Introduzione

Valeria Leonardi

@vleonardi

The (Python) Web framework for perfectionists with deadlines

python™

- Interpretato
- Interattivo
- Multiparadigma
  - OO, programmazione strutturata..
- Indentazione per la definizione di blocchi
- Dynamic typing
- Modulare

Web-Poll Application

3....2....1....

- Install Python (2.3 <= versione <= 2.7)
  - pip install ipython
- Install Django
  - pip install django
- Install sqlite
  - pip install sqlite3

Test With:

- import Ipython
- import django
- import sqlite3

… Let’s start !

- **Create a new Project**
  - Enter the Command
    - django-admin startproject mysite
  - Resulting file structure is:
    - mysite/
      - manage.py
      - mysite/

        __init__.py

        manage.py

        settings.py

        urls.py

… Let’s start !

- Run the command:
  - **python manage.py migrate**
- Loads up django database tables into the database as follows:

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

… Let’s start !

- Run the command:
  - **python manage.py createsuperuser**
- Creates a super user in the django database tables as follows:

```
$ /opt/anaconda/bin/python manage.py createsuperuser
        Username (leave blank to use 'user'):       admin
        Email address:              me@email.com
        Password:                   abc=123!
        Password (again):           abc=123!
```

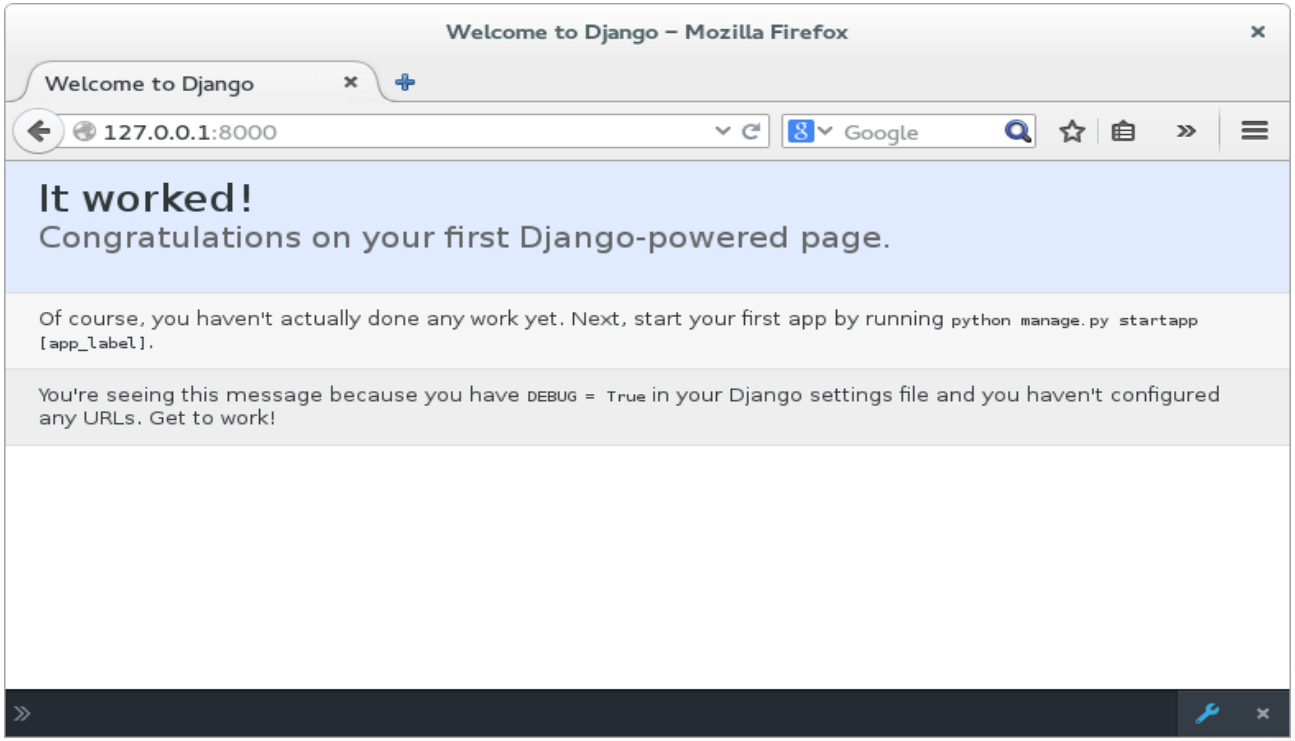
… Let’s start !

