Django 1: Route, Model & Admin Site

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session’s Content

- Concurrency
- Parallelism

Today’s Content

- Django
  - Overview
  - Pipenv
  - Installation
  - Creating a project
  - Development server
  - Creating an app
- MVC vs. MVT
- Route
- Model
- Admin Site

Django

Overview

- High-level Python web framework
- Encourages rapid development
- Takes care of much of the hassle of web development
- Resource: https://www.djangoproject.com
- Alternative - Flask (micro-framework)
    - Resource: https://flask.palletsprojects.com/en/1.1.x

Pipenv

- Automatically creates & manages a virtual environment for your projects
- Adds & removes packages from your `Pipfile` as you install & uninstall packages
- Generates a `Pipfile.lock` which is used to produce deterministic builds
- It should look very similar to the `package.json` & `package-lock.json` you saw in the Fundamentals of Web Development course
- Resource: https://pipenv.pypa.io/en/latest

Pipenv

- From the command line, cd into a directory where you would like to create your `Pipfile`, then run the following command:

```
# Windows
...\> pipenv shell

# Linux or macOS
$ pipenv shell
```

Django Installation

- Requires Python
- Python includes a lightweight database
  - For development, we will use SQLite
  - For production, we will use MariaDB or PostgreSQL
- Resource: https://docs.djangoproject.com/en/3.0/topics/install/#install-the-django-code

```
# Windows
...\> pipenv install Django

# Linux or macOS
$ pipenv install Django
```

Installation

- View `Pipfile` & `Pipfile.lock`

```
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]

[packages]
django = "*"

[requires]
python_version = "3.7"
```

Creating a Project

- Auto-generate code that creates a Django project
- Each project contains settings for an instance of Django including:
  - Database configurations
  - Django-specific options
  - App-specific options
- Avoid naming projects after in-built Python or Django components
- Resources:
  - https://docs.djangoproject.com/en/3.0/intro/tutorial01/#creating-a-project
  - https://docs.djangoproject.com/en/3.0/faq/troubleshooting/#troubleshooting-django-admin

Creating a Project

- In the same directory as your `Pipfile` & `Pipfile.lock`, run the following command:

```
# Windows
...\> django-admin startproject mvt

# Linux or macOS
$ django-admin startproject mvt
```

Creating a Project

- Project structure:
  - `mvt/` - Container for the Django project (outer directory)
  - `manage.py` - Command-line utility for interacting with the project
  - `mvt/` - Python package for the project (inner directory)
  - `mvt/__init__.py` - Empty file which tells Python that this directory should be considered a package*
  - `mvt/asgi.py` - Entry-point for ASGI-compatible web servers to serve the project
  - `mvt/settings.py` - Settings/configurations for the project
  - `mvt/urls.py` - URL declarations for the project
  - `mvt/wsgi.py` - Entry-point for WSGI-compatible web servers to serve the project

- Resources:
  - https://docs.djangoproject.com/en/3.0/ref/django-admin
  - https://docs.python.org/3/tutorial/modules.html#tut-packages
  - https://docs.djangoproject.com/en/3.0/topics/settings
  - https://docs.djangoproject.com/en/3.0/topics/http/urls

Development Server

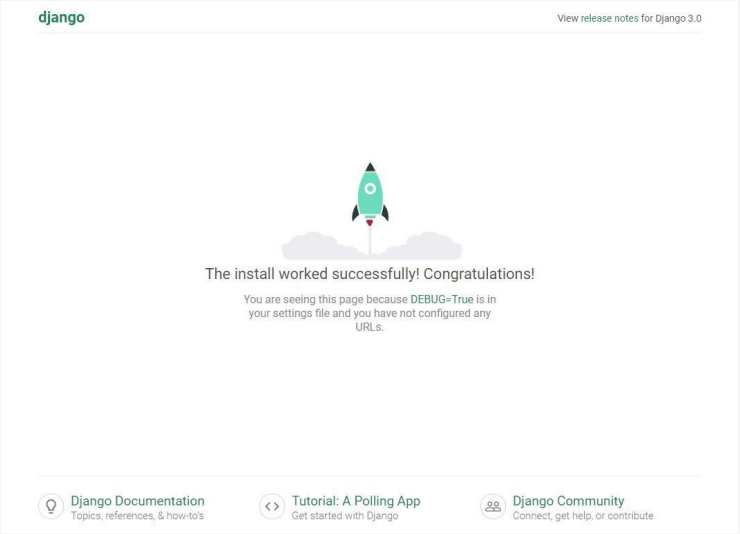
- Lightweight web server written purely in Python
- From the command line, cd into the outer `mvt/` directory, then run the following command:

```
# Windows
...\> python manage.py runserver

# Linux or macOS
$ python manage.py runserver
```

Development Server

- Navigate to http://127.0.0.1:8000



Creating an App

- Command-line utility which automatically generates the basic structure of an app

```
# Windows
...\> python manage.py startapp polls

# Linux or macOS
$ python manage.py startapp polls
```

Creating an App

- App structure:

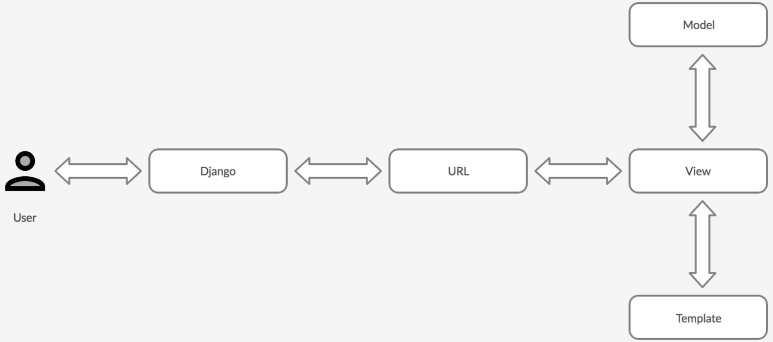
```
mvt/

polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

MVC vs. MVT

MVC vs. MVT

- What is the difference between MVC & MVT?
- Popular web frameworks which use MVC:
  - ASP.NET Core MVC
  - Laravel
  - Ruby on Rails
  - Spring MVC



Route

Route

- To create an URLconf in the `polls/` directory, create a file called `urls.py`
- App structure:

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    urls.py
    views.py
```

