Test-Driven Development of Django Projects

By: Clayton S. Cook, P.E.

Oct. 12, 2021

Who Am I’m

- I’m Clayton Cook, currently a Geotechnical Engineer in the United States
- Looking to switch into a Django web development role
- Went through the Python Developer Mind (PDM) from this year with Bob Belderbos
- Worked through two Django MVP projects
  - PDM Auto Session MVP: Mixing and matching audio files to make an Irish jig to simulate and Irish Session
    - http://ec2-54-89-158-149.compute-1.amazonaws.com:8000/
  - PDM Geotech Database: Database to display hole locations and encountered soil layers for geotechnical drilling

What? and Why?

- At the end of my Django projects I had two major interests that I wanted to learn more about
  - Test-Driven Development for Django projects
    - Testing your Python Code by Martin Héroux Presentation on PDM
  - Deployment to AWS Elastic Beanstalk (with automated deployments using Github actions) using Docker Containers

Overview

- Highlight Test-Driven Development Concepts with an example project
- A Django project that houses mp3 files and shows a view with the title of the audio file

What is Django

- "A high-level Python web framework that encourages rapid development and clean, pragmatic design"

  - Djangoproject.com

  Key Features
  - Object-relational mapper
  - Automatic admin interface
  - Robust template system

- Is a web framework with batteries included as opposed to Flask that is more barebones

Starting a New Django Project

```
# Make a vitural environment
python -m venv venv

# activate virutal environment
venv\scripts\activate

# Install Django
pip install django

# Start Project
django-admin startproject core

# Rename top folder from core to DjangoApp
```

```
# Start a new app
python manage.py startapp audioapp

# Add to apps in core/settings.py
INSTALLED_APPS = [
      ...,
•     'audioapp’,
]

# Make Migrations and Migrate
python manage.py makemigrations
python manage.py migrate
```

Test-Driven Development with Django

What is Test-Driven Development

Practice of

1. Writing a deliberately failing test
2. Writing application code to make the test pass
3. Refactoring to optimize the code while the test continues to pass
4. Repeating the process until your project is complete

From Test-Driven Development with Django (2015), Kevin Harvey

Framework for Writing a Test, AAA

- **Arrange** – Set up your data and any necessary preconditions for your test
- **Act** – run the application code that you want to test
- **Assert** – check that your action is what you expected

Testing Setup and Tools

- Add a “test_settings.py” file

```
#  DjangoApp/core/test_settings.py
from .settings import *

DATABASES = {
   "default": {
      "ENGINE": "django.db.backends.sqlite3",
      "NAME": ":memory:",
   }
}

EMAIL_BACKEND = 'django.core.mail.backends.locmen.EmailBackend'
```

Testing Setup and Tools

- Add a “pytest.ini” file

```
#  DjangoApp/pytest.ini
[pytest]
DJANGO_SETTINGS_MODULE = core.test_settings
django_debug_mode = true
addopts = --nomigrations --cov=. --cov-report=html
```

Testing Setup and Tools

- Installing Tools

```
$ pip install pytest
$ pip install pytest-django
$ pip install pytest-cov
$ pip install factory-boy
```

- Try It

```
$ pytest
```

Testing Setup and Tools

- Create ".coveragerc" and see coverage

```
#  DjangoApp/.coveragerc
[run]
omit =
    *apps.py,
    *migrations/*,
    *settings*,
    *tests/*,
    *urls.py,
    *wsgi.py,
    *asgi.py,
    manage.py
```

```
# Looking at testing coverage
$ pytest
$ htmlcov/index.html
```

Pytest Testing Files

- Ready to write first test
- pytest will find all files called “test_*.py”
- It will execute all functions called “test_*()” on all classes that start with “Test*”

Pytest Testing Files

- Remove standard testing file created and create testing folder
  - Remove DjangoApp/audioapp/test.py
  - Add DjangoApp/audioapp/tests folder and __init__.py

- First Test File will be testing our models
  - Add DjangoApp/audioapp/tests/test_models.py file

