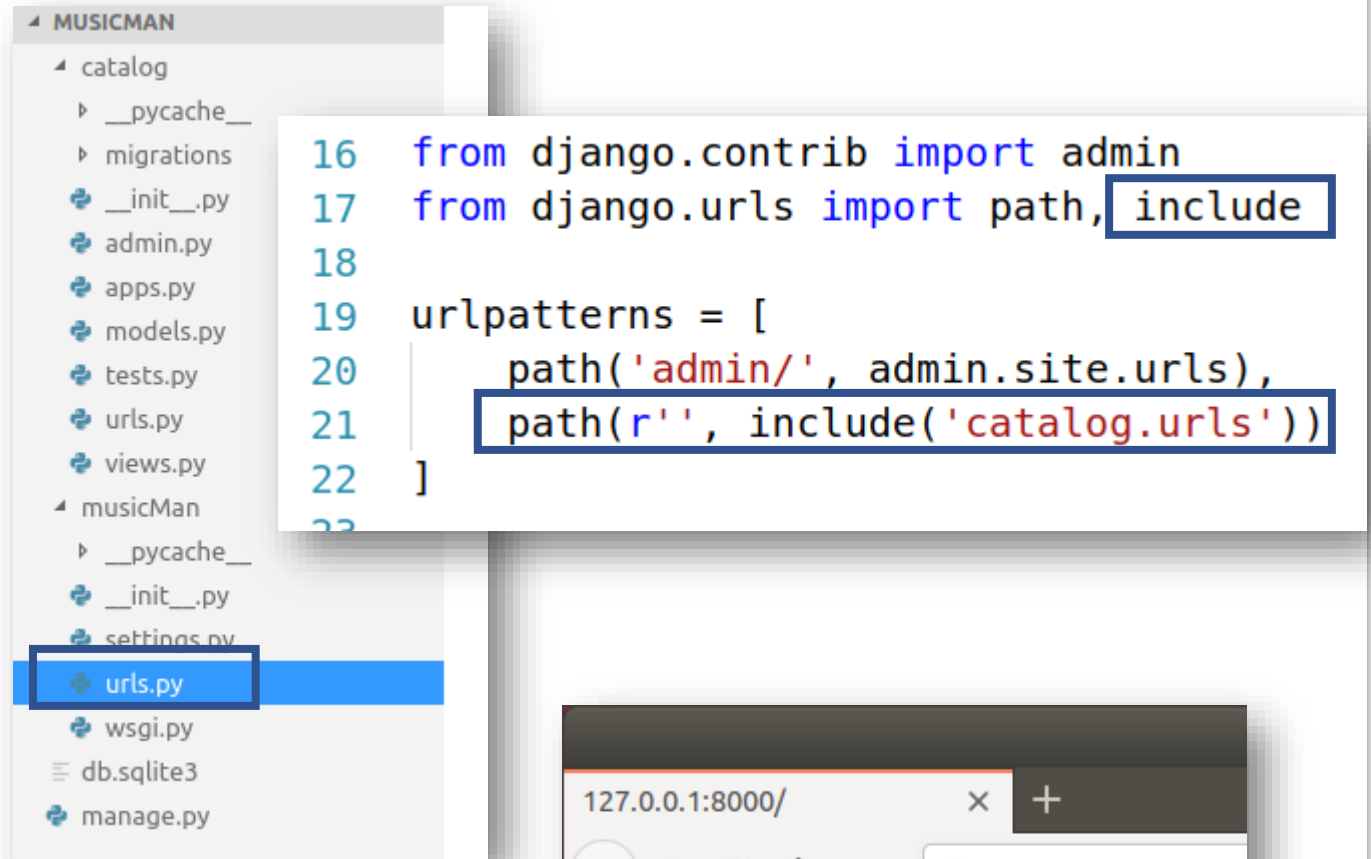
Views

- Configure the routes of the project to use the routes of the application
 - i.e. let the urls.py file in the musicMan utilize the urls.py in the catalog



