MVC & MTV (= 디자인 패턴)

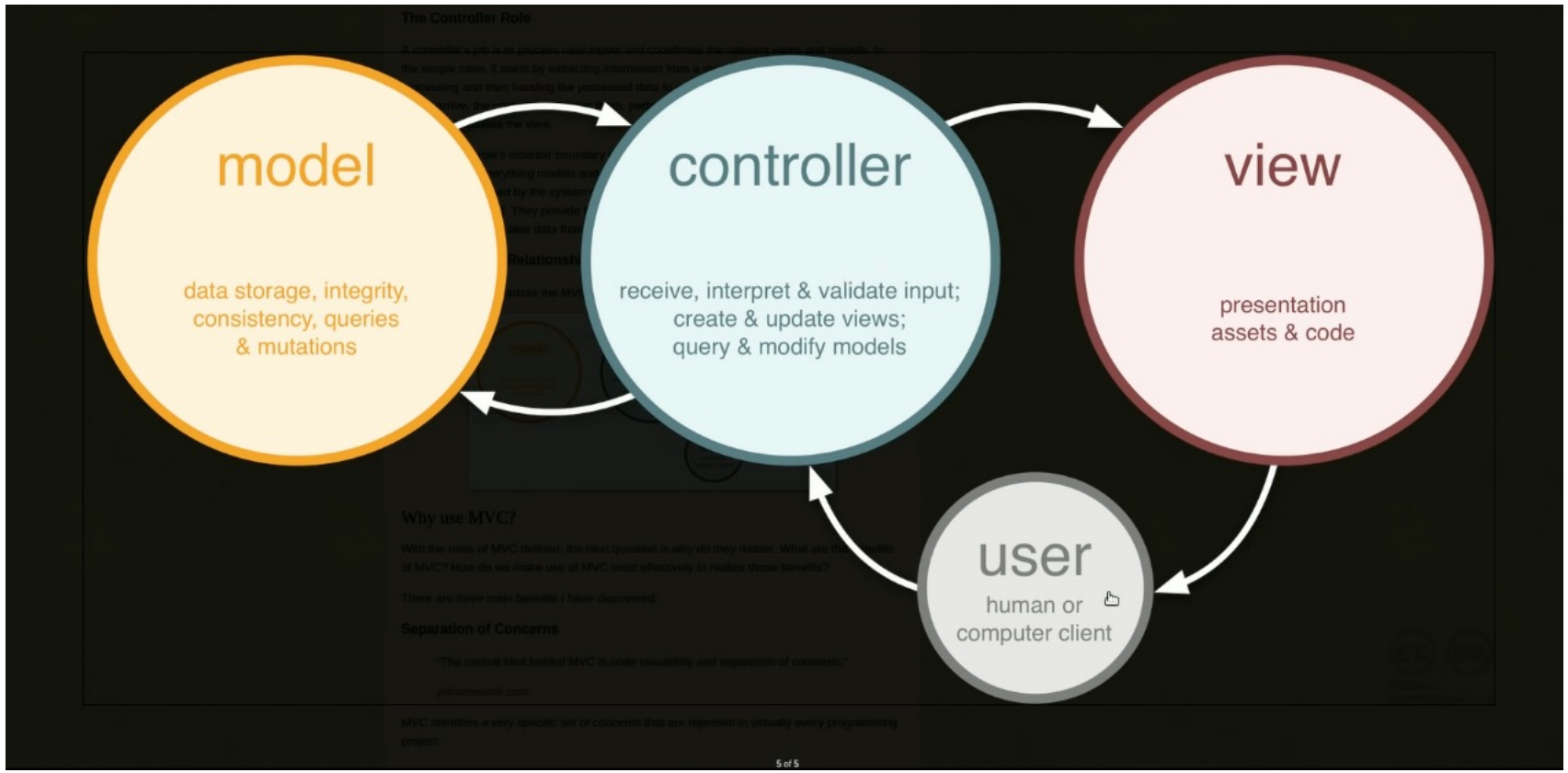
- **M**odel
  - 안전하게 데이터를 저장
- **V**iew
  - 데이터를 적절하게 유저에게 보여줌
- **C**ontrol, **T**emplate(Django)
  - 사용자의 입력과 이벤트에 반응하여 Model과 View를 업데이트

(= 유저 인터페이스)