Testing Models

- Model Test

```
# DjangoApp/audioapp/tests/test_model.py
import pytest

from audioapp.tests import factories

@pytest.mark.django_db
class TestAudio:
    def test_init(self):
        obj = factories.AudioFactory()
        assert obj.pk == 1, "Should save an instance"
```

Factory-boy factories

Testing class Audio in model.py

Testing function init in class Audio

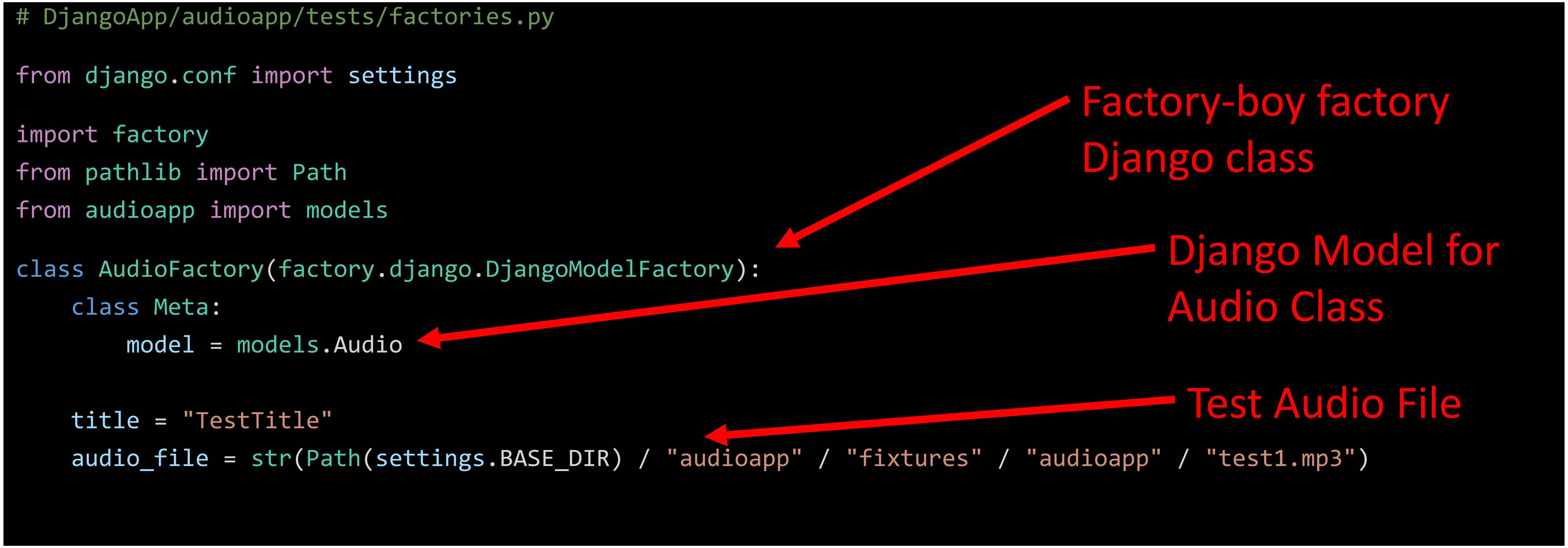
Testing Models

- Run pytest
- Import Error for factories

```
$ pytest
...
E   ImportError: cannot import name 'factories' from 'audioapp.tests'
(unknown location)


----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov


======================== short test summary info ==========================
ERROR audioapp/tests/test_models.py
!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!
============================ 1 error in 0.63s ============================
```

Testing Models

- Factory Boy Factory



```python
# DjangoApp/audioapp/tests/factories.py

from django.conf import settings

import factory
from pathlib import Path
from audioapp import models

class AudioFactory(factory.django.DjangoModelFactory):
    class Meta:
        model = models.Audio

    title = "TestTitle"
    audio_file = str(Path(settings.BASE_DIR) / "audioapp" / "fixtures" / "audioapp" / "test1.mp3")
```