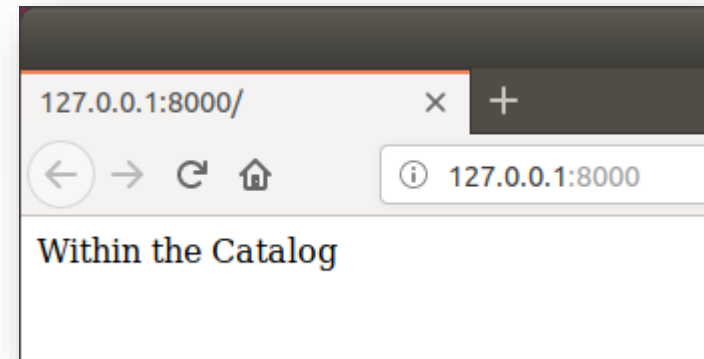
Views

- Configure the routes of the project to use the routes of the application
 - i.e. let the urls.py file in the musicMan utilize the urls.py in the catalog
- Run server and test the URL to view the http response generated
- Can you trace how the routes will load the appropriate view?



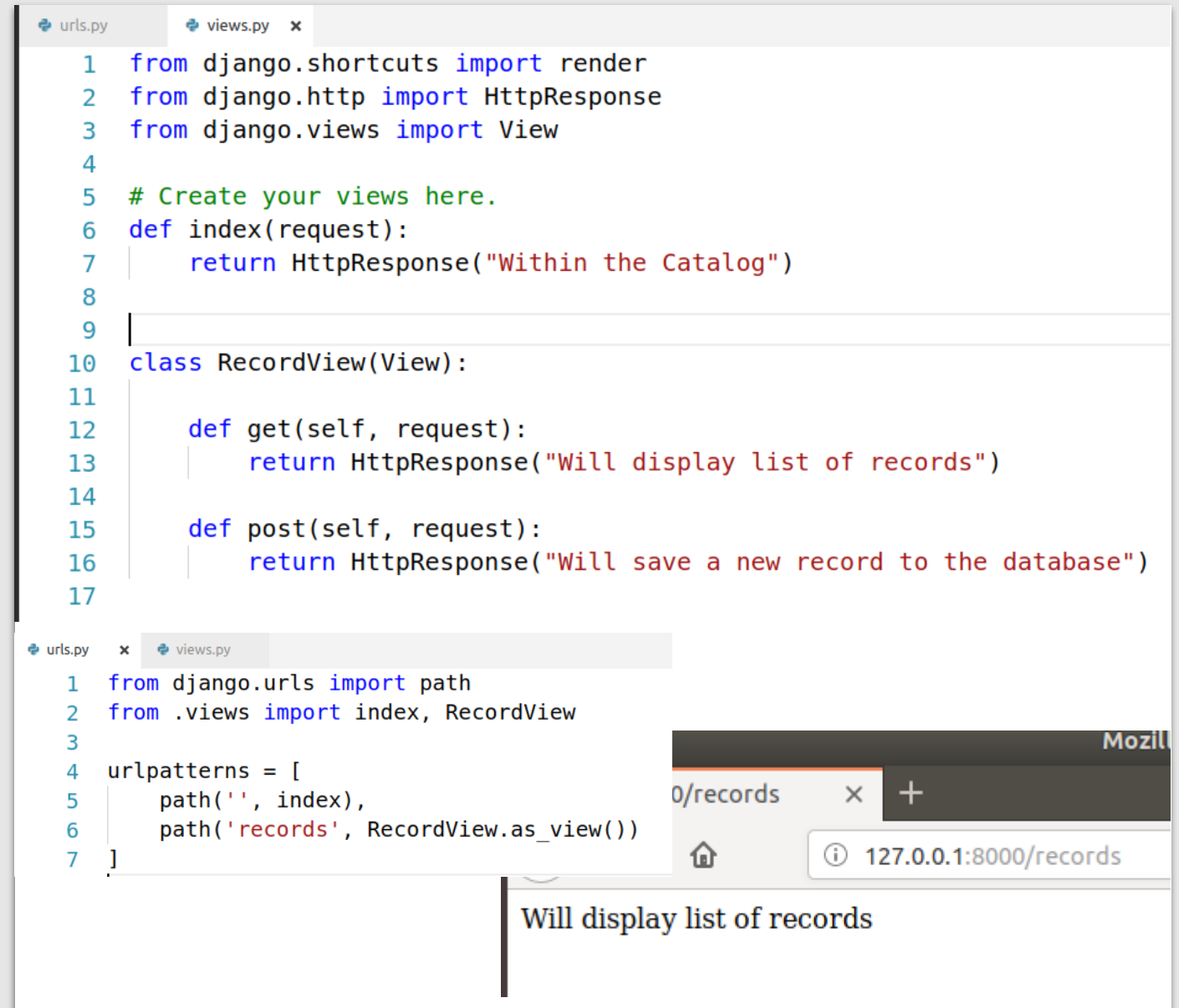
The image shows a file explorer on the left for a project named 'MUSICMAN'. The 'catalog' directory is expanded, showing files like '__pycache__', 'migrations', '__init__.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py', and 'views.py'. The 'musicMan' directory is also expanded, showing '__pycache__', '__init__.py', 'settings.py', 'urls.py' (highlighted with a blue box), 'wsgi.py', 'db.sqlite3', and 'manage.py'. To the right, a code snippet from 'urls.py' is shown with line numbers 16 to 22. The code imports 'admin' from 'django.contrib' and 'path', 'include' from 'django.urls'. It then defines 'urlpatterns' as a list containing 'path('admin/', admin.site.urls)' and 'path(r'', include('catalog.urls'))'. The 'include' function and the 'catalog.urls' string are highlighted with blue boxes.

```
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path(r'', include('catalog.urls'))
22 ]
```