- Run the command:
  - **python manage.py dbshell**
- Starts the database application in the shell for sql queries

Command	Description
.tables	Show all tables
.headers on	Turn header row on/off
.mode column	Displays rows in columns
Select * from auth_user;	Display rows from superuser table
.quit	Exit sqlite

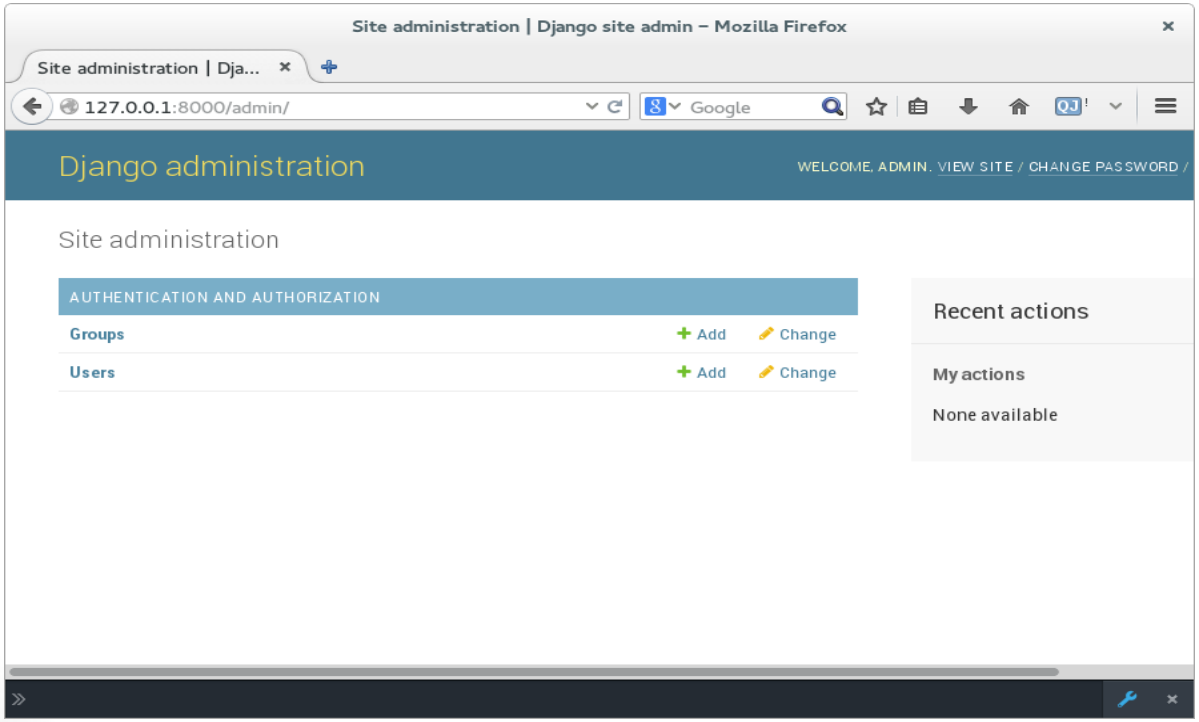
… Let’s start !

- Run the commands:
  - **python manage.py runserver**
- Open a browser window http://localhost:8000



… Let’s start !

- Open http://localhost:8000/admin
- Enter the superuser name and password to access the admin screen as follows:



… Let’s start !

Create a new App in the mysite/ project directory

Enter the command:

- python manage.py startapp polls

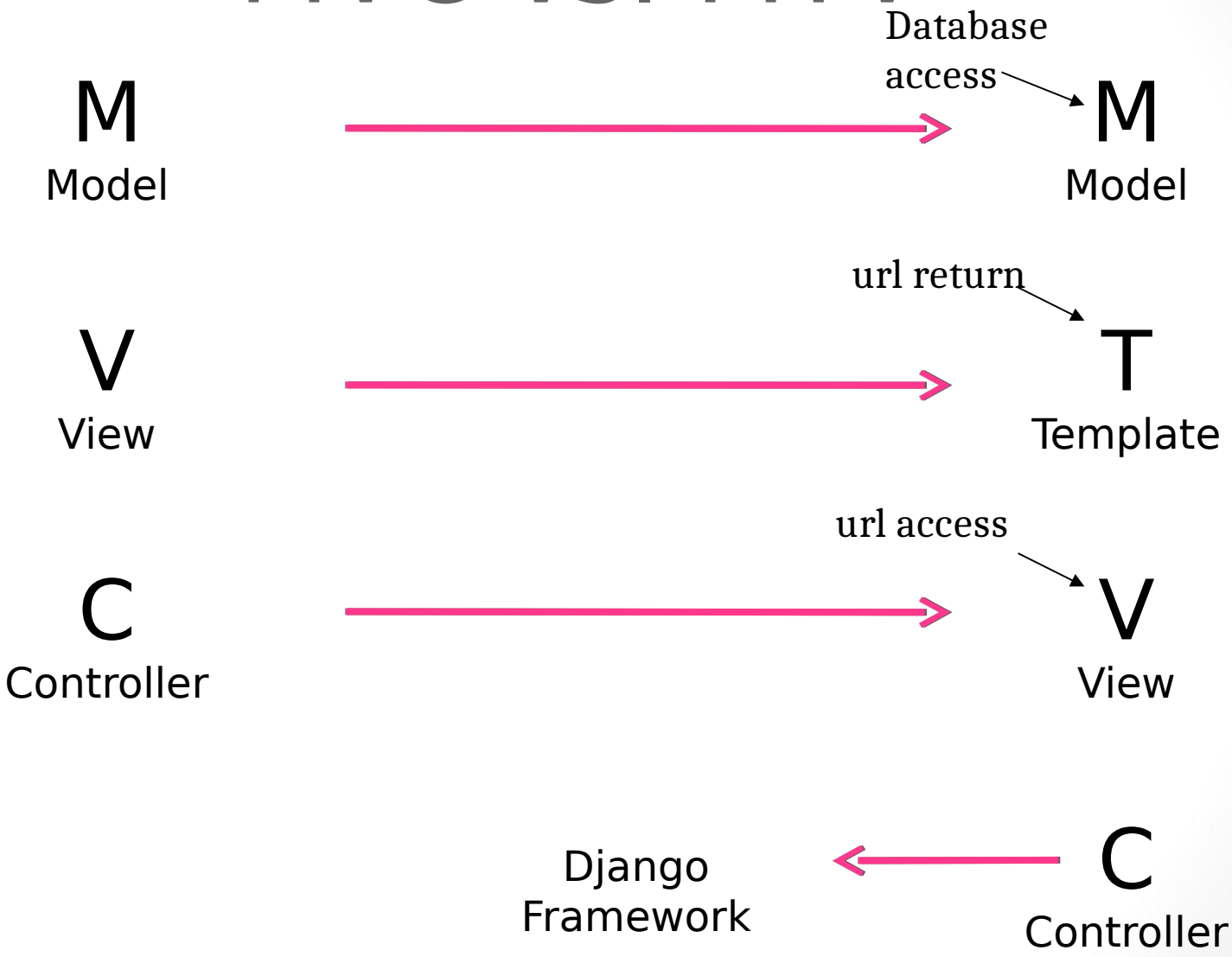
Resulting file structure is

- polls/

  __init__.py
  models.py
  tests.py
  views.py

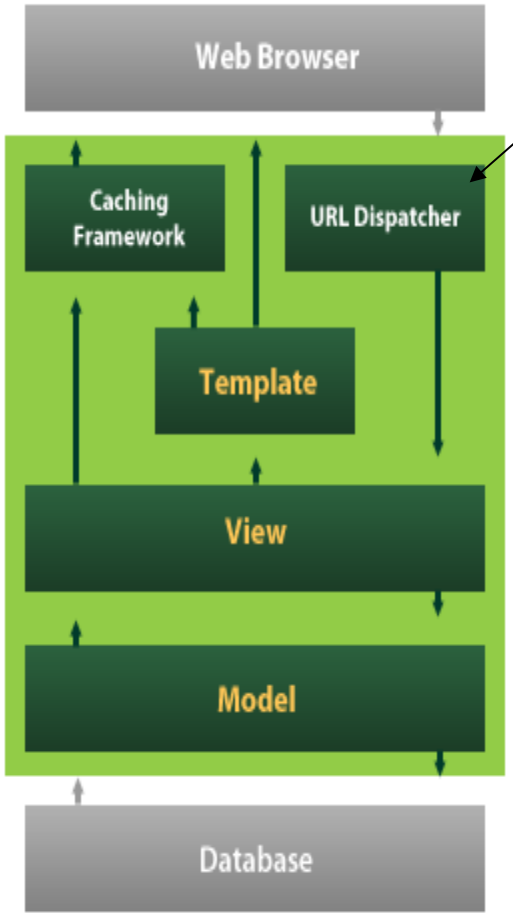
- Apps will use the same database file as their project (mysite)
- Projects need to import apps in their settings.py file.

MVC vs. MTV


Database access
M
Model
M
Model
url return
V
View
T
Template
url access
C
Controller
V
View
Django Framework
C
Controller

Django Architecture

Views Controls what users sees
Templates How user sees it
Controller URL dispatcher



url access is basically controller

1. The URL dispatcher (urls.py) maps the requested URL to a view function and calls it. If caching is enabled, the view function can check to see if a cached version of the page exists and bypass all further steps, returning the cached version, instead. Note that this page-level caching is only one available caching option in Django. You can cache more granularly, as well.

2. The view function (usually in views.py) performs the requested action, which typically involves reading or writing to the database. It may include other tasks, as well.

3. The model (usually in models.py) defines the data in Python and interacts with it. Although typically contained in a relational database (MySQL, PostgreSQL, SQLite, etc.), other data storage mechanisims are possible as well (XML, text files, LDAP, etc.).

4. After performing any requested tasks, the view returns an HTTP response object (usually after passing the data through a template) to the web browser. Optionally, the view can save a version of the HTTP response object in the caching system for a specified length of time.

5. Templates typically return HTML pages. The Django template language offers HTML authors a simple-to-learn syntax while providing all the power needed for presentation logic.

Model

- Model: Python
- python manage.py dbshell
- python manage.py syncdb

View

- View
  - Return HttpResponse object OR
  - Raise an exception
- Examples:
  - Index
  - Detail
  - Vote
  - Results

Template Url

- Templates:
  - Index.html
  - Details.html
  - Results.html
- “Masterpage”: base.html

- URL.py: regulate routing
  - (regular expression, Python callback function [, optional dictionary])

«Batteries included»

- Django.contrib:
  - Admin-site
  - Authentication
  - Formtools
  - Gis
  - …
- Admin-site
  - INSTALLED_APPS
  - URLS
  - admin.py
  - Python manage.py createsuperuser