Factory-boy factory Django class
Django Model for Audio Class
Test Audio File

Testing Models

- Run pytest
- AttributeError, no Audio found

```
$ pytest
...
E   AttributeError: module 'audioapp.models' has no attribute 'Audio'


----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov


======================== short test summary info =========================
ERROR audioapp/tests/test_models.py - AttributeError: module
'audioapp....!!!!!!!!!!!!!T!!! Interrupted: 1 error during collection
!!!!!!!!!!!!!!!!!
=========================== 1 error in 0.71s ============================
```

Testing Models

- Finally Make Django Model

```python
# DjangoApp/audioapp/models.py

from django.db import models

class Audio(models.Model):
    title = models.CharField('Audio File Title', max_length=50)
    audio_file = models.FileField(upload_to="audioapp/", verbose_
name="Audio MP3 file")
```

Testing Models

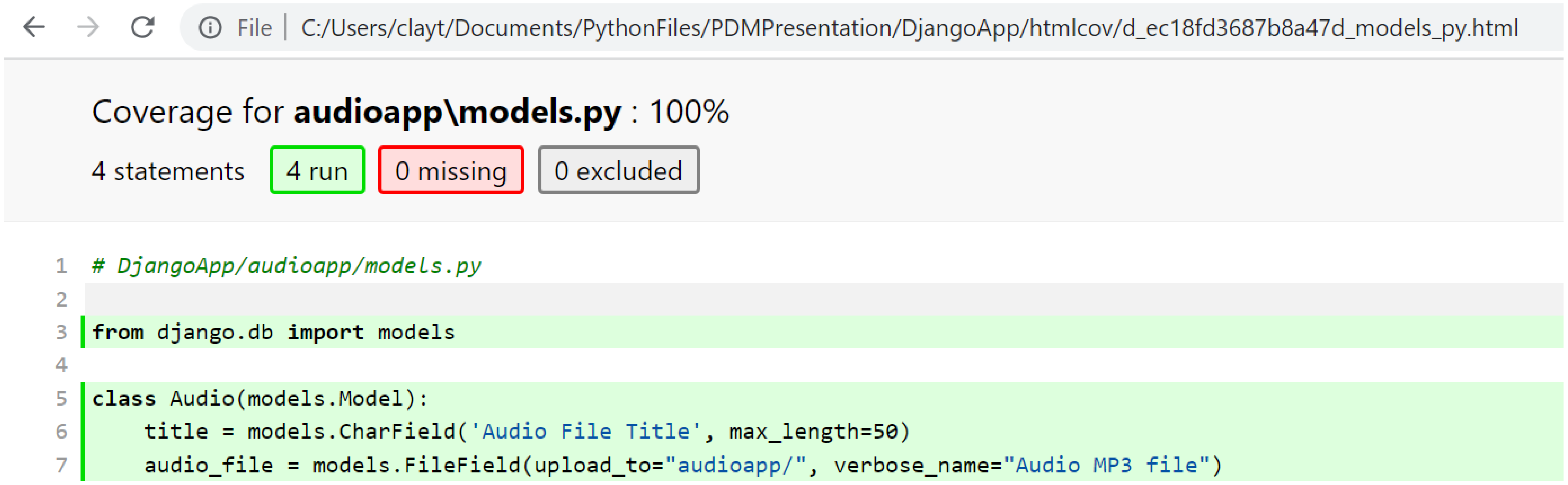
- Run Pytest, It Passed!

```
$ pytest
...
audioapp\tests\test_models.py .                                    [100%]

----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov


=========================== 1 passed in 1.00s ===========================
```

Testing Models

- Hows Coverage?

```
$ htmlcov\index.html
```



File | C:/Users/clayt/Documents/PythonFiles/PDMPresentation/DjangoApp/htmlcov/d_ec18fd3687b8a47d_models_py.html

Coverage for **audioapp\models.py** : 100%

4 statements | 4 run | 0 missing | 0 excluded

```python
# DjangoApp/audioapp/models.py

from django.db import models

class Audio(models.Model):
    title = models.CharField('Audio File Title', max_length=50)
    audio_file = models.FileField(upload_to="audioapp/", verbose_name="Audio MP3 file")
```

