Django로 웹 서비스 만들기
- 기초편 -

싸이지먼트
정광윤

본 자료는 싸이지먼트에서 운영하는 소비자ON 스터디에서 발표한 자료입니다.
Django를 간단하게 소개하고 있습니다.

Table of Contents

제가 쉽게 빠르게 개발할 수 있다는 Django를 공부하면서 느낀 것은,
Django는 빙산의 일각이라는 점입니다. 산 넘어 산입니다.

1. 동네 한 바퀴

Django Overview

Definition

Django (/ˈdʒæŋgoʊ/ *JANG-goh) is a free and open source web application framework, written in Python, which follows the model–view–controller (MVC) architectural pattern. (by Wikipedia)*

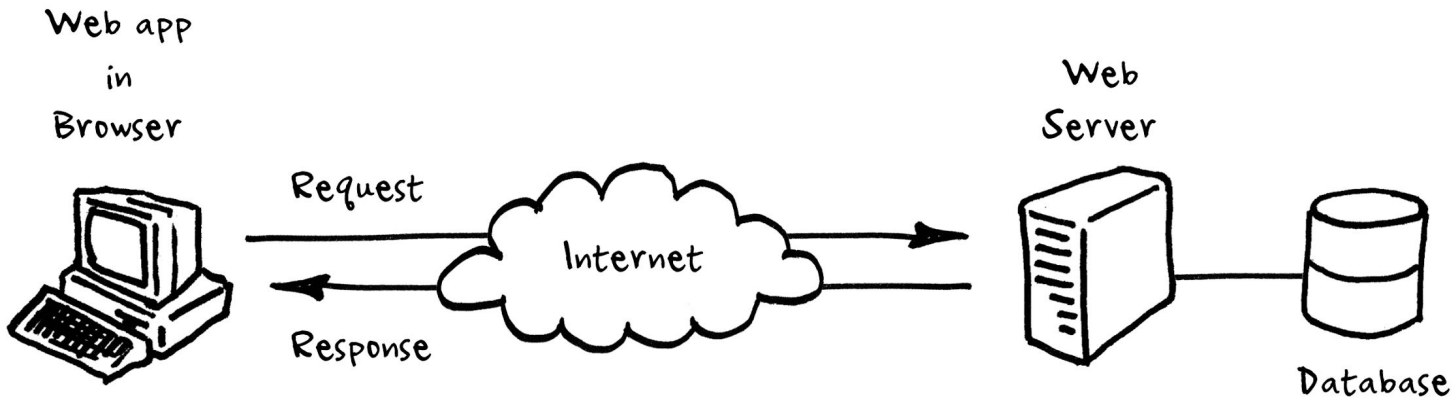
- *Open Source*
- *Web Application Framework*
- *Written in Python*
- *Follows Model-View-Controller Architectural Pattern*

www.djangoproject.com
Server-Side Programming
Django 1.8.6 (11/20/2015)
Python 선지식 필요

2. 산책로

Server - Client

Server - Client, 그렇고 그런 사이



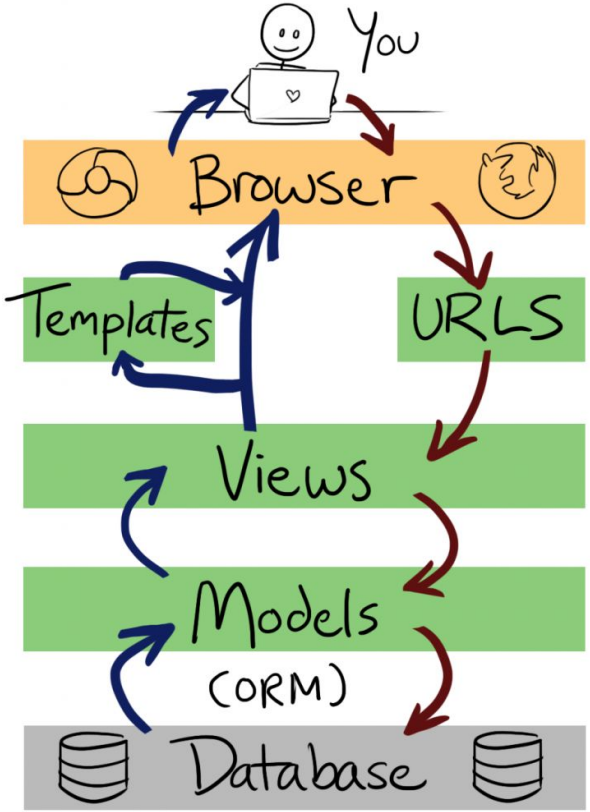
우리는 Request하고 Response하는 사이
(HTTP라는 규약에 따라)