Views

- Recall from the lecture that views can be specified using either functions or classes
- The following is an example of a [class-based view](#)
- Configure the view class to be utilized for request on the url “records”



The image shows a code editor with two files: `urls.py` and `views.py`. The `views.py` file contains the following code:

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3 from django.views import View
4
5 # Create your views here.
6 def index(request):
7     return HttpResponse("Within the Catalog")
8
9
10 class RecordView(View):
11
12     def get(self, request):
13         return HttpResponse("Will display list of records")
14
15     def post(self, request):
16         return HttpResponse("Will save a new record to the database")
17
```

The `urls.py` file contains the following code:

```
1 from django.urls import path
2 from .views import index, RecordView
3
4 urlpatterns = [
5     path('', index),
6     path('records', RecordView.as_view())
7 ]
```

Below the code editor, a web browser window is visible. The address bar shows the URL `127.0.0.1:8000/records`. The browser page displays the text "Will display list of records".

Models

Models

- python manage.py makemigrations
- If you attempt to make the migrations to setup the database you will encounter an error.

```
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py makemigrations
SystemCheckError: System check identified some issues:

ERRORS:
catalog.Catalog.album_photo: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.python.org/pypi/Pillow or run command "pip install Pillow".
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$
```

- Resolve the error by installing the required python library

- pip install Pillow

```
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ pip install Pillow
Collecting Pillow
  Using cached Pillow-5.0.0-cp36-cp36m-manylinux1_x86_64.whl
Installing collected packages: Pillow
Successfully installed Pillow-5.0.0
```

- Then run the make migration command again

```
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py makemigrations
Migrations for 'catalog':
  catalog/migrations/0001_initial.py
    - Create model Catalog
```

Models

- Apply the migrations
 - `python manage.py migrate`

```
File Edit View Search Terminal Help
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, catalog, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying catalog.0001_initial... OK
  Applying sessions.0001_initial... OK
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$
```

Models

- Register Model with the admin
- Create the super user
 - `python manage.py createsuperuser`
- Log in to the administrators interface and created a Record