Testing Models

- Model Test Additional test_init assertion

```
# DjangoApp/audioapp/tests/test_model.py
class TestAudio:
    def test_init(self):
        obj = factories.AudioFactory()
        assert obj.pk == 1, "Should save an instance"

        # Check varing properties about default file
        # Size in bytes
        assert obj.audio_file.size == 1, "Should be able to check audio file size"
```

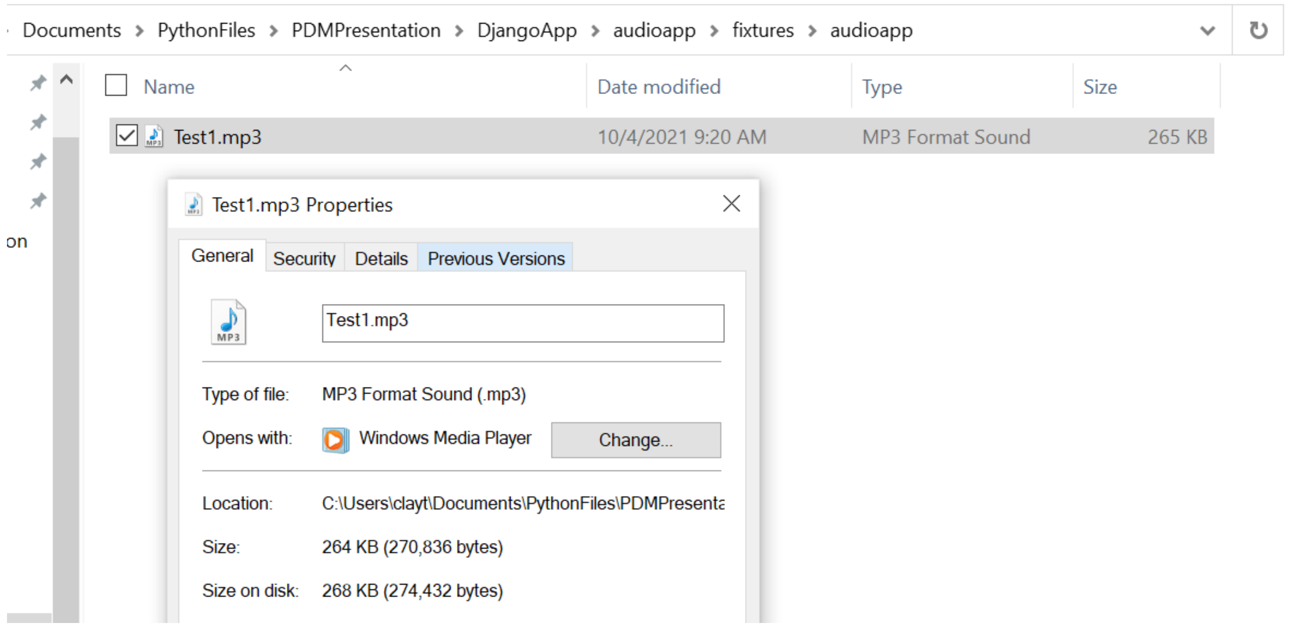
Testing Models

- Run pytest
- Failed, FileNotFoundError, So Need file in fixtures

```
$ pytest
...
E       FileNotFoundError: [WinError 2] The system cannot find the file specified:
'C:\\Users\\clayt\\Documents\\PythonFiles\\PDMPresentation\\DjangoApp\\audioapp\\fixtures\\audioap
p\\test1.mp3'


..\..\..\..\AppData\Local\Programs\Python\Python38\lib\genericpath.py:50: FileNotFoundError


----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov


=========================== short test summary info ============================
FAILED audioapp/tests/test_models.py::TestAudio::test_init -
FileNotFoundErro...============================== 1 failed in 0.95s ==============================
```

Testing Models

- Add file to fixtures and check file size to check during test
  - So 270836 bytes for Test1.mp3



Documents › PythonFiles › PDMPresentation › DjangoApp › audioapp › fixtures › audioapp