첨부 : http://www.essenceandartifact.com/2012/12/the-essence-of-mvc.html

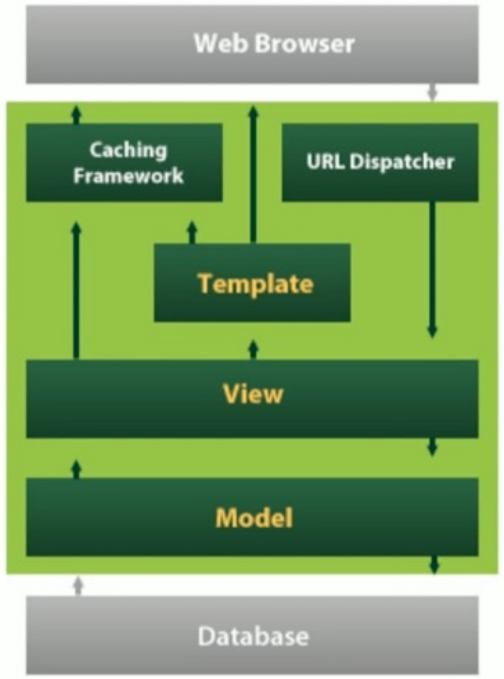
옛날에는 M, V, C 가
섞여 있었음

핵심은 분리되어 있다는 것


model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client

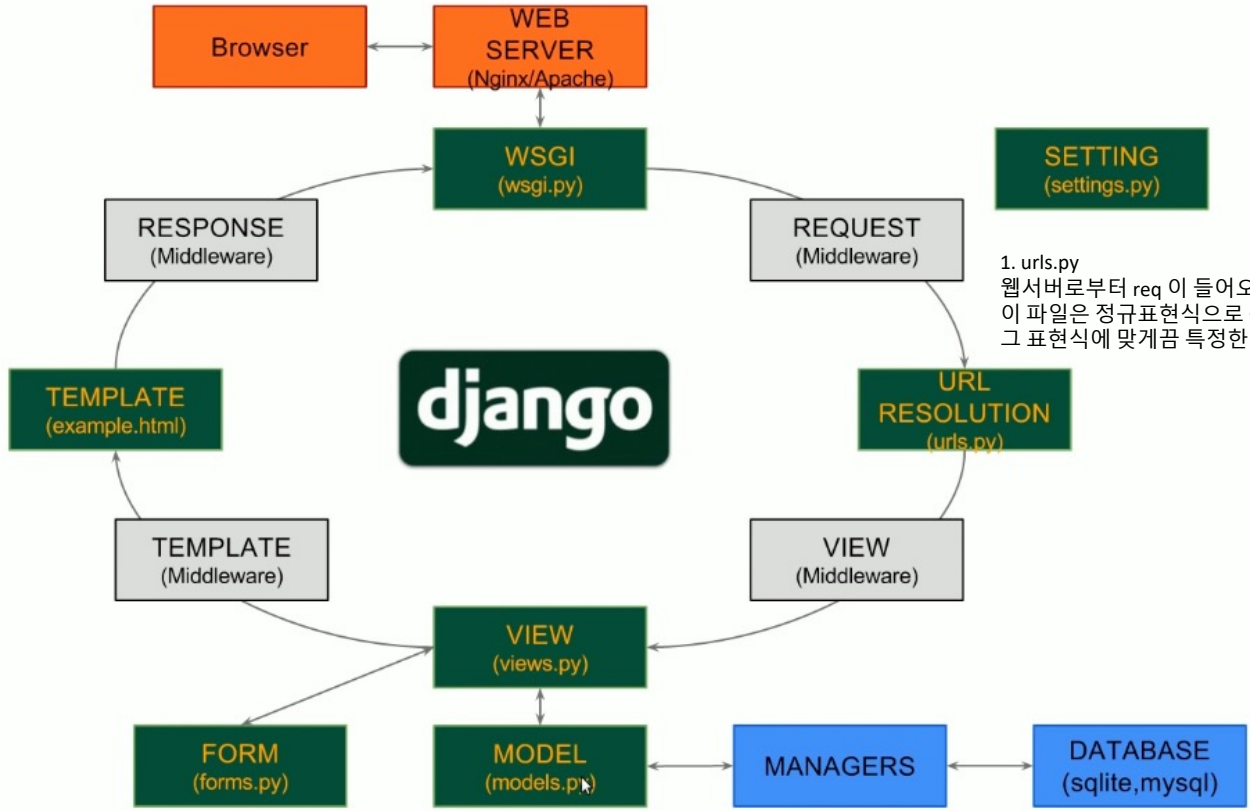
녹색으로 표시한 파일들이 실질적으로 개발자가 다루는 파일

Django 개념



출처: https://mytardis.readthedocs.org/en/latest/architecture

View 에서 데이터를 가공한 다음,
사용자에게 보여주는 ui 작업할 차례
장고는 Template 을 만들어 웹서버로 전송.
템플릿은 html 파일인데 그 안에 로직을 삽입할 수 있음
컨트롤과 관련된 다양한 로직.
Template 안에는 view 에서 받은 데이터들을 어떻게 html 에 보여줄지에 대한 코드들이 담김.



1. urls.py
웹서버로부터 req 이 들어오면 urls.py 가 받게 됨
이 파일은 정규표현식으로 구성되어 있는데,
그 표현식에 맞게끔 특정한 view (=view.py) 로 보냄

Form 을 이용해서 모델과 템플릿에 있는 사용자들이 쓰는 ui 가 아주 쉽게 관리됨.

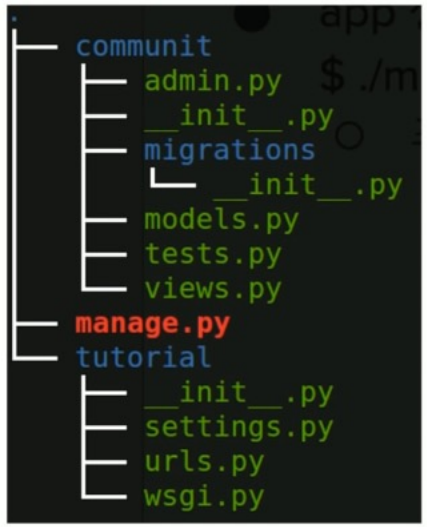
Model 에서 변수만 다루면 그 뒤에서 MANAGERS 가 알아서 sql 관련한 처리 다해줌.

Project와 App

- 프로젝트 생성

  $ django-admin startproject tutorial
- app 생성

  $ ./manage.py startapp community
  - 프로젝트 내부에 다수의 app 생성



Admin.py 는 관리자 권한을 가진 유저가 볼수 있는 페이지를 다루는 것

하나의 Project 가 하나의 웹사이트라고 생각하면 됨. 그 안에 여러 의미있는 기능들이 있을 텐데,
이는 앱으로 관리됨. 블로그가 하나의 어플리케이션, 계시판도 하나의 어플리케이션, 등등 분리해서 프로그래밍
앱은 다른 프로젝트에서 재사용 할 수 있음

settings.py

프로젝트 환경 설정 파일

- DEBUG
  - 디버그 모드 설정
- INSTALLED_APPS