```
urls.py  models.py  admin.py  views.py
1  from django.contrib import admin
2
3  # Register your models here.
4  from .models import Catalog
5  |
6  admin.site.register(Catalog)
7
```

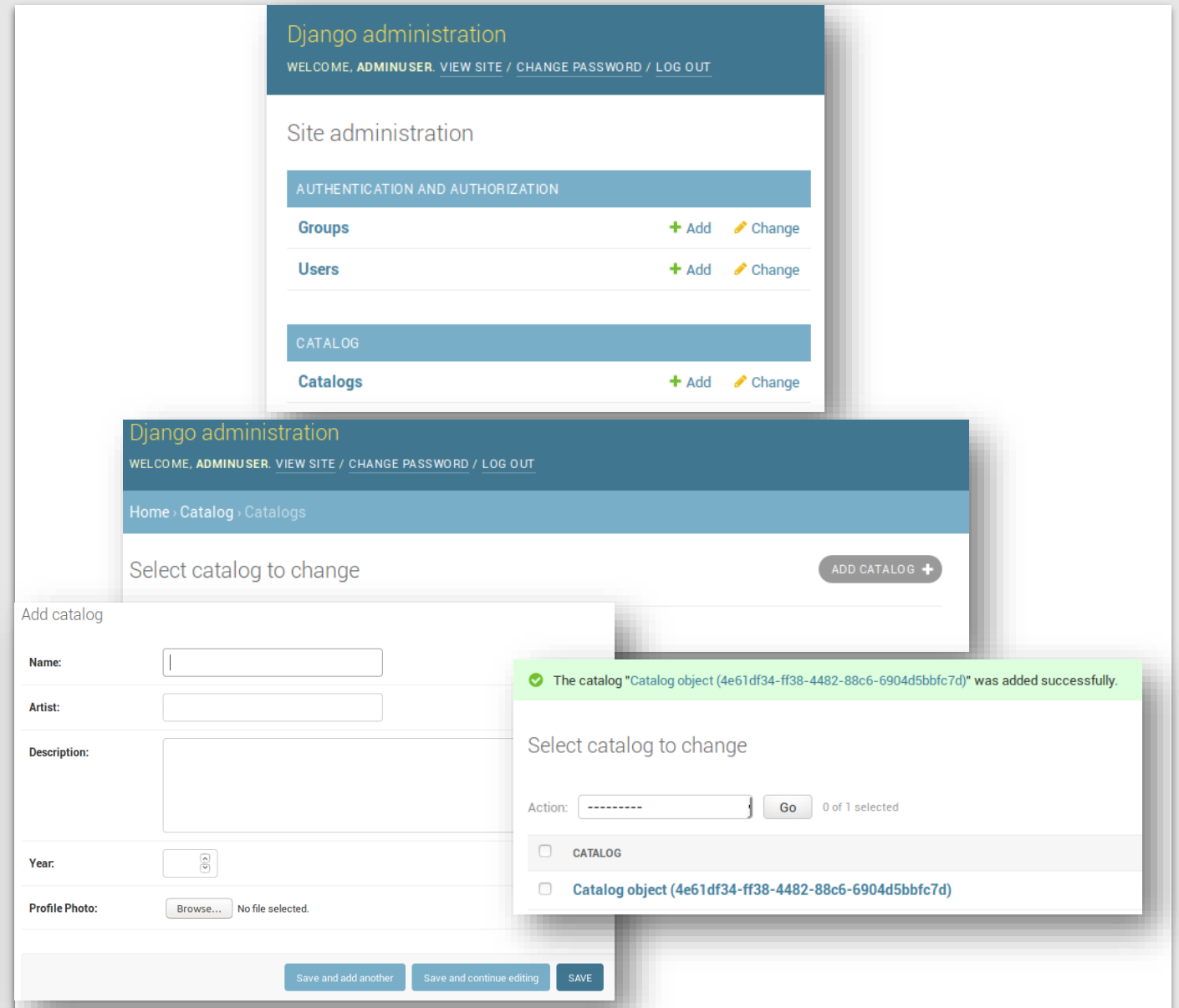
```
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py createsuperuser
Username (leave blank to use 'adminuser'):
Email address:
Password:
Password (again):
Superuser created successfully.
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$
```