Name	Date modified	Type	Size
Test1.mp3	10/4/2021 9:20 AM	MP3 Format Sound	265 KB

Test1.mp3 Properties

Test1.mp3

Type of file:	MP3 Format Sound (.mp3)
Opens with:	Windows Media Player
Location:	C:\Users\clayt\Documents\PythonFiles\PDMPresenta
Size:	264 KB (270,836 bytes)
Size on disk:	268 KB (274,432 bytes)

```python
# DjangoApp/audioapp/tests/test_model.py
class TestAudio:
    def test_init(self):
        obj = factories.AudioFactory()
        assert obj.pk == 1, "Should save an instance"

        # Check varing properties about default file
        # Size in bytes
        assert obj.audio_file.size == 270836, "Should be able to check audio file size"
```

```
$ pytest
...
audioapp\tests\test_models.py .                                   [100%]

----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov


========================== 1 passed in 0.77s ===========================
```

Testing Views

- View Test

```python
# DjangoApp/audioapp/tests/test_views.py
import pytest

from django.test import Client
from django.urls import reverse

from audioapp.tests import factories

@pytest.mark.django_db
class TestDetailViewAudio:
    def test_shows_title(self):
        client = Client()

        # Create Audio and Get pk
        obj = factories.AudioFactory()
        pk  = obj.pk
        # Get detail view of audio created above
        response = client.get(reverse("audioapp:detail", args=(pk,)))
        assert response.status_code == 200, "View returns 200 status code"
        assert obj.title in response.content.decode(), "The audioapp.detail view shows the audio.title"
```

Factory-boy factories

Testing url path

Testing status and content

Testing Views

- Run pytest
- Failed, ‘audio’ is not a registered namespace, so go make the view

```
$ pytest
...
E                       django.urls.exceptions.NoReverseMatch: 'audio' is not a registered namespace

..\venv\lib\site-packages\django\urls\base.py:82: NoReverseMatch

----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov

=============================== short test summary info ================================
FAILED audioapp/tests/test_views.py::TestDetailViewAudio::test_shows_title -
django.ur...============================= 1 failed, 1 passed in 1.07s ==============================
```

Testing Views

- Update Core app Url path

```
# DjangoApp/core/urls.py

from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('audio/', include('audioapp.urls', namespace='audioapp')),
]
```

Testing Views

- Make urls.py in AudioApp files and add detail view

```
# DjangoApp/audioapp/urls.py

from django.urls import path
from audioapp import views

app_name = 'audioapp'
urlpatterns = [
    path('<int:pk>/', views.detail, name='detail'),
]
```

Testing Views

- In views.py in AudioApp files and add detail view

```
# DjangoApp/audioapp/views.py

from django.shortcuts import render
from django.shortcuts import get_object_or_404

from .models import Audio

# Create your views here.
def detail(request, pk):

    obj = get_object_or_404(Audio, pk=pk)

    context = {
        "obj": obj,
    }

    return render(request, ’audioapp/detail.html', context)
```

Testing Views

- Run pytest
- Failed, ‘audioapp/detail.html’ does not exist, So make html

```
$ pytest
...
django.template.exceptions.TemplateDoesNotExist: audio/detail.html

----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov

====================== short test summary info =======================
FAILED audioapp/tests/test_views.py::TestDetailViewAudio::test_shows_title
==================== 1 failed, 1 passed in 1.34s =====================
```

Testing Views

- Detail.html

```
# DjangoApp/audioapp/templates/audioapp/detail.html


<h1>{{ obj.title }}</h1>
```