  - pip로 설치한 앱 또는 본인이 만든 app을 추가
- MIDDELWARE_CLASSES
  - request와 response 사이의 주요 기능 레이어
- TEMPLATES
  - django template관련 설정, 실제 뷰(html, 변수)
- DATABASES
  - 데이터베이스 엔진의 연결 설정
- STATIC_URL
  - 정적 파일의 URL(css, javascript, image, etc.)

프로그래밍 할 때 에러같은 걸 보고 싶으면 Debug 모드를 true 로.
배포할 때는 false 해야 노출 안됨.

manage.py

- 프로젝트 관리 명령어 모음
- 주요 명령어
  - startapp - 앱 생성
  - runserver - 서버 실행
  - createsuperuser - 관리자 생성
  - makemigrations app - app의 모델 변경 사항 체크
  - migrate - 변경 사항을 DB에 반영
  - shell - 쉘을 통해 데이터를 확인
  - collectstatic - static파일을 한 곳에 모음
- ex)

  ./manage.py runserver 0.0.0.0:8080

```
[django]
    check
    compilemessages
    createcachetable
    dbshell
    diffsettings
    dumpdata
    flush
    inspectdb
    loaddata
    makemessages
    makemigrations
    migrate
    runfcgi
    shell
    showmigrations
    sql
    sqlall
    sqlclear
    sqlcustom
    sqldropindexes
    sqlflush
    sqlindexes
    sqlmigrate
    sqlsequencereset
    squashmigrations
    startapp
    startproject
    syncdb
    test
    testserver
    validate
```

```
[auth]
    changepassword
    createsuperuser

[sessions]
    clearsessions

[staticfiles]
    collectstatic
    findstatic
    runserver
```