```
(venv) adminuser@AdminVirt:~/dev/musicManager/musicMan$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
March 21, 2018 - 03:53:51
Django version 2.0.3, using settings 'musicMan.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Models

- Register Model with the admin
- Create the super user
 - `python manage.py createsuperuser`
- Log in to the administrators interface and created a Record



Models

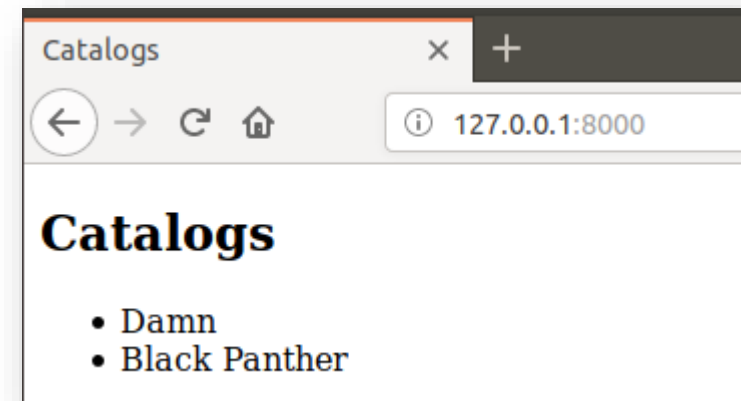
- Create a template for catalog
- Add the template to view records as a list
- Use view to retrieve records from database through ORM and display within the template through rendering as a HTTP response

```
urls.py  models.py  list.html  views.py
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport"
6          content="width=device-width,">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge"
8      <title>Catalogs</title>
9  </head>
10 <body>
11     <h2>Catalogs</h2>
12     <ul>
13         {% for rec in catalogs %}
14         <li>{{ rec.name }}</li>
15         {% endfor %}
16     </ul>
17 </body>
18 </html>
```

Models

- Create a template for catalog
- Add the template to view records as a list
- Use view to retrieve records from database through ORM and display within the template through rendering as a HTTP response

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3 from django.views import View
4
5 # Create your views here.
6 def index(request):
7     allRecords = Catalog.objects.all()
8     return render(request, 'catalogs/list.html', {
9         'catalogs': allRecords
10    })
```



ModelForms

Models

- Create a template for catalog
- Add the template to view records as a list
- Use view to retrieve records from database through ORM and display within the template through rendering as a HTTP response

EXPLORER

OPEN EDITORS

forms.py catalog

MUSICMAN

catalog

▸ __pycache__

▸ migrations

▸ templates

__init__.py

admin.py

apps.py

forms.py

models.py

forms.py x

```
1 from django.forms import ModelForm
2 from .models import Catalog
3
4 class CatalogForm(ModelForm):
5     class Meta:
6         model = Catalog
7         exclude = ()
8
```

Models

- Create a template for catalog
- Add the template to view records as a list
- Use view to retrieve records from database through ORM and display within the template through rendering as a HTTP response

```
15 from django.http import HttpResponseRedirect
16
17 class RecordView(View):
18
19     def get(self, request):
20         catForm = CatalogForm()
21         return render(request, 'catalogs/add.html', {
22             'form' : catForm
23         })
24
25     def post(self, request):
26         catForm = CatalogForm(request.POST, request.FILES)
27         if catForm.is_valid():
28             catForm.save()
29             return HttpResponseRedirect("/?oper=cat&res=true")
30         # Else request was invalid
31         return render(request, 'catalogs/add.html', {
32             'form' : catForm
33         })
34
```

Models

- Create a template for catalog
- Add the template to view records as a list
- Use view to retrieve records from database through ORM and display within the template through rendering as a HTTP response

New Catalog

Name:

Artist:

Description:

Year:

Album Photo: No file selected.