Testing Views

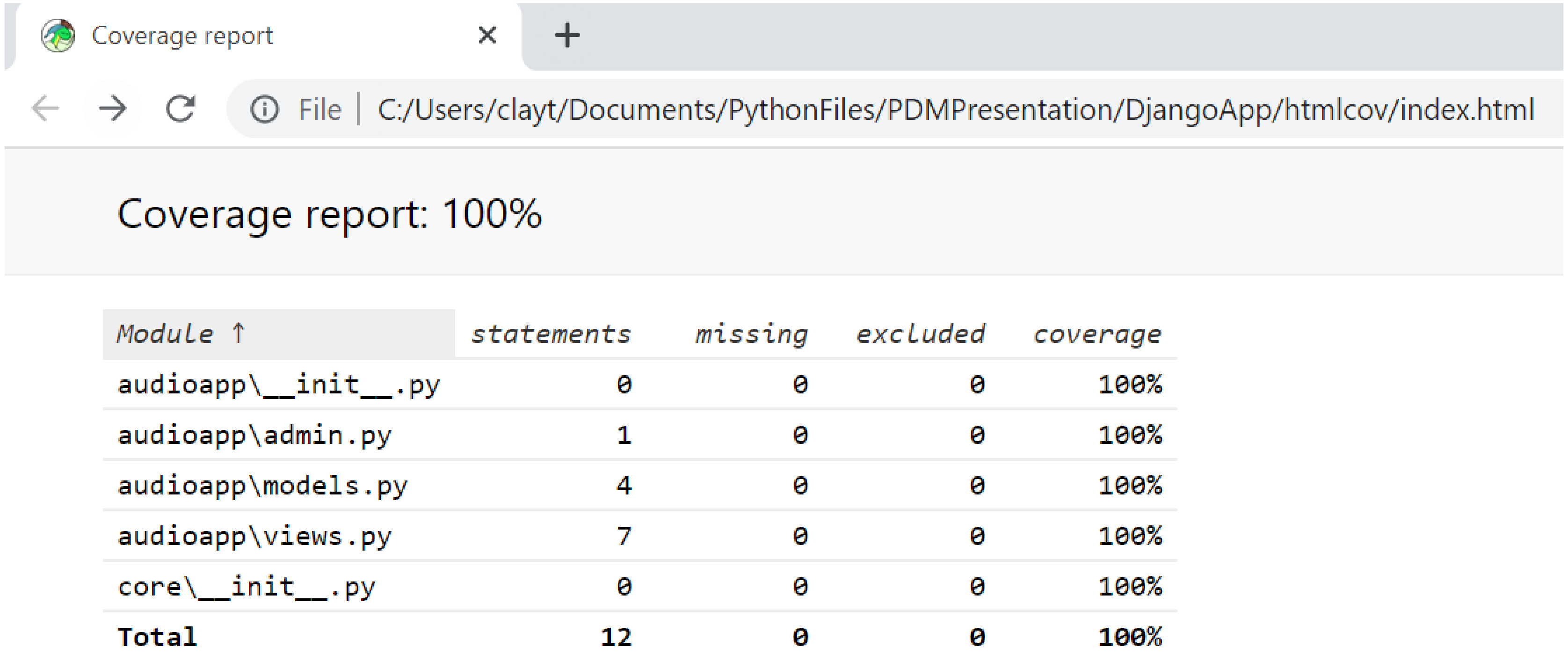
- Run pytest
- Passed, ‘detail.html’ shows title of audio object

```
$ pytest
...
audioapp\tests\test_models.py .                                            [ 50%]
audioapp\tests\test_views.py .                                             [100%]

----------- coverage: platform win32, python 3.8.6-final-0 -----------
Coverage HTML written to dir htmlcov


=============================== 2 passed in 0.85s ===============================
```

Testing Views

- Hows Coverage?

```
$ htmlcov\index.html
```



Coverage report

File | C:/Users/clayt/Documents/PythonFiles/PDMPresentation/DjangoApp/htmlcov/index.html

Coverage report: 100%

Module ↑	statements	missing	excluded	coverage
audioapp__init__.py	0	0	0	100%
audioapp\admin.py	1	0	0	100%
audioapp\models.py	4	0	0	100%
audioapp\views.py	7	0	0	100%
core__init__.py	0	0	0	100%
Total	**12**	**0**	**0**	**100%**

Testing Closing Comments

- Write the Test Then Test the Test
    - AAA (Arrange, Act, Assert)
- Tools Used
    - pytest, pytest-Django, factory-boy, pytest-cov
- Additional resources for Django Testing
    - https://www.mattlayman.com/django-riffs/13-automated-tests/
    - The Django Test Driven Development Cookbook - Singapore Djangonauts Talk
        - https://www.youtube.com/watch?v=41ek3VNx_6Q
    - Test-Driven Development with Django, Kevin Harvey
- Code at https://github.com/claytoncook12/PDMPresentation202108

Questions?