III B.Tech.

Computer Science & Engineering

CSE304: PYTHON PROGRAMMING WITH WEB FRAMEWORKS

UNIT-IV: Django

By

Mrs. S. KAMAKSHI, AP-III / CSE

School of Computing

Django Installation

- Create a virtual environment by executing the following from command prompt
  - py -m venv project-name
- Activate the virtual environment by executing
  - project-name\Scripts\activate.bat
- Install Django by executing
  - py -m pip install Django
- Verify installation
  - django-admin --version
- To verify from command prompt
  - python -m django --version

Starting New Project

- Execute
  - django-admin startproject mysite
  - It creates
    - mysite/
      - manage.py
      - mysite/
        - __init__.py
        - settings.py
        - urls.py
        - asgi.py
        - wsgi.py

Details of files created

- The outer mysite/ root directory is a container for your project.
- manage.py: A command-line utility that lets you interact with this Django project in various ways.
- The inner mysite/ directory is the actual Python package for your project.
- mysite/__init__.py: An empty file that tells Python that this directory should be considered a Python package.
- mysite/settings.py: Settings/configuration for this Django project.
- mysite/urls.py: The URL declarations for this Django project; a “table of contents” of your Django-powered site.
- mysite/asgi.py: An entry-point for ASGI-compatible web servers to serve your project.
- mysite/wsgi.py: An entry-point for WSGI-compatible web servers to serve your project.

Starting server

- To start the Django Development Server execute [from the outer project directory]
  - python manage.py runserver # default port 8000
  - python manage.py runserver 8080
- To create an app
  - python manage.py startapp poll
- Directory structure of poll:
  - poll/
    - __init__.py
    - admin.py
    - apps.py
    - migrations/
      - __init__.py
      - models.py
      - tests.py
      - views.py

views.py

- Type the following code in views.py

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello,
                world. You're at the
                 poll index.")
```

urls.py

- Create urls.py in poll directory and type the following code:

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index,
                name='index'),
]
```

Update mysite/urls.py

- In mysite/urls.py include which are given in red to point the root URLconf at the poll.urls module

```
from django.contrib import admin
from django.urls import include, path
urlpatterns = [ path('poll/', include('poll.urls')),
                path('admin/', admin.site.urls), ]
```

- Start server again
  - python manage.py runserver
- Go to http://localhost:8000/poll/ in browser
  - *"Hello, world. You're at the polls index."*

Defining Models (Table)

In Poll/models.py add the following

```
from django.db import models
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Database setup

- To include the app in our project
  - Add a reference to its configuration class in the INSTALLED_APPS setting.
  - The PollConfig class in the polls/apps.py file, so its dotted path is 'polls.apps.PollConfig'.
  - Edit the mysite/settings.py file and add that dotted path (which is in Red) to the INSTALLED_APPS setting

```
INSTALLED_APPS = [
    'poll.apps.PollConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Migrating the changes

- To create the database schema (To include the poll app in Django)
  - python manage.py makemigrations poll
- To display the sql commands executed
  - python manage.py sqlmigrate polls 0001
- Again run
  - python manage.py migrate

Interact with database through API shell

- Update poll/models.py

```
import datetime as dt
from django.db import models
from django.utils import timezone as tz

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    def __str__(self):
        return self.question_text
    def was_published_recently(self):
        return self.pub_date >= tz.now() - dt.timedelta(days=1)

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
    def __str__(self):
        return self.choice_text
```

Getting into interactive shell

- Open another command window and go to interactive python shell
  - python manage.py shell
- In interactive prompt type

```
>>> from polls.models import Choice, Question
>>>Question.objects.all()
>>> from django.utils import timezone as tz
>>> q = Question(question_text="What's new?",
pub_date=tz.now())
>>> q.save()
>>> q.id
>>> q.question_text
>>> q.pub_date
>>> q.question_text = "What's up?"
>>> q.save()
>>> Question.objects.all()
```

Getting into interactive shell

- In interactive prompt type

```
>>> from django.utils import timezone as tz
>>> current_year = tz.now().year
>>> Question.objects.get(pub_date__year=current_year)
>>> Question.objects.get(id=1)
>>> Question.objects.get(pk=1)        #pk →primarykey
>>> q = Question.objects.get(pk=1)
>>> q.was_published_recently()
```

Creating Choices for Question

```
>>> q = Question.objects.get(pk=1)
>>> q.choice_set.create(choice_text='Not much', votes=0)
>>> q.choice_set.create(choice_text='The sky', votes=0)
>>> c = q.choice_set.create(choice_text='Hack again', votes=0)
>>> c.question
>>> q.choice_set.all()
>>> q.choice_set.count()
>>> Choice.objects.filter(question__pub_date__year=current_year)
>>> c = q.choice_set.filter(choice_text__startswith='Hack')
>>> c.delete()
```