Introduction to Database Systems: CS312 Django Detailed Views by HTML, Part3

Oliver Bonham-Carter

8 March 2019

Django

An easy-to-create web site and online database server

- `https://www.djangoproject.com/`

Learn More About Django!

Blog Server by the "Django Girls" Tutorial

- https://tutorial.djangogirls.org/en/

Steps to Begin a Project

Prepare somewhere to place your project

- mkdir djangoWorking
- cd djangoWorking

Create a new `virtualenv` (the virtual environment)

```
virtualenv myenv -p python3.6
source myenv/bin/activate # activate virtualenv
```

Install Your `virtualenv`

```
pip install django
```

Steps to Begin a Project

Create your Django project!

```
django-admin startproject mysite
```

Use manage.py to run the webserver to see your project!

```
cd mysite/
# we are now in: mysite/
python manage.py runserver
```

Test the local web server from browser

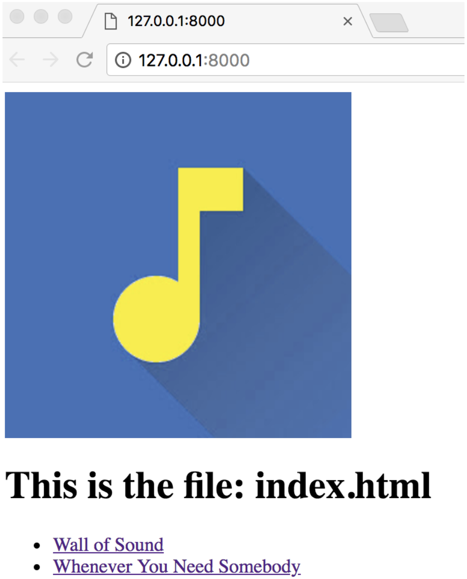
- http://127.0.0.1:8000/

Show Database On a Webpage

To Do Today

- Add to the Music App
  - Make our database display information to pages
  - Want to have the domain look like: `https://127.0.0.1:8000/music/`x
  - Note: x is the ID of the album
  - Use a template: index.html to show outputs from queries

To Make: An online music album viewer



This is the file: index.html

- Wall of Sound
- Whenever You Need Somebody

Refine the URL Scanner

- Regular expressions for patterns in text.
- Accepts an arbitrary URL address of pattern:
  - /music/712/
- "712" is a single number that is queried in DB (primary keys)

A line from `music/urls.py`

```
url(r'^(?P<album_id>[0-9]+)$', views.detail,
    name = 'detail'),
```

- <album_id> is a variable
- [0-9] is a number between 0 and 9
- The '+' means that the number may be longer than a single digit similar to the '*' in 'sigma$\hat{*}$'
- The '/' is a part of the URL itself and we ignore this character

Add URL support
Same as before ...

File: `mysite/urls.py`

```
from django.conf.urls import url, include
from django.contrib import admin

urlpatterns = [
        #url(r'', admin.site.urls),
        url(r'', include('music.urls')),#default page
        url(r'^admin/', admin.site.urls),
        url(r'^music/', include('music.urls')),]
```

Your file should look like this

Add URL support
Updated

File: `music/urls.py`

```
from django.conf.urls import include, url
# pull the local views.py file from local dir
from . import views

urlpatterns = [
        # pattern to use: music/X/
        url(r'^(?P<album_id>[0-9]+)/$', views.detail, name = 'detail'),
        url(r'^$', views.index, name = 'index'),]
```

Your file should look like this

Update Views
Updated

File: `music/views.py`

```
#####
from django.http import Http404 # for missing albums
from django.shortcuts import render
from django.template import loader
from .models import Album
#
def index(request):
        all_albums = Album.objects.all()
        template = loader.get_template('music/index.html')
        return render(request, 'music/index.html',{'all_albums':all_albums})
# check for valid film_ids.
#
#
# check for valid album_ids.
def detail(request, album_id):
        try: # check to see that the album exists
                album_current = Album.objects.get(pk=album_id)
        except Album.DoesNotExist:
                #display this message
                raise Http404("Sorry to say but the album does not exist.")
        return render(request, 'music/detail.html', {'album' : album_current})
```