Route

- Resources:
  - https://docs.djangoproject.com/en/3.0/ref/urls/#path
  - https://docs.djangoproject.com/en/3.0/ref/urls/#django.urls.include

- `polls/urls.py`

```
from django.urls import path
from . import views

# Placeholder route. We can not proceed with no routes
urlpatterns = [
    path('', views.index, name='index'),
]
```

Route

- mvt/urls.py

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

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

View

View

- polls/views.py

```
from django.http import HttpResponse

# Again...another placeholder
def index(request):
    return HttpResponse('hello')
```

Model

Model

- Single, definitive source of truth about your data
- Contains the essentials fields & behaviours of your data
- Follows the DRY principle
- Define your data model in one place & automatically derive from it

Model

- Resources:
  - https://docs.djangoproject.com/en/3.0/ref/models/instances/#django.db.models.Model
  - https://docs.djangoproject.com/en/3.0/ref/models/fields/#module-django.db.models.fields

- `polls/models.py`

```
from django.db import models
from django.utils import timezone

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return self.question_text

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
```

Model

- Custom method
- Import `timedelta` method from `datetime`

```
from django.db import models
from django.utils import timezone
from datetime import timedelta

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField(default=timezone.now)

    def was_published_recently(self):
        return self.pub_date >= timezone.now() - timedelta(days=1)

    def __str__(self):
        return self.question_text

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
```

Model

- mvt/settings.py

```
INSTALLED_APPS = [
    'polls.apps.PollsConfig'
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'Django.contrib.staticfiles',
]
```

Model

- Changes made to `polls/models.py`
- Store changes as a migration
  - How Django stores changes to your model

```
# Windows
...\> python manage.py makemigrations polls

# Linux or macOS
$ python manage.py makemigrations polls

Migrations for 'polls':
  polls/migrations/0001_initial.py
    - Create model Question
    - Create model Choice
```

Model

- Run all migrations that have not be applied against your database

```
# Windows
...\> python manage.py migrate

# Linux or macOS
$ python manage.py migrate

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
```

Admin Site

Admin Site

- Create a user who can login to the Django admin site
- Resource: https://docs.djangoproject.com/en/3.0/ref/contrib/admin

```
# Windows
...\> python manage.py createsuperuser

# Linux or macOS
$ python manage.py createsuperuser

Username: admin
Email address: admin@example.com
Password: ********** # P@ssw0rd123
Password: ********** # P@ssw0rd123
Superuser created successfully.
```

Admin Site

- `polls/admins.py`

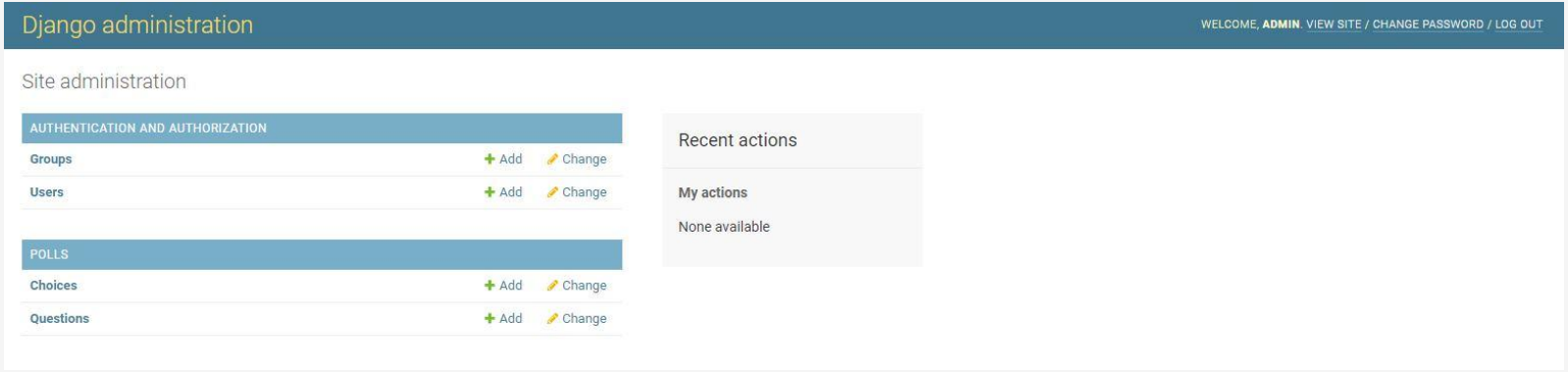
```
from django.contrib import admin
from .models import Question, Choice

admin.site.register(Question)
admin.site.register(Choice)
```

Admin Site

- Start development web server - `python manage.py createsuperuser`
  - Username
  - Email
  - Password
- Navigate to http://127.0.0.1:8000/admin

Admin Site


Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
POLLS
Choices + Add Change
Questions + Add Change
Recent actions
My actions
None available


Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Polls › Questions › Add question
Add question
Question text:
Pub date:
Date: 2020-07-14 Today
Time: 09:43:17 Now
Note: You are 12 hours ahead of server time.
Save and add another
Save and continue editing
SAVE