http://i-connect-137.blogspot.kr/2013/07/post-4.html

3. 약수터

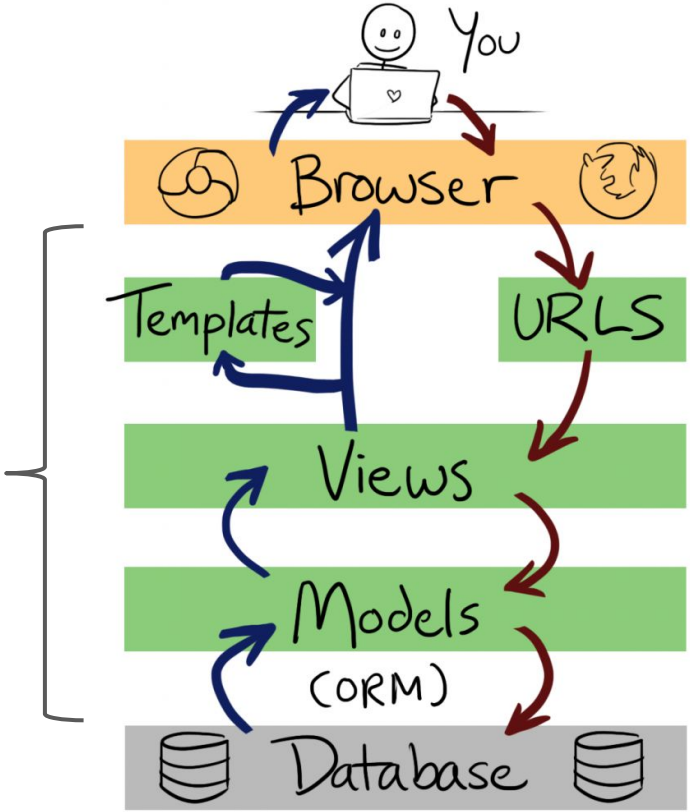
MVC 패턴

Django에서 그렇고 그런 사이



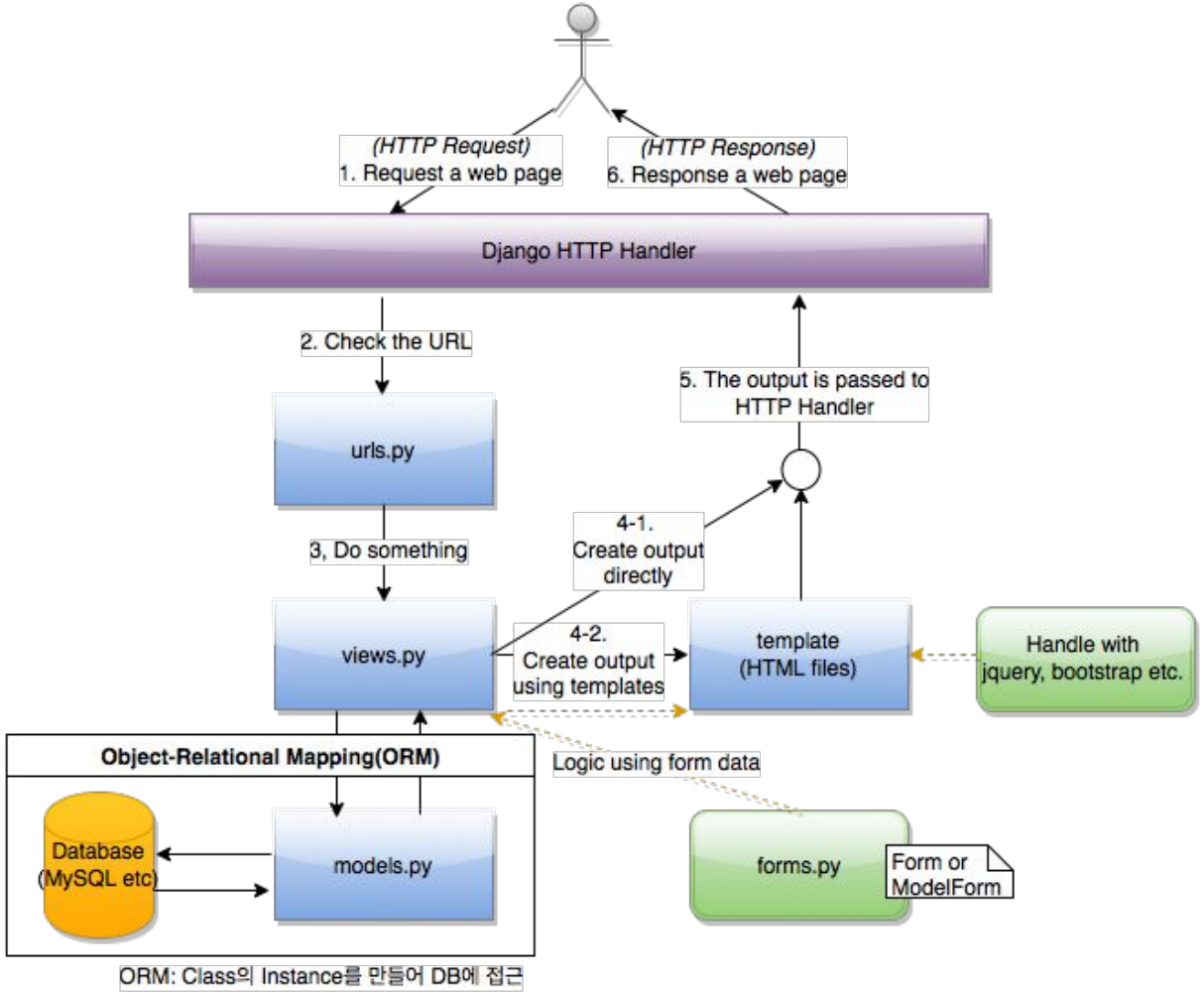
Request - Response Flow

Django가 Request하고 Response하는 흐름



MTV Pattern

MVC	MTV
Model	Model
View	Template
Controller	View



4. 뒷동산

Web Framework

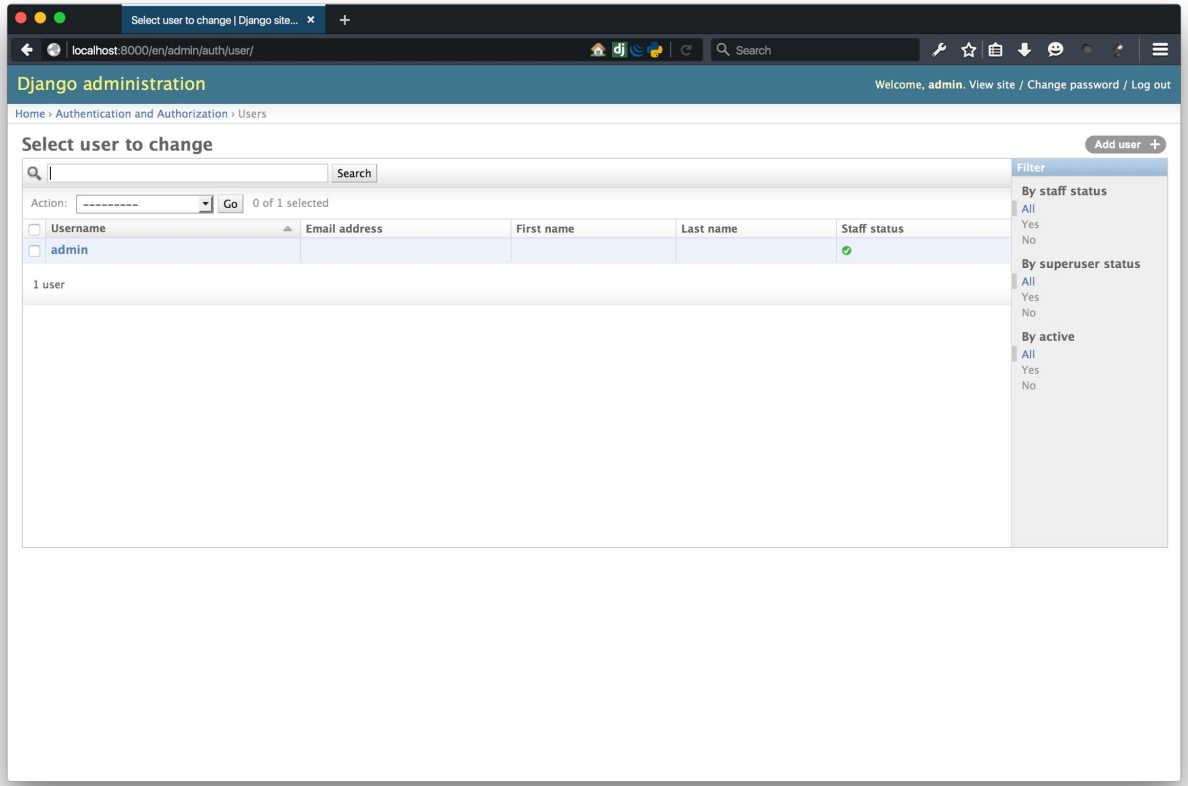
Web Framework

- The web framework for perfectionists with deadlines.
- 빠르게 개발하기 위해 미리 필요한 기능을 만들어 놓은 것
- 혼용 용어: Library, Package

Feature 1/4 - Admin Page

- 관리자 페이지
- 모든 데이터에 대해 CRUD 가능
  - Create
  - Read
  - Update
  - Delete
- 이미 Django에 만들어져 있는 하나의 시스템
- Django 패키지만 설치하면 바로 사용 가능

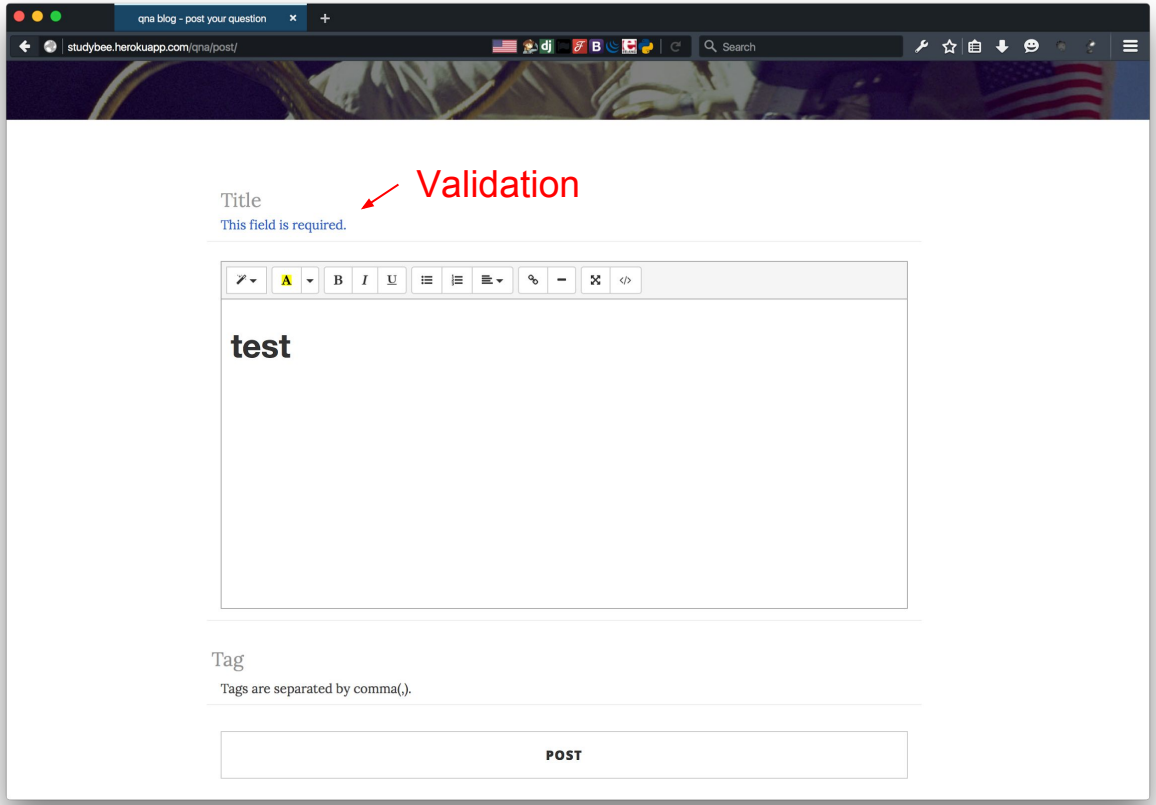
Feature 1/4 - Admin Page



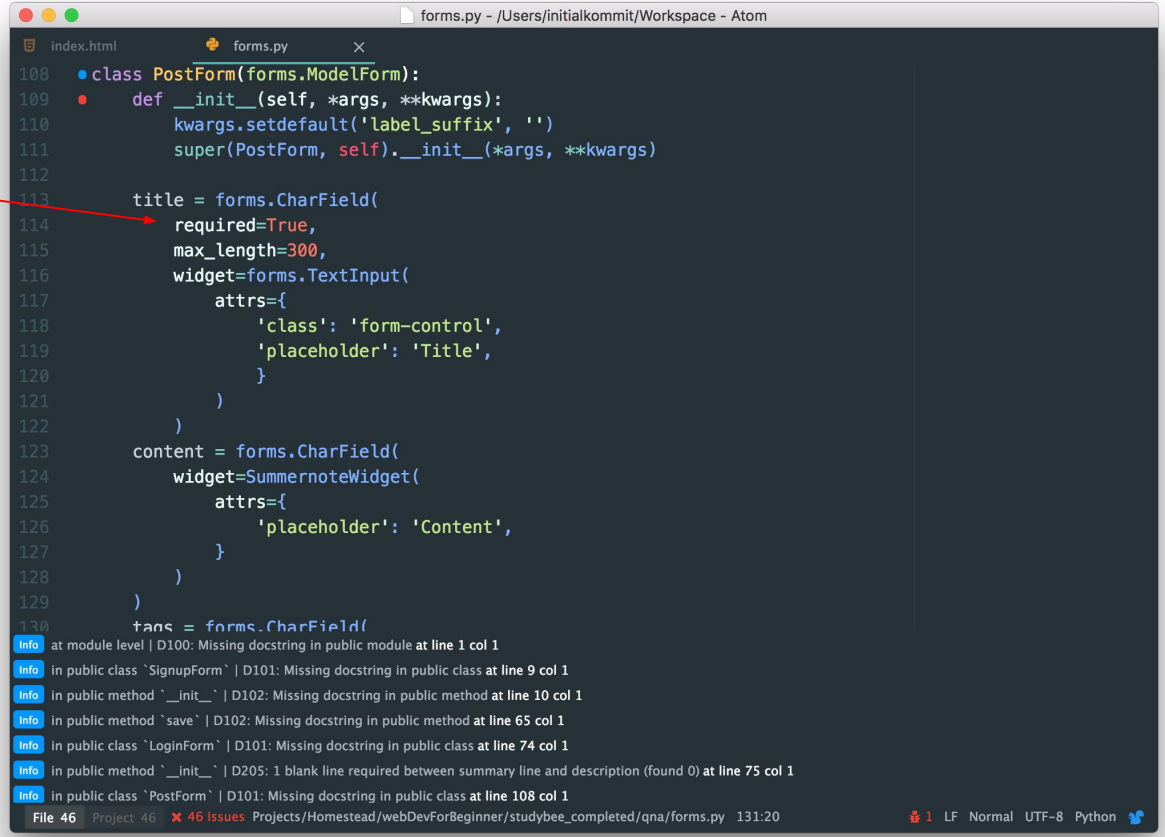
Feature 2/4 - Forms

- 사용자로부터 특정값을 입력받는 양식
- 그러나 언제나 휴먼 에러는 발생
- 따라서 유효성 검사(Validation)을 해야할 필요가 있음
  - 예를 들어
    - 숫자를 입력해야 하는데 문자를 입력
- Django는
  - 입력을 받는 양식과
  - 유효성 검사를 하는 과정과
  - 입력받은 값을 데이터베이스에 저장하는 과정
- 을 기능으로 만들어 편리하게 처리할 수 있게 해주고 있음

Feature 2/4 - Forms



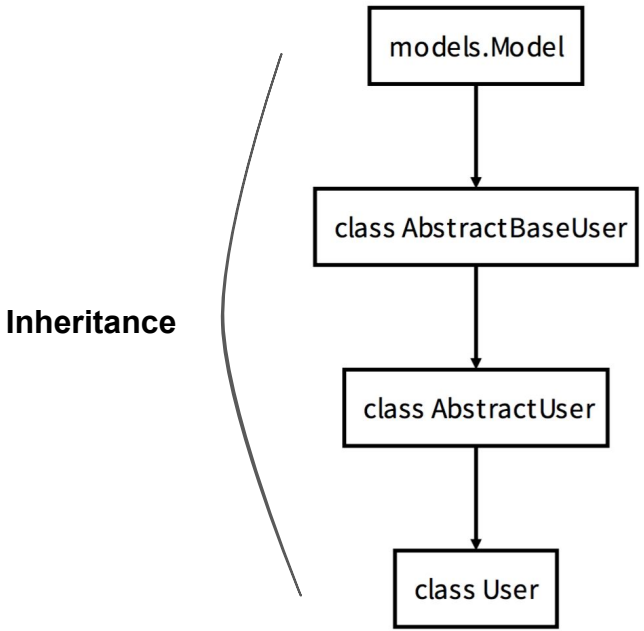
Feature 2/4 - Forms



Feature 3/4 - User Authentication

- 이미 만들어져 있는 User Model
- 이미 만들어져 있는 User와 관련된 기능들
  - 패스워드 암호화(Hash)
  - 회원가입 폼
  - 패스워드 리셋 폼
  - 로그인
  - 로그아웃
  - 등
- 심지어 Group과 Permission 기능도 포함되어 있다!
- Django의 User Model에서 간단하게 확장 가능

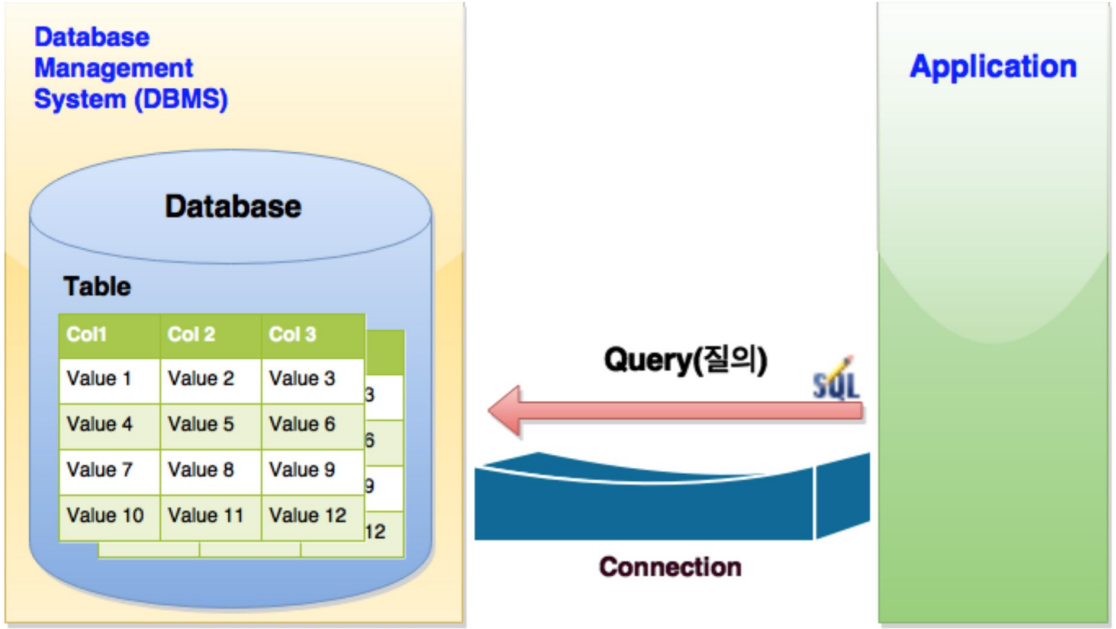
Feature 3/4 - User Authentication



User Schema

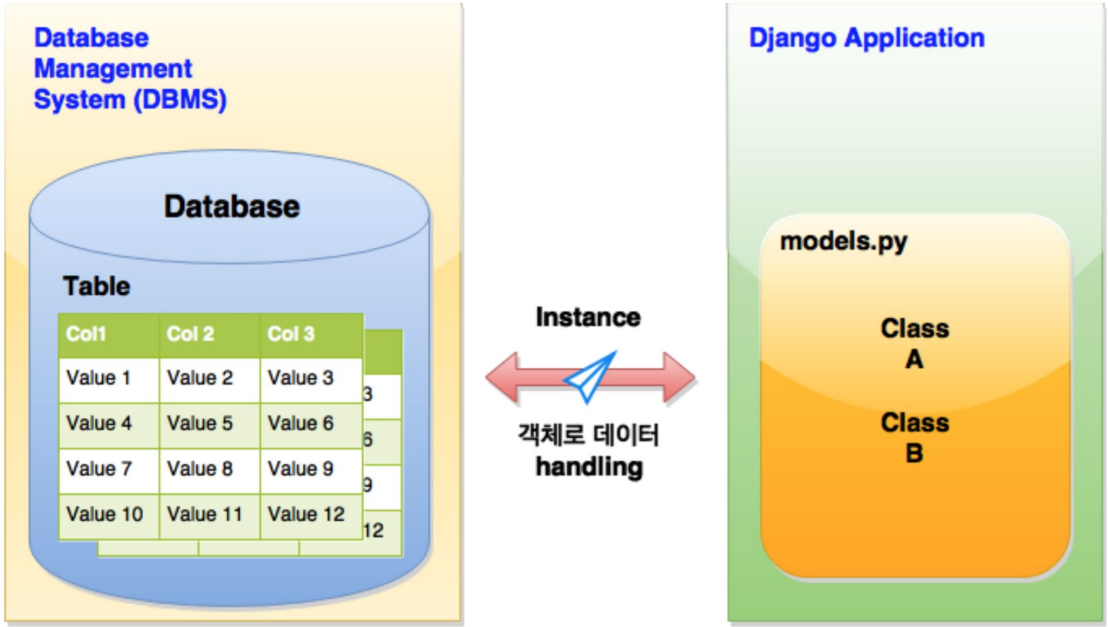
필드명	Django 필드타입	DB 필드타입	제약조건
id	AutoField	Integer	Primary Key
username	CharField	varchar(30)	Unique
password	CharField	varchar(128)	
last_login	DateTimeField	datetime	
first_name	CharField	varchar(30)	
last_name	CharField	varchar(30)	
email	EmailField	varchar(75)	
is_staff	BooleanField	bool	
is_active	BooleanField	bool	
date_joined	DateTimeField	datetime	
is_superuser	BooleanField	bool	

Feature 4/4 - ORM(Object-Relational Mapping)



[어플리케이션이 데이터를 다루는 방법]

Feature 4/4 - ORM(Object-Relational Mapping)



[Django에서의 ORM]

Feature 4/4 - ORM(Object-Relational Mapping)

SQL vs. ORM

항목	방법
SQL	Select username from USER where id=1;
ORM	User.objects.get(id=1).username

5. 하늘공원

Hello World!

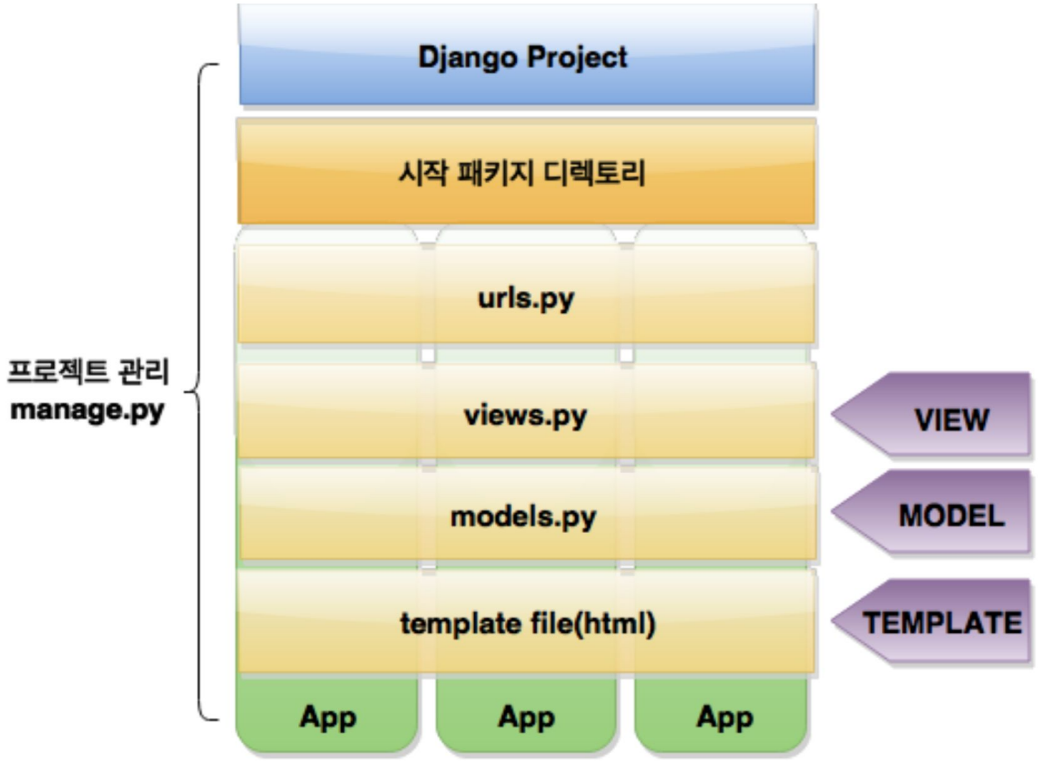
Prerequisite

1. 나는 Python 기본 문법은 알고 있다.
2. 나는 가상환경이 무엇인지 알고 만들 수 있다.
3. 나는 pip이 무엇인지 설명할 수 있다.
4. 나는 HTML, CSS 등 프론트 엔드 기술을 조금은 다룰 수 있다.

Django Installation

1. 가상환경에서
2. pip install django
3. pip freeze or pip list로 확인

Django의 기본 구조



Hello World 프로젝트를 만들어 보겠습니다.

1. django-admin.py startproject helloworld

2. cd helloworld

3. python manage.py startapp say

프로젝트 구조

```
.
├── db.sqlite3
├── helloworld
|   ├── __init__.py
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
├── manage.py
└── say
    ├── __init__.py
    ├── admin.py
    ├── migrations
    |   └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py
```

django-admin.py startproject

MTV 패턴에 맞춰 파일을 자동생성 해주는 명령어

프로젝트 구조

```
.
├── db.sqlite3
├── helloworld
|   ├── __init__.py
|   ├── __pycache__
|   |   ├── __init__.cpython-35.pyc
|   |   ├── settings.cpython-35.pyc
|   |   ├── urls.cpython-35.pyc
|   |   └── wsgi.cpython-35.pyc
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
├── manage.py
└── say
    ├── __init__.py
    ├── __pycache__
    |   ├── __init__.cpython-35.pyc
    |   ├── admin.cpython-35.pyc
    |   ├── models.cpython-35.pyc
    |   └── views.cpython-35.pyc
    ├── admin.py
    ├── migrations
    |   ├── __init__.py
    |   └── __pycache__
    |       └── __init__.cpython-35.pyc
    ├── models.py
    ├── tests.py
    └── views.py
```

서버를 실행하면 pyc 파일이 생깁니다.

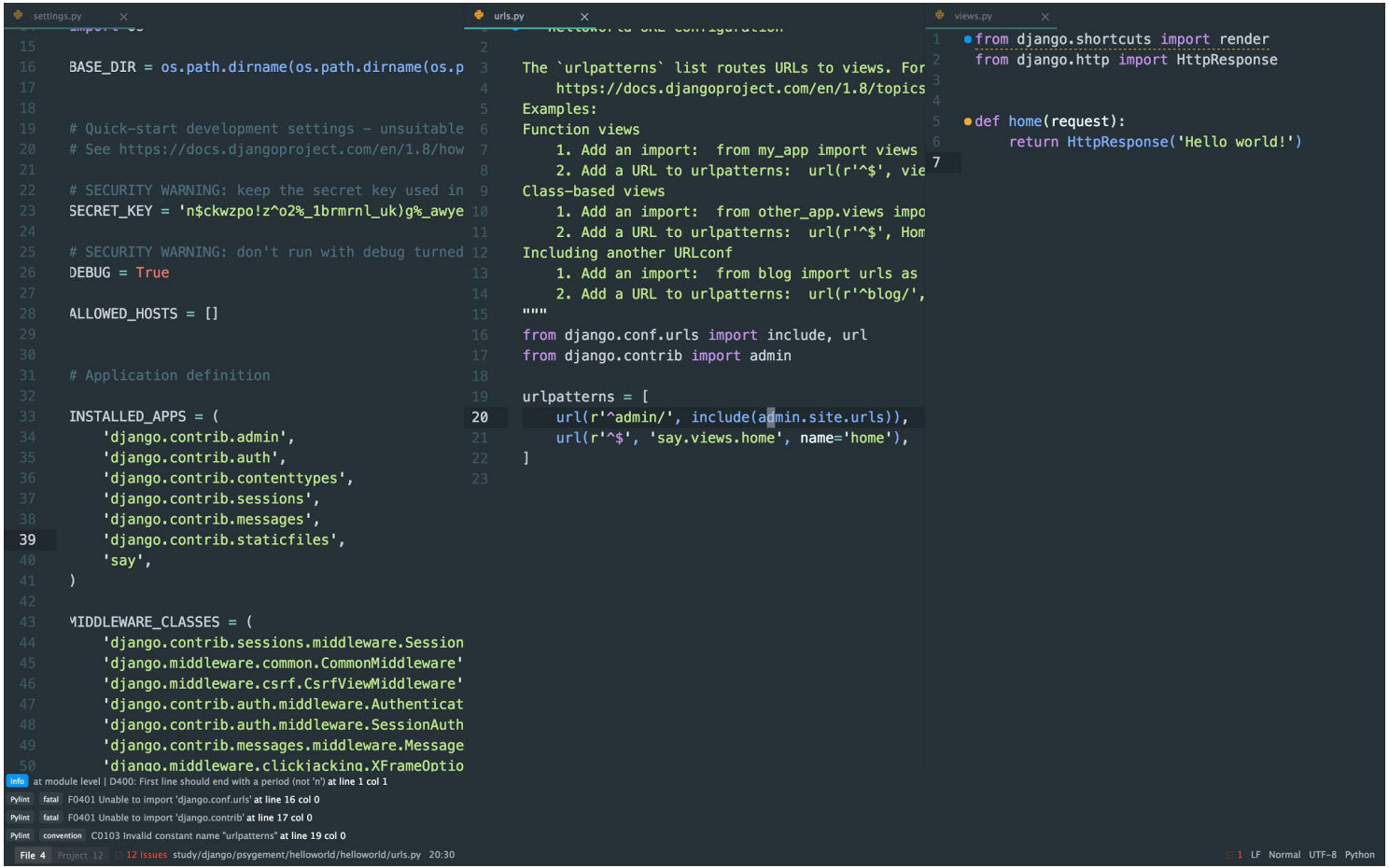
***.pyc file**

pyc 파일은 Byte code로 Compiled된 파일을 말한다. 이는 python으로 실행하면 자동으로 생성되며, Script source가 수정되면 역시 자동으로 Compile된다. Script가 Compile되는 이유는 성능의 문제이기 때문이며 파일 이름에 cpython이라고 되어있는 것은 우리가 C언어로 만든 Python을 사용하기 때문이다. 자동으로 생성되기 때문에 지워도 상관없으며 일반적으로 소스 이력 관리 시 pyc 파일은 포함시키지 않는다.

settings.py

urls.py

say/views.py



settings.py

```
15
16  BASE_DIR = os.path.dirname(os.path.dirname(os.p
17
18
19  # Quick-start development settings - unsuitable
20  # See https://docs.djangoproject.com/en/1.8/how
21
22  # SECURITY WARNING: keep the secret key used in
23  SECRET_KEY = 'n$ckwzpo!z^o2%_1brmrnl_uk)g%_awye
24
25  # SECURITY WARNING: don't run with debug turned
26  DEBUG = True
27
28  ALLOWED_HOSTS = []
29
30
31  # Application definition
32
33  INSTALLED_APPS = (
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'say',
41  )
42
43  MIDDLEWARE_CLASSES = (
44      'django.contrib.sessions.middleware.Session
45      'django.middleware.common.CommonMiddleware'
46      'django.middleware.csrf.CsrfViewMiddleware'
47      'django.contrib.auth.middleware.Authenticat
48      'django.contrib.auth.middleware.SessionAuth
49      'django.contrib.messages.middleware.Message
50      'django.middleware.clickjacking.XFrameOptio
```

urls.py

```
2
3   The `urlpatterns` list routes URLs to views. For
4       https://docs.djangoproject.com/en/1.8/topics
5   Examples:
6   Function views
7       1. Add an import:  from my_app import views
8       2. Add a URL to urlpatterns:  url(r'^$', vie
9   Class-based views
10      1. Add an import:  from other_app.views impo
11      2. Add a URL to urlpatterns:  url(r'^$', Hom
12  Including another URLconf
13      1. Add an import:  from blog import urls as
14      2. Add a URL to urlpatterns:  url(r'^blog/',
15  """
16  from django.conf.urls import include, url
17  from django.contrib import admin
18
19  urlpatterns = [
20      url(r'^admin/', include(admin.site.urls)),
21      url(r'^$', 'say.views.home', name='home'),
22  ]
23
```

views.py

```
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4
5  def home(request):
6      return HttpResponse('Hello world!')
7
```

Info at module level | D400: First line should end with a period (not 'n') at line 1 col 1

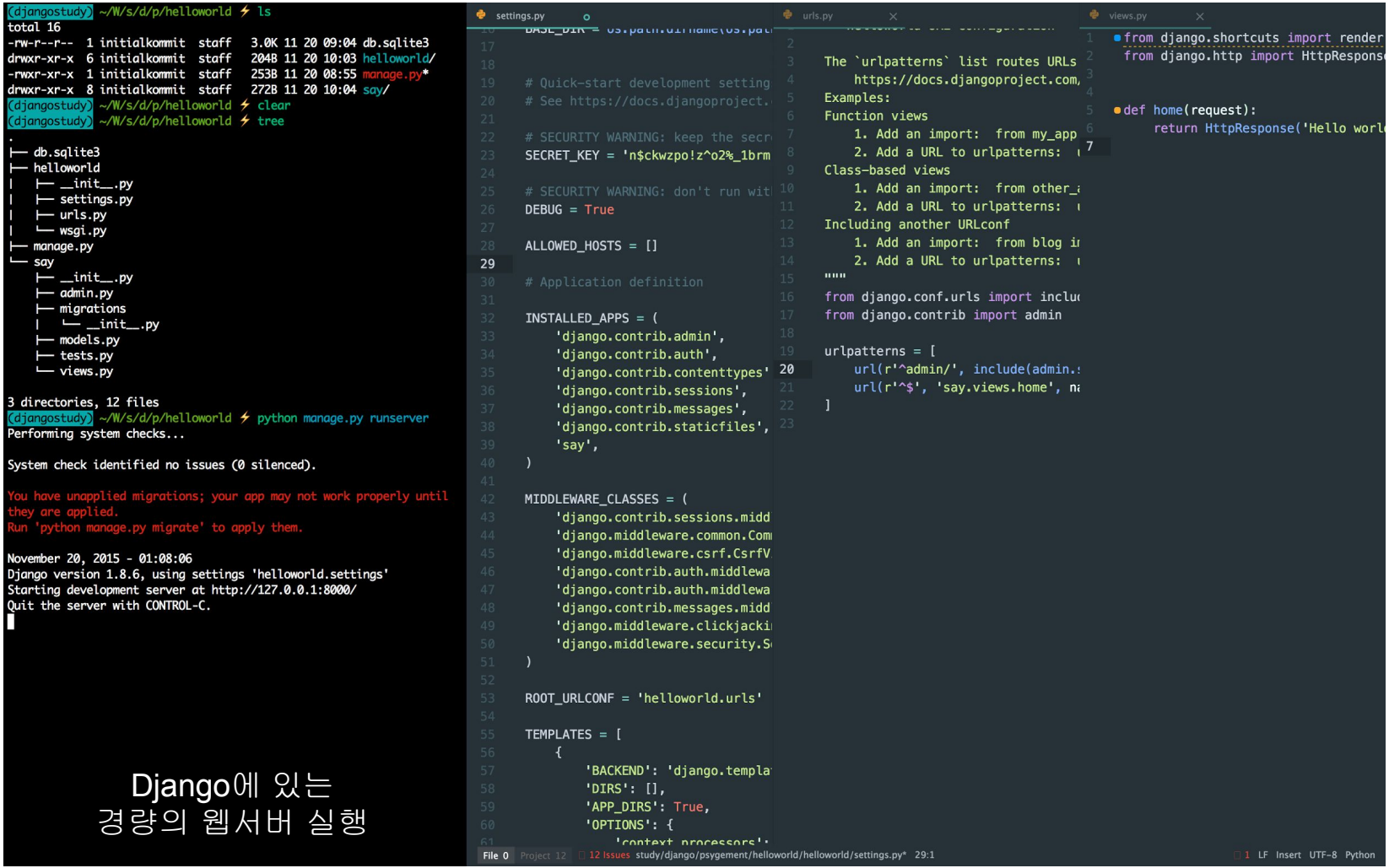
Pylint fatal F0401 Unable to import 'django.conf.urls' at line 16 col 0

Pylint fatal F0401 Unable to import 'django.contrib' at line 17 col 0

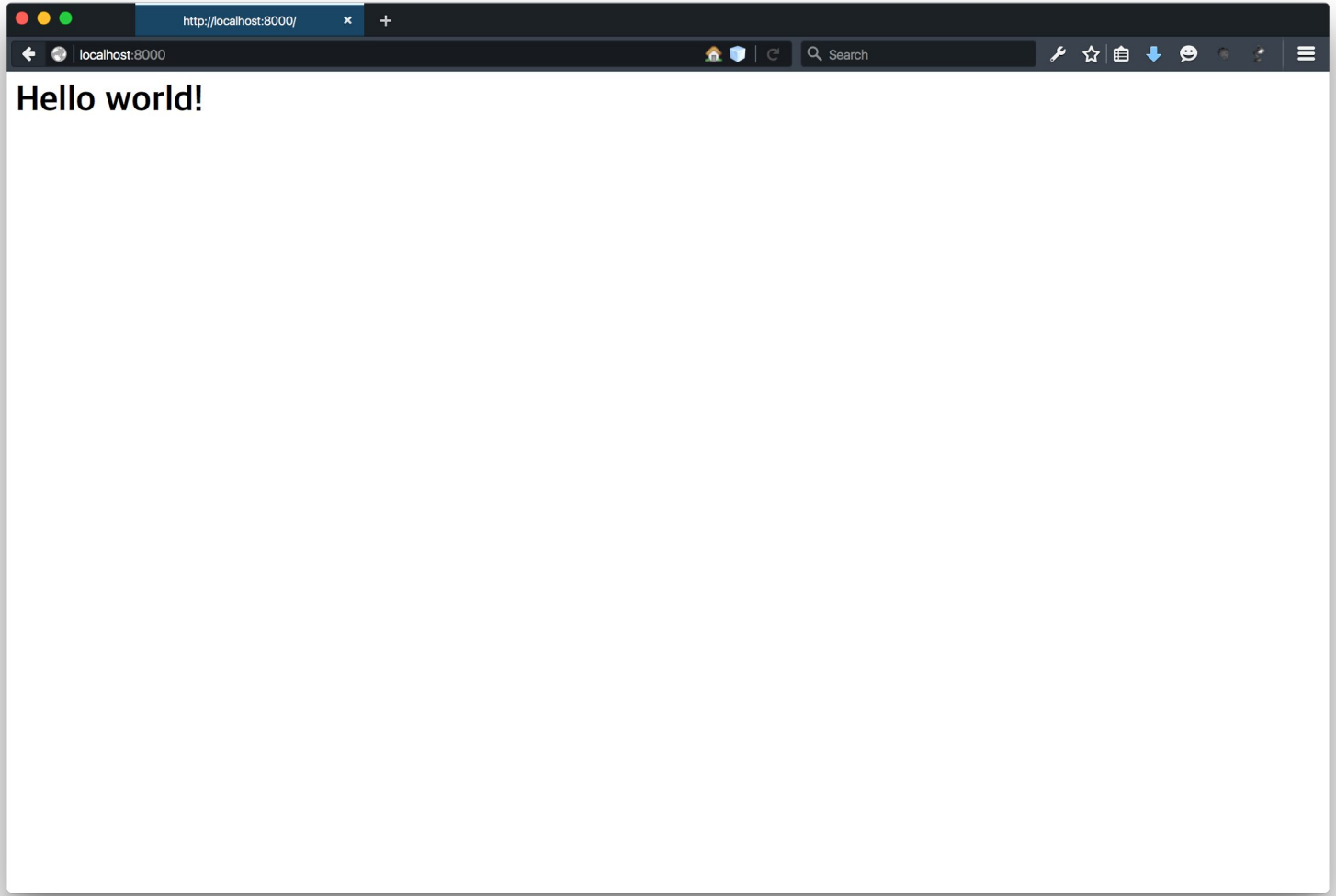
Pylint convention C0103 Invalid constant name "urlpatterns" at line 19 col 0

File 4 Project 12 12 issues study/django/psygement/helloworld/urls.py 20:30

1 LF Normal UTF-8 Python



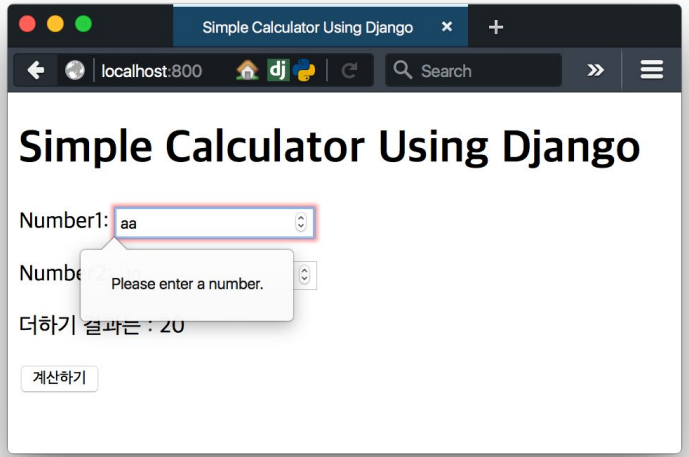
Django에 있는
경량의 웹서버 실행

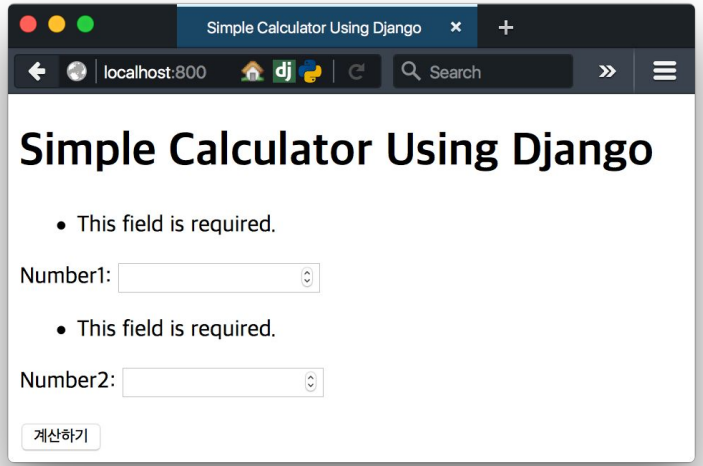

http://localhost:8000/
localhost:8000
Search
Hello world!

6. 남산

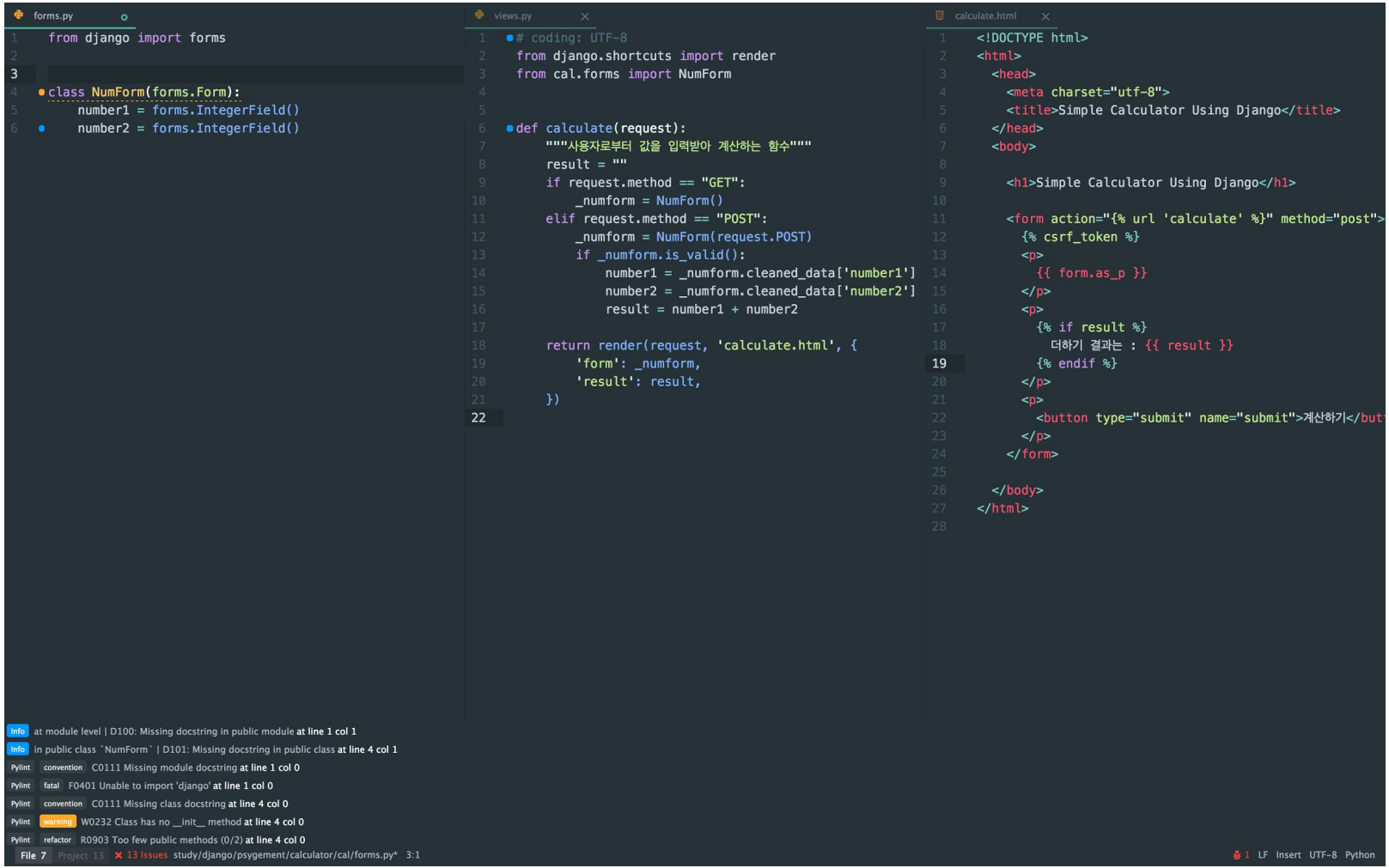
calculator

Form을 이용한 더하기 계산





Django의 Form 기능으로 Validation check를 해주고 있는 모습



settings.py에서 template 디렉토리 설정한 후 작업

6. 관악산

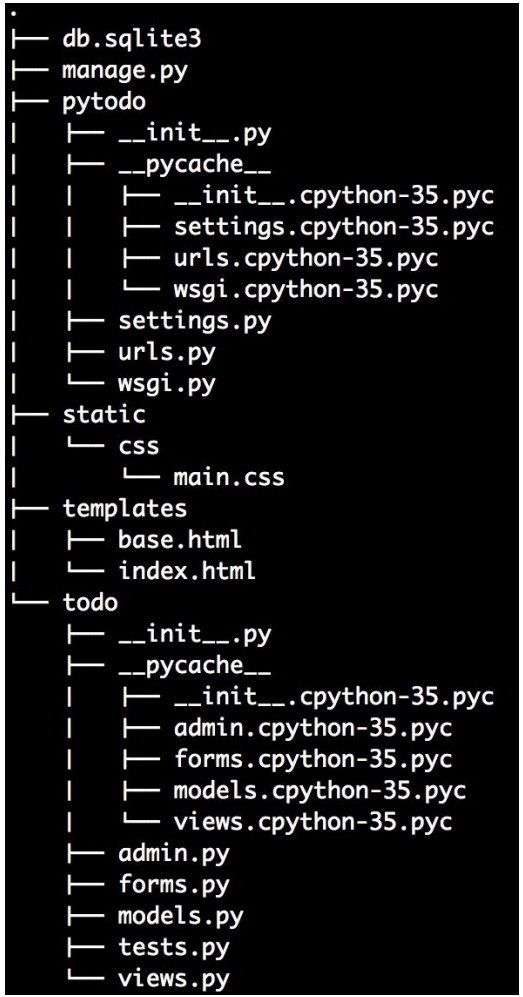
예제 - todo list

본 예제는 절대 많은 기능이 포함되어 있지 않으니 무서워하지 마세요.

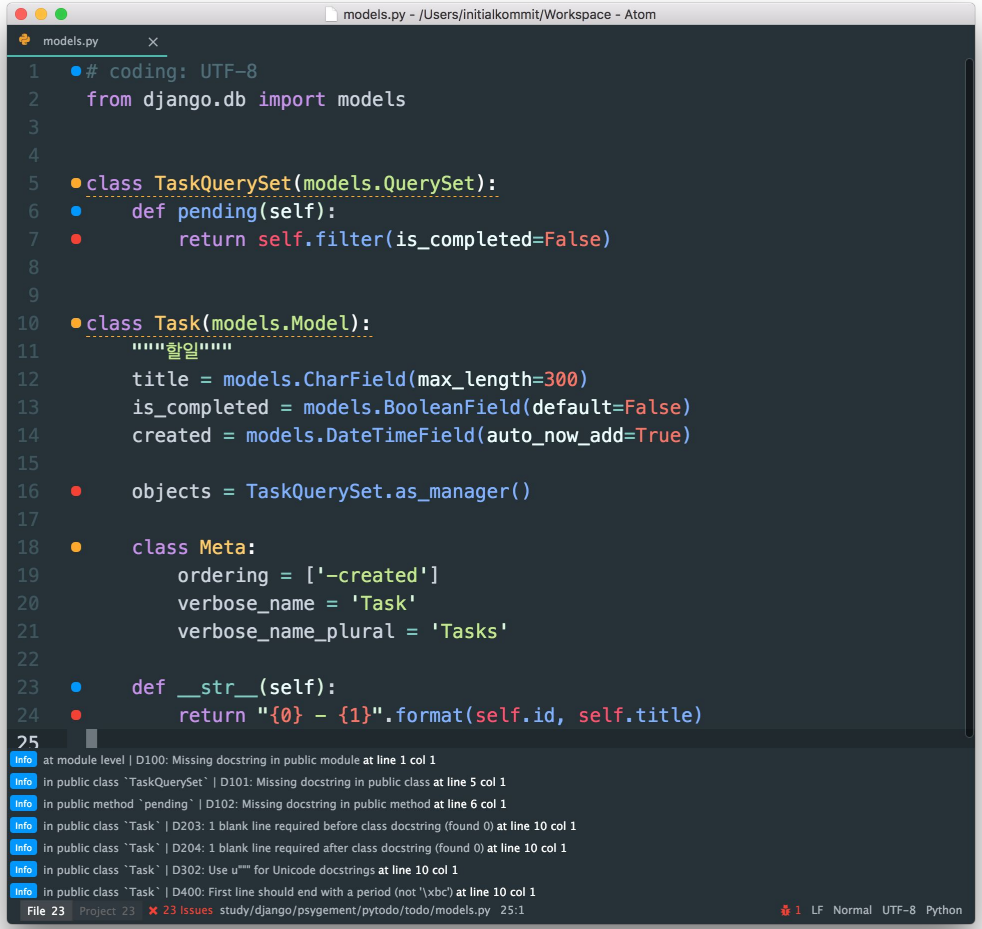
Specification

- Django 1.8.6
- Python 3.5
- DB: Sqlite3
  - Task
    - id, autoincrement
    - title, charfield
    - is_completed, booleanfield
    - created, datetimefield

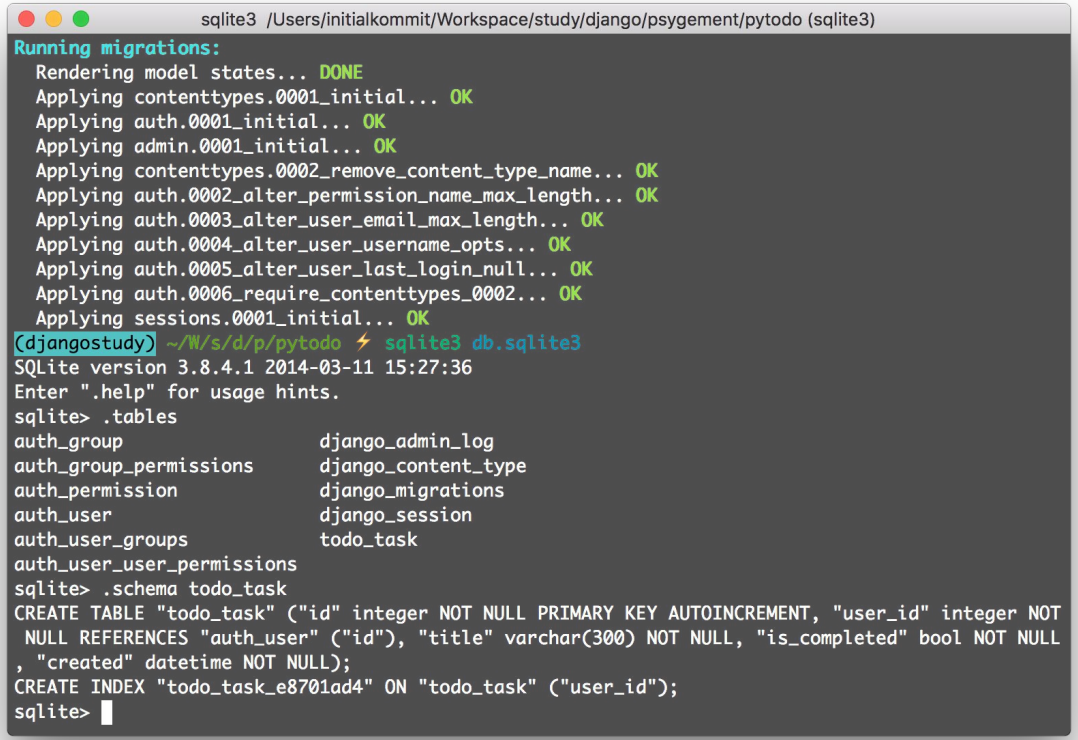
구조


.
├── db.sqlite3
├── manage.py
├── pytodo
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-35.pyc
│   │   ├── settings.cpython-35.pyc
│   │   ├── urls.cpython-35.pyc
│   │   └── wsgi.cpython-35.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── static
│   └── css
│   └── main.css
├── templates
│   ├── base.html
│   └── index.html
└── todo
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-35.pyc
    │   ├── admin.cpython-35.pyc
    │   ├── forms.cpython-35.pyc
    │   ├── models.cpython-35.pyc
    │   └── views.cpython-35.pyc
    ├── admin.py
    ├── forms.py
    ├── models.py
    ├── tests.py
    └── views.py

Model

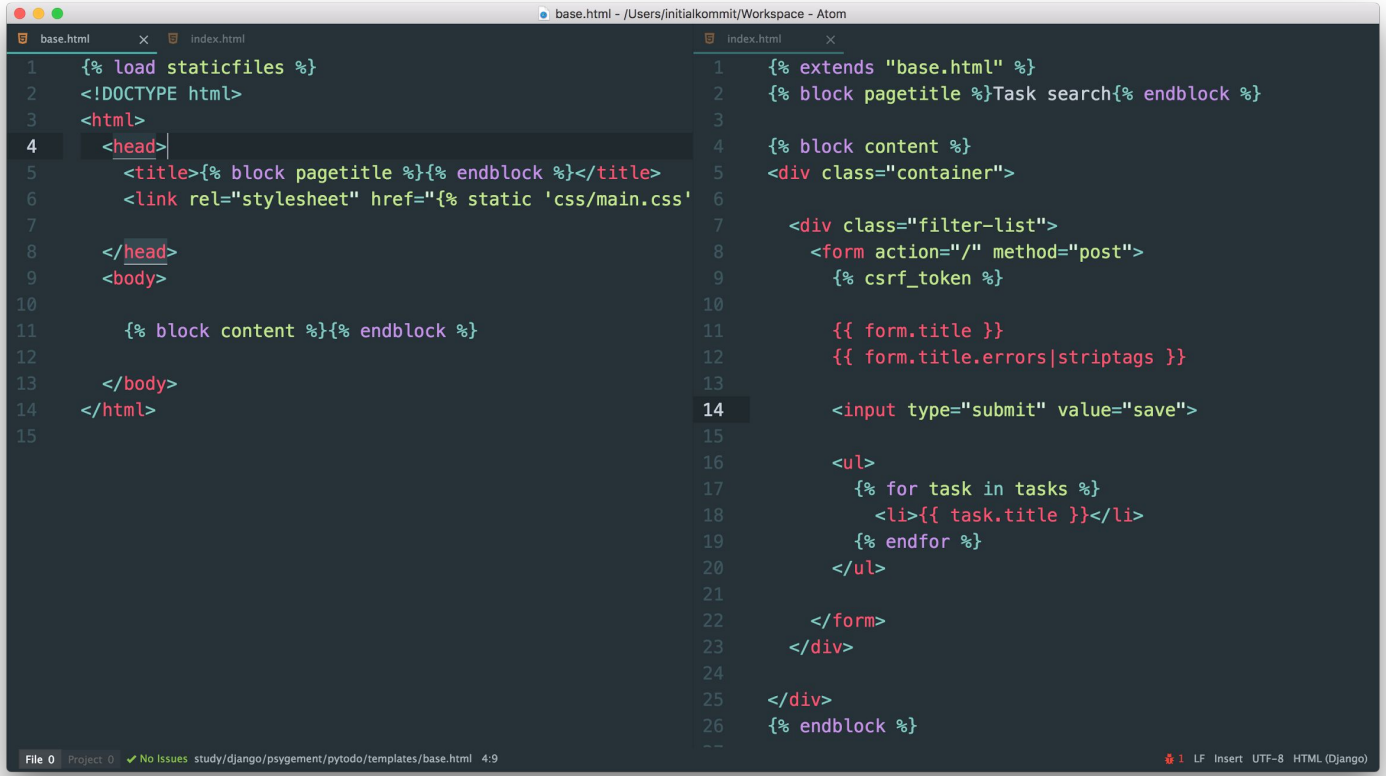

models.py - /Users/initialkommit/Workspace - Atom
models.py
# coding: UTF-8
from django.db import models
class TaskQuerySet(models.QuerySet):
def pending(self):
return self.filter(is_completed=False)
class Task(models.Model):
"""할일"""
title = models.CharField(max_length=300)
is_completed = models.BooleanField(default=False)
created = models.DateTimeField(auto_now_add=True)
objects = TaskQuerySet.as_manager()
class Meta:
ordering = ['-created']
verbose_name = 'Task'
verbose_name_plural = 'Tasks'
def __str__(self):
return "{0} - {1}".format(self.id, self.title)
Info at module level | D100: Missing docstring in public module at line 1 col 1
Info in public class `TaskQuerySet` | D101: Missing docstring in public class at line 5 col 1
Info in public method `pending` | D102: Missing docstring in public method at line 6 col 1
Info in public class `Task` | D203: 1 blank line required before class docstring (found 0) at line 10 col 1
Info in public class `Task` | D204: 1 blank line required after class docstring (found 0) at line 10 col 1
Info in public class `Task` | D302: Use u""" for Unicode docstrings at line 10 col 1
Info in public class `Task` | D400: First line should end with a period (not '\xbc') at line 10 col 1
File 23 Project 23 23 Issues study/django/psygement/pytodo/todo/models.py 25:1
1 LF Normal UTF-8 Python

migrations



위에서 정의 models.py를 통해 DB를 만들 수 있는 Scripts를 만들고
이를 통해 실제로 물리적 데이터베이스를 구현합니다.

Template


base.html - /Users/initialkommit/Workspace - Atom
base.html
index.html
{% load staticfiles %}
<!DOCTYPE html>
<html>
<head>
<title>{% block pagetitle %}{% endblock %}</title>
<link rel="stylesheet" href="{% static 'css/main.css'
</head>
<body>
{% block content %}{% endblock %}
</body>
</html>
index.html
{% extends "base.html" %}
{% block pagetitle %}Task search{% endblock %}
{% block content %}

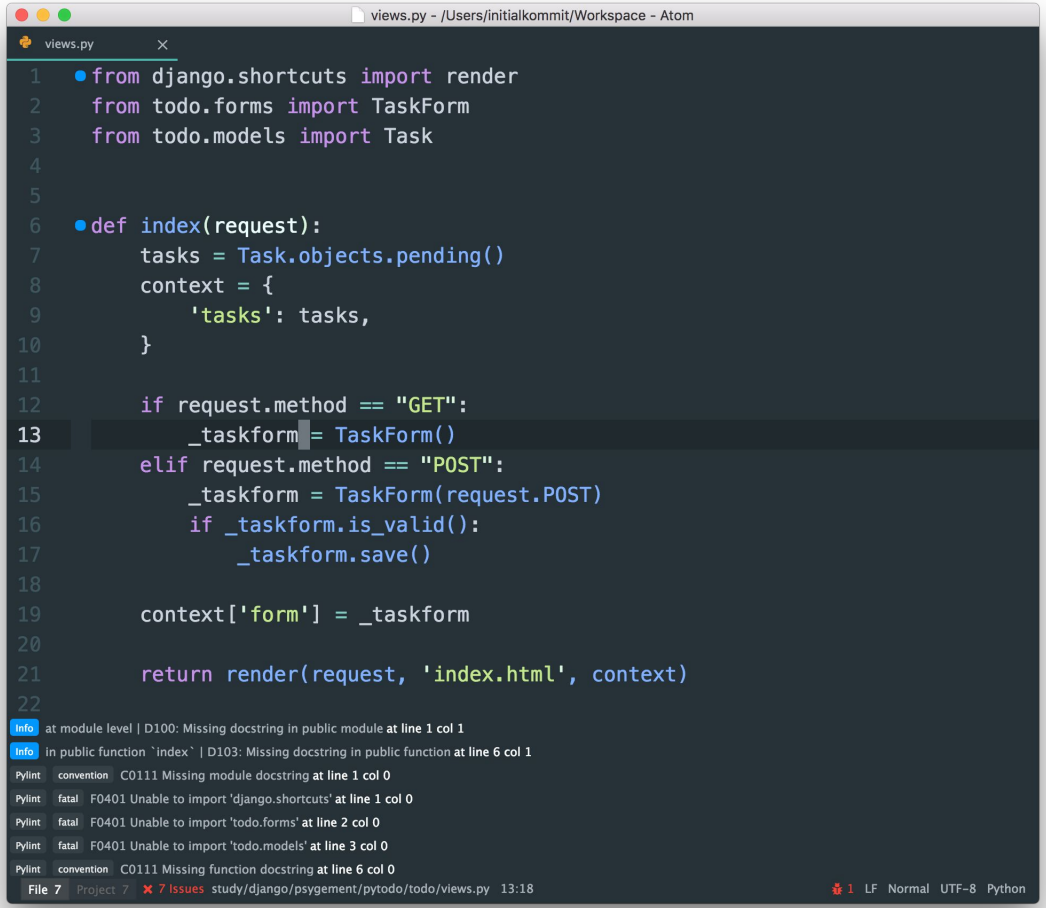
<form action="/" method="post">
{% csrf_token %}
{{ form.title }}
{{ form.title.errors|striptags }}
<input type="submit" value="save">

{% for task in tasks %}
{{ task.title }}
{% endfor %}

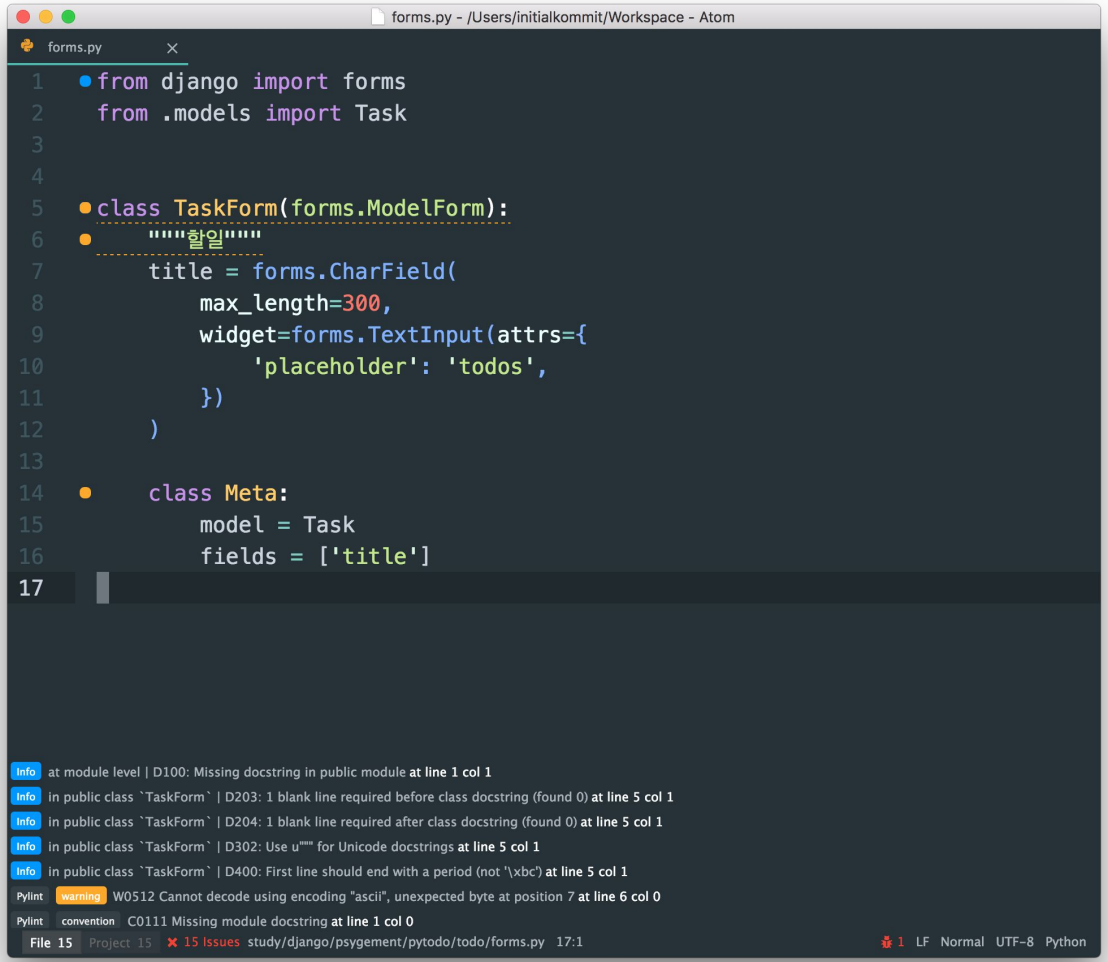
</form>

{% endblock %}
File 0
Project 0
No Issues
study/django/psygement/pytodo/templates/base.html
4:9
1
LF
Insert
UTF-8
HTML (Django)

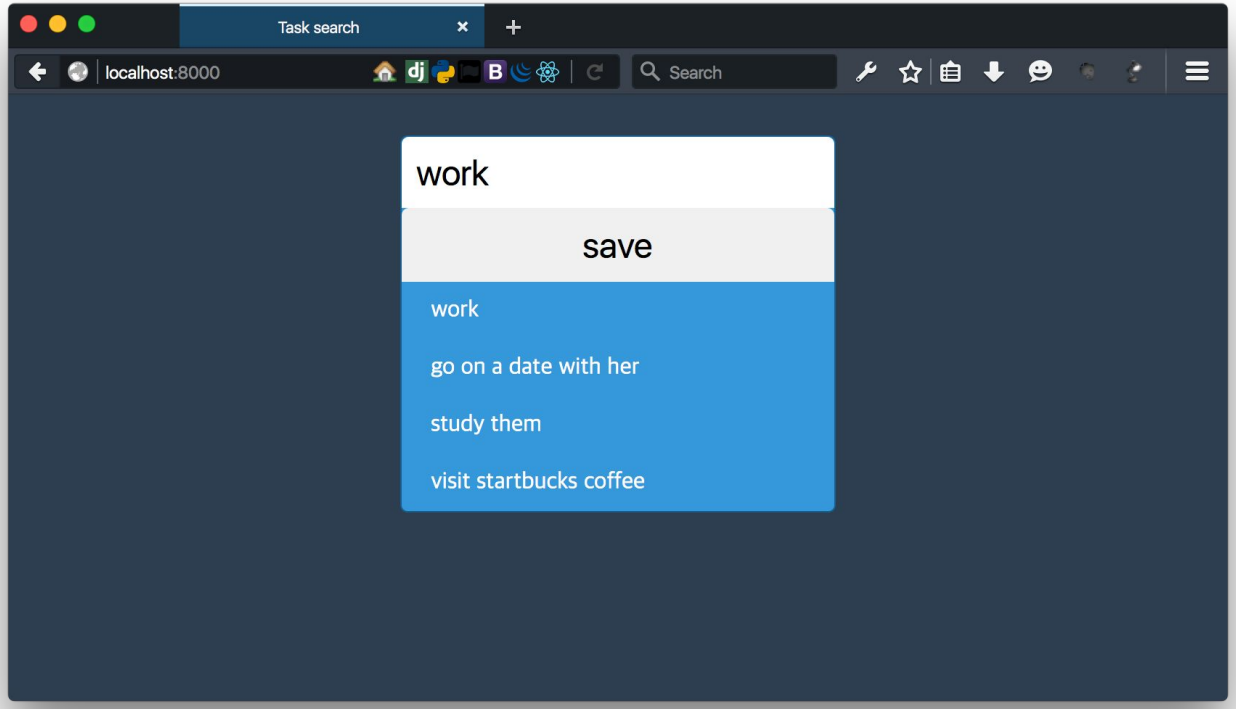
View


views.py - /Users/initialkommit/Workspace - Atom
views.py
1 from django.shortcuts import render
2 from todo.forms import TaskForm
3 from todo.models import Task
4
5
6 def index(request):
7 tasks = Task.objects.pending()
8 context = {
9 'tasks': tasks,
10 }
11
12 if request.method == "GET":
13 _taskform = TaskForm()
14 elif request.method == "POST":
15 _taskform = TaskForm(request.POST)
16 if _taskform.is_valid():
17 _taskform.save()
18
19 context['form'] = _taskform
20
21 return render(request, 'index.html', context)
22
Info at module level | D100: Missing docstring in public module at line 1 col 1
Info in public function `index` | D103: Missing docstring in public function at line 6 col 1
Pylint convention C0111 Missing module docstring at line 1 col 0
Pylint fatal F0401 Unable to import 'django.shortcuts' at line 1 col 0
Pylint fatal F0401 Unable to import 'todo.forms' at line 2 col 0
Pylint fatal F0401 Unable to import 'todo.models' at line 3 col 0
Pylint convention C0111 Missing function docstring at line 6 col 0
File 7 Project 7 7 Issues study/django/psygement/pytodo/todo/views.py 13:18
1 LF Normal UTF-8 Python

Form


forms.py - /Users/initialkommit/Workspace - Atom
forms.py
1 from django import forms
2 from .models import Task
3
4
5 class TaskForm(forms.ModelForm):
6 """할일"""
7 title = forms.CharField(
8 max_length=300,
9 widget=forms.TextInput(attrs={
10 'placeholder': 'todos',
11 })
12)
13
14 class Meta:
15 model = Task
16 fields = ['title']
17
Info at module level | D100: Missing docstring in public module at line 1 col 1
Info in public class `TaskForm` | D203: 1 blank line required before class docstring (found 0) at line 5 col 1
Info in public class `TaskForm` | D204: 1 blank line required after class docstring (found 0) at line 5 col 1
Info in public class `TaskForm` | D302: Use u""" for Unicode docstrings at line 5 col 1
Info in public class `TaskForm` | D400: First line should end with a period (not '\xbc') at line 5 col 1
Pylint warning W0512 Cannot decode using encoding "ascii", unexpected byte at position 7 at line 6 col 0
Pylint convention C0111 Missing module docstring at line 1 col 0
File 15 Project 15 15 Issues study/django/psygement/pytodo/todo/forms.py 17:1
1 LF Normal UTF-8 Python

todo list



완성 처리 등의 기능이 없습니다. 생각해서 넣어보면 좋겠습니다.

7. 백두산

RESTful Service

예제없이 개념만 집고 넘어갑니다.

REST(REpresentational State Transfer)

6가지 아키텍처 스타일을 조합한 REST

- 클라이언트/서버 : 유저 인터페이스와 처리를 분리한다.
- 스테이트리스 서버 : 서버 측에서 애플리케이션의 상태를 가지지 않는다.
- 캐시 : 클라이언트와 서버의 통신 횟수와 양을 감소시킨다.
- 유니폼 인터페이스 : 인터페이스를 고정한다.
- 계층화 시스템 : 시스템을 계층별로 분리한다.
- 코드 온 디맨드 : 프로그램을 클라이언트에 다운로드하여 실행한다.

REST(REpresentational State Transfer)

햄버거 가게 예시 : 스테이트리스와 스테이트풀Stateful한 대화를 들어본다.

- 스테이트풀한 대화

손님 : 안녕하세요
점원 : 어서 오세요. **햄버거입니다~
손님 : 햄버거 세트 하나 주세요
점원 : 사이드 메뉴는 무엇으로 하시겠습니까?
손님 : 포테이토요
점원 : 음료수는 무엇으로 하시겠습니까?
손님 : 콜라로 주세요
점원 : 500원 추가하시면 음료수를 L사이즈로 할 수 있는데 어떠신지요?
손님 : 그냥 M사이즈로 주세요
점원 : 추가하실 건 없으시고요?
손님 : 네
점원 : 알겠습니다.

REST(REpresentational State Transfer)

햄버거 가게 예시 : 스테이트리스와 스테이트풀Stateful한 대화를 들어본다.

- 스테이트리스한 대화

손님 : 안녕하세요
점원 : 어서 오세요. **햄버거입니다~
손님 : 햄버거 세트 하나 주세요
점원 : 사이드 메뉴는 무엇으로 하시겠습니까?
손님 : 햄버거 세트랑 포테이토요
점원 : 음료수는 무엇으로 하시겠습니까?
손님 : 햄버거 세트랑 포테이토랑 콜라로 주세요
점원 : 500원 추가하시면 음료수를 L사이즈로 할 수 있는데 어떠신지요?
손님 : 햄버거 세트랑 포테이토랑 콜라는 M사이즈로 주세요
점원 : 추가하실 건 없으시고요?
손님 : 햄버거 세트랑 포테이토랑 콜라는 M사이즈로 주세요. 이상
점원 : 알겠습니다.

REST(REpresentational State Transfer)

햄버거 가게 예시 : 스테이트리스와 스테이트풀Stateful한 대화를 들어본다.

- 스테이트풀의 결점

서버가 클라이언트의 애플리케이션 상태를 기억하는 것은 클라이언트의 수가 증가함에 따라 어려워지게 된다. 하나의 서버가 동시에 상대할 수 있는 클라이언트 수에는 한계가 있다. 또한 복수의 서버 간에 애플리케이션 상태를 동기화하여 사용할 수 있도록 해야하는데 이렇게 되면 동기화에 필요한 오버헤드를 커지게 된다. 이처럼 스테이트풀한 아키텍처로는 클라이언트의 수가 증가했을 경우 규모를 확장하기 어렵다.

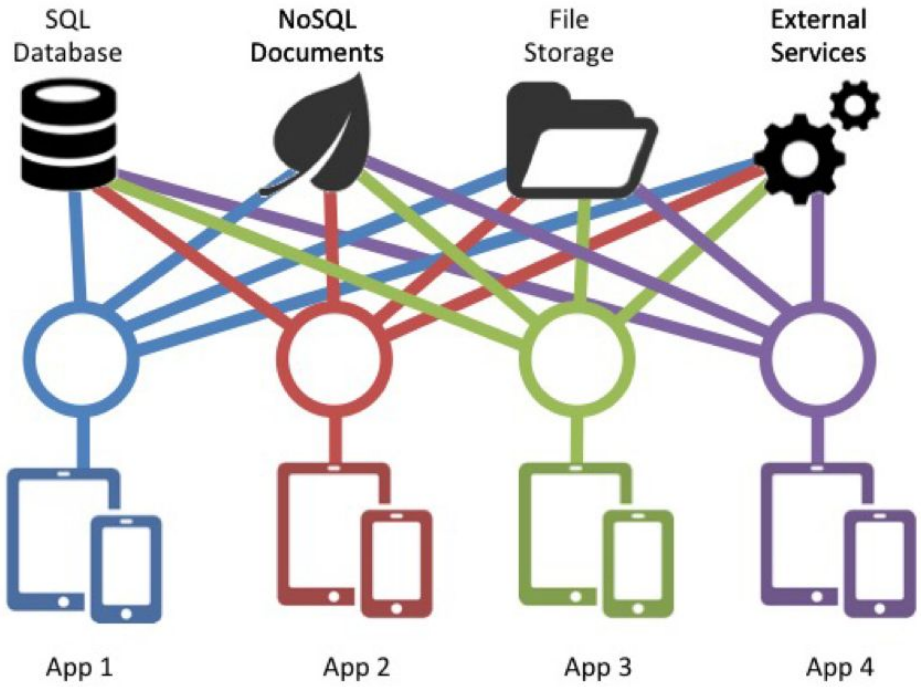
- 스테이트리스의 이점

이런 문제점을 해결한 것이 스테이트리스한 아키텍처이다. 스테이트리스는 클라이언트가 요청 메시지에 필요한 정보를 모두 포함시킨다. 요청을 처리하는데 필요한 정보가 모두 포함되어 있는 메시지를 가리켜 '자기 기술적 메시지Self Descriptive Message'라고 한다.
스테이트리스한 서버는 애플리케이션 상태를 기억할 필요가 없기 때문에 서버 시스템이 단순해진다. 서버는 이제까지의 일은 모두 잊고 새로 오는 요청을 처리하는데만 집중하면 되는 것이다. 따라서 시스템을 확장시키는 것이 단단해진다. 클라이언트가 늘어나면 단순히 서버를 증설하면 된다.

- 스테이트리스의 결점

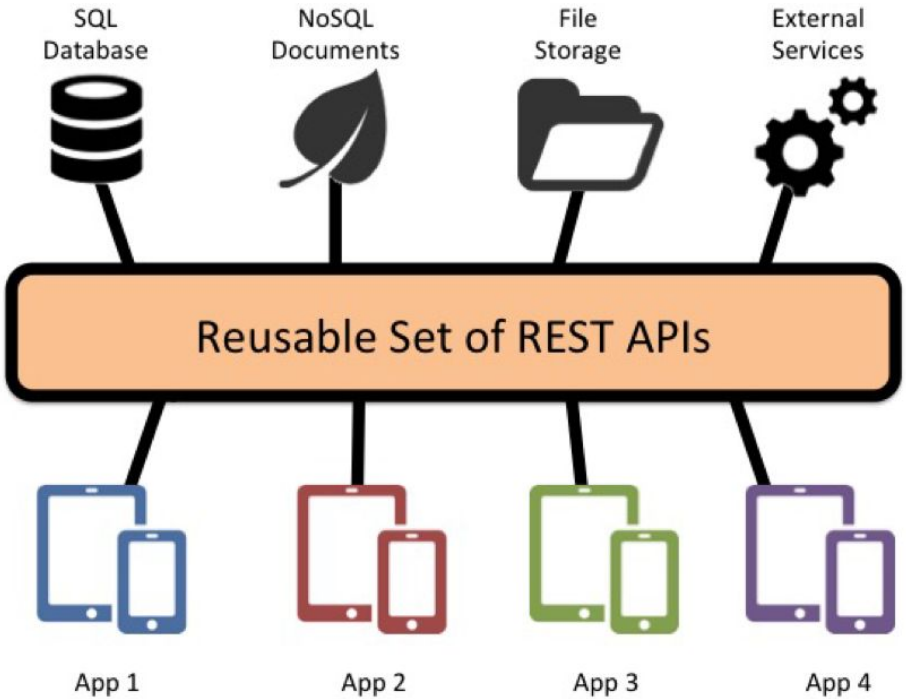
스테이트리스한 아키텍처는 확장성 면에서는 이점이 있지만 결점도 가지고 있다.
① 퍼포먼스 저하 : 송신할 데이터의 양이 많아지며, 인증 등 서버에 부하가 걸리는 처리를 반복한다.
② 통신 에러에 대한 대처 : 현재 애플리케이션 상태를 서버단에서는 알지 못하기 때문에 중간에 통신에 에러가 생기면 대처하기가 힘들다.

REST APIs



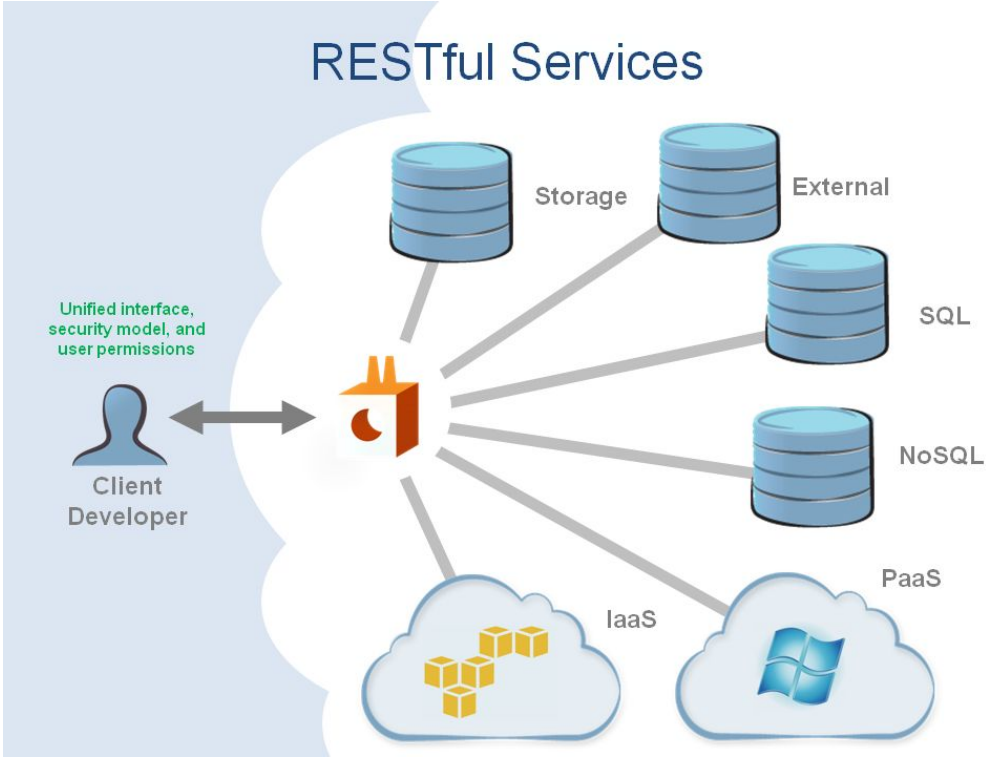
https://www.ibm.com/developerworks/community/blogs/dreamfactory

REST APIs