- Album.objects.all() is the collection of records in Album

music/models.py

Same as before ...

```
from django.db import models
class Album(models.Model):
    #The following is a method of Album class
    def __str__(self):
    # show the album_title, artist
        return self.album_title+ ' - ' + self.artist
    #end of __str__()

    # holds the name of max length 250 chars
    artist = models.CharField(max_length = 250)
    # holds album name
    album_title = models.CharField(max_length = 500)
    # holds the genre
    genre = models.CharField(max_length = 100)
    #holds a url for music logo (link to graphic)
    album_logo = models.CharField(max_length = 1000)
#end of class Album()

class Song(models.Model):
    def __str__(self):
        return self.song_title
    #Class links the songs to the album class
    # foreign keys link the songs to a particular album.
    # when you delete an album, remove its
    # associated songs, as well.
    album = models.ForeignKey(Album,on_delete=models.CASCADE) # all on one line
    # holds the type of file containing music
    file_type = models.CharField(max_length = 10)
    # holds the song title.
    song_title = models.CharField(max_length = 250)
# end of class Song ()
```

Check the Settings

Same as before ...

- The music app must work with a connected database.
- We add a line to the *INSTALLED_APPS* in *mysite/settings.py* to make this connection.

Edit: `mysite/settings.py`

```
INSTALLED_APPS = [
    # we have added this top line to the rest below
    'music.apps.MusicConfig', # Link Music App to DB
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Migrations Setup

- Error: *You have xx unapplied migration(s)...*
- After the changes, connection to databases must be built and made. (Use *makemigrations* for this)
- We need to install these tables. (Use *migrate* for this)

mysite/manage.py

```
python manage.py migrate
python manage.py makemigrations music
```

Output

```
Migrations for 'music':
  music/migrations/0001_initial.py:
    - Create model Album
    - Create model Song
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK ...
```

What does our database schema look like?

```
./manage.py sqlmigrate music 0001
```

To see titles from shell, add to music/models.py

```
 # Add to Album Class
def __str__(self):
     # show the album_title, artist
     return self.album_title+ ' - ' + self.artist
#end of __str__()
```

To see titles from shell, add to music/models.py

```
 # Add to Song Class
 def __str__(self):
     return self.song_title
```

Create a Super User to View Database

- Need to make a user for the site.

```
python manage.py createsuperuser
```

- Username (leave blank to use 'user'): admin
- Email address: obonhamcarter@allegheny.edu
- Password: *"pass1234"*
- Password (again): *"pass1234"*
- Superuser created successfully.

Plug in (Register) the Databases

Just as before ...

- Add the *Albums* database
- Then, add the *Songs* database

File: `music/admin.py`

```
from django.contrib import admin
from .models import Album, Song
#
admin.site.register(Album)
admin.site.register(Song)
```

Adding Albums

- We are adding tuples into the shell to begin our database.
- We enter the data using objects.

python manage.py shell

```
# Add an album: Royksopp
#
from music.models import Album, Song
a = Album(artist="Royksopp",
album_title ="Wall of Sound",
genre="Electronic",
album_logo="http://www.cs.allegheny.edu/sites/obonhamcarter/cs380/graphics/royksopp.jpg")
#Above line: all on one line
a.save()
#
# add a song for the album
s = Song()
s.album_id = 1
s.Album = "Royksopp"
s.song_title = "What Else is there"
s.file_type="mp3"
s.save()
```

All songs have own ID

```
a.id # show the ID (primary key)
```

An Important Song To Add
“Never Gonna Give You Up”

An important link

`https://www.youtube.com/watch?v=dQw4w9WgXcQ`

Important to Add
"Never Gonna Give You Up"

python manage.py shell

```
#Add an album: Rick Astley
#
from music.models import Album, Song
a = Album(artist="Rick Astley",
album_title ="Whenever You Need Somebody",
genre="rock",
album_logo="http://www.cs.allegheny.edu/sites/obonhamcarter/cs380/graphics/rick.jpg")
#Above line: all on one line
a.save()
#
# add a song for the album
s = Song()
s.album_id = 2
s.Album = "Whenever You Need Somebody"
s.song_title = "Never Gonna Give You Up"
s.file_type="mp3"
s.save()
```

Checking-In On the Database

./manage shell

```
from music.models import Album, Song

Album.objects.all()
#<QuerySet [<Album: Wall of Sound - Royksopp>,
<Album: Whenever You Need Somebody - Rick Astley>]>

Album.objects.get(pk=1)
#<Album: Wall of Sound - Royksopp>

Album.objects.get(pk=2)
#<Album: Whenever You Need Somebody - Rick Astley>
```

- We find the albums by their primary keys (`pk`) that Django handles automatically

Create new directory path and file

New directory path: `mysite/music/templates/music/`

```
cd mysite/music/
mkdir templates/
cd templates/
mkdir music/
cd music/
```

File: `mysite/music/templates/music/index.html`

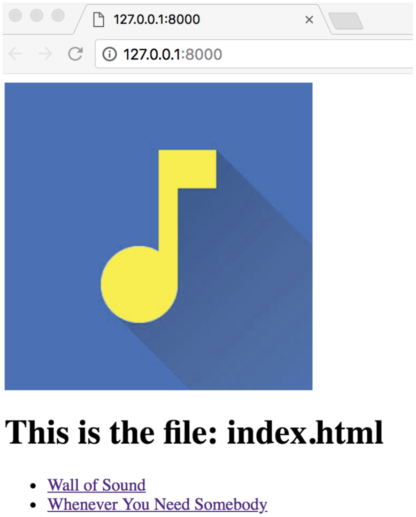
```
<img src = "http://www.cs.allegheny.edu/sites/obonhamcarter/cs380/graphics/music.jpg">
{% if all_albums %}
<h1> This is the file: index.html </h1>
 <ul>
      {% for album in all_albums %}
         <li><a href="/music/{{ album.id }}/">{{ album.album_title }}</a></li>
      {% endfor %}
  </ul>
{% else %}
  <h3> You do not have any albums to display. </h3>
{% endif %}
```

Create new file

File: `mysite/music/templates/music/detail.html`

```
<img src = "{{ album.album_logo }}">
<h1> This is the file: detail.html </h1>
<ul>
       <li>Artist: {{ album.artist }} </li>
       <li>Album Title:  {{ album.album_title }} </li>
       <li>Genre: {{ album.genre }} </li>
</ul>
```

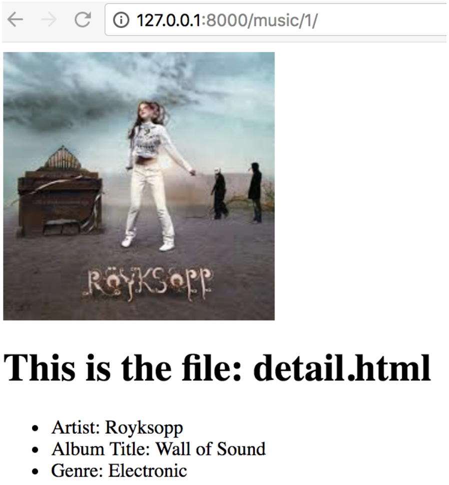
Main Web Page



This is the file: index.html

- Wall of Sound
- Whenever You Need Somebody

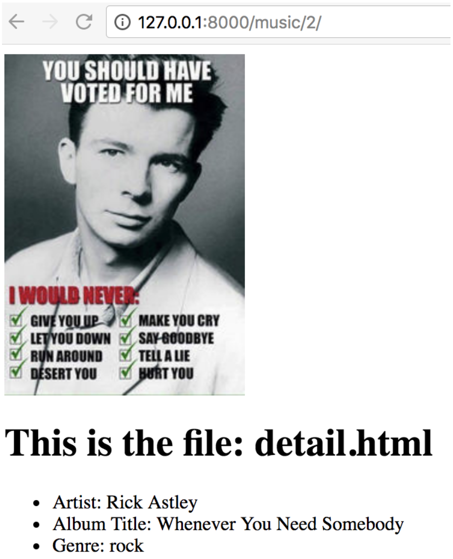
Royksopp Web Page

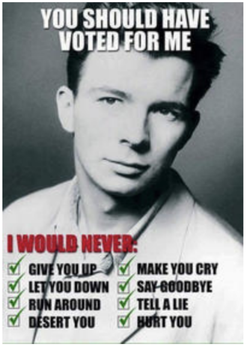

127.0.0.1:8000/music/1/

This is the file: detail.html

- Artist: Royksopp
- Album Title: Wall of Sound
- Genre: Electronic

Rick's Web Page

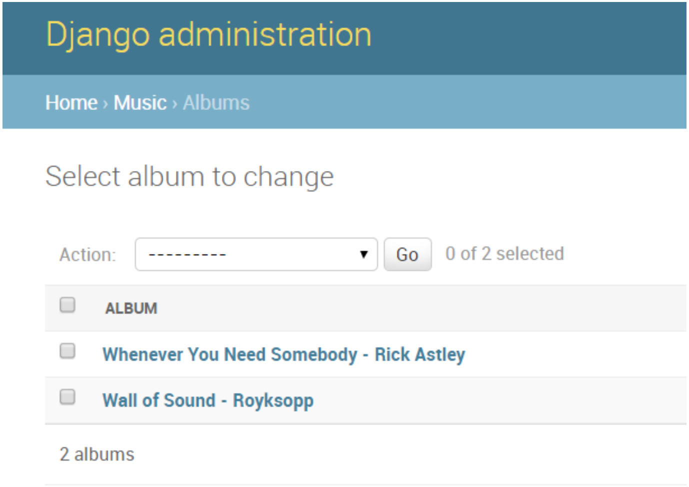

127.0.0.1:8000/music/2/



This is the file: detail.html

- Artist: Rick Astley
- Album Title: Whenever You Need Somebody
- Genre: rock

Admin


Django administration

Home › Music › Albums

Select album to change

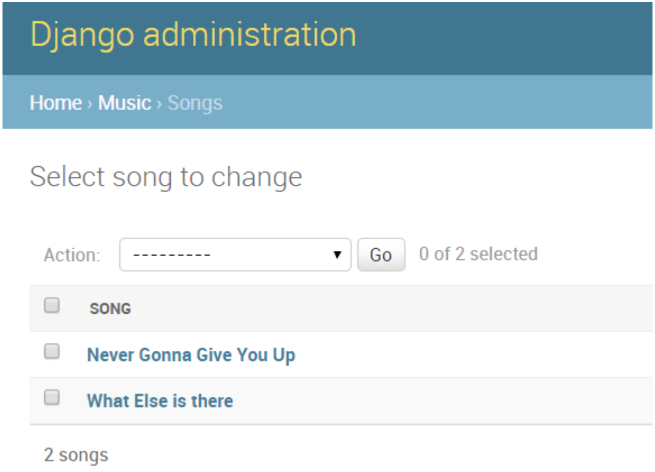
Action: --------- Go 0 of 2 selected

	ALBUM
☐	Whenever You Need Somebody - Rick Astley
☐	Wall of Sound - Royksopp

2 albums

- The `Albums` DB with two albums

Admin


Django administration

Home › Music › Songs

Select song to change

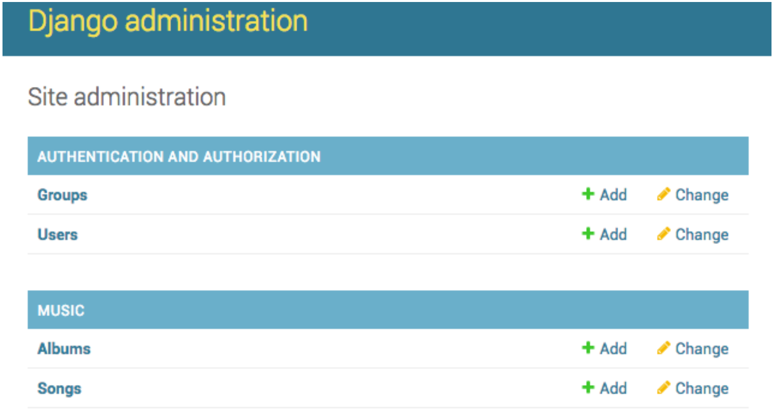
Action: --------- Go 0 of 2 selected

	SONG
☐	Never Gonna Give You Up
☐	What Else is there

2 songs

- The Songs DB with two songs

Admin


Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

MUSIC		
Albums	+ Add	Change
Songs	+ Add	Change

- You now have two DBs installed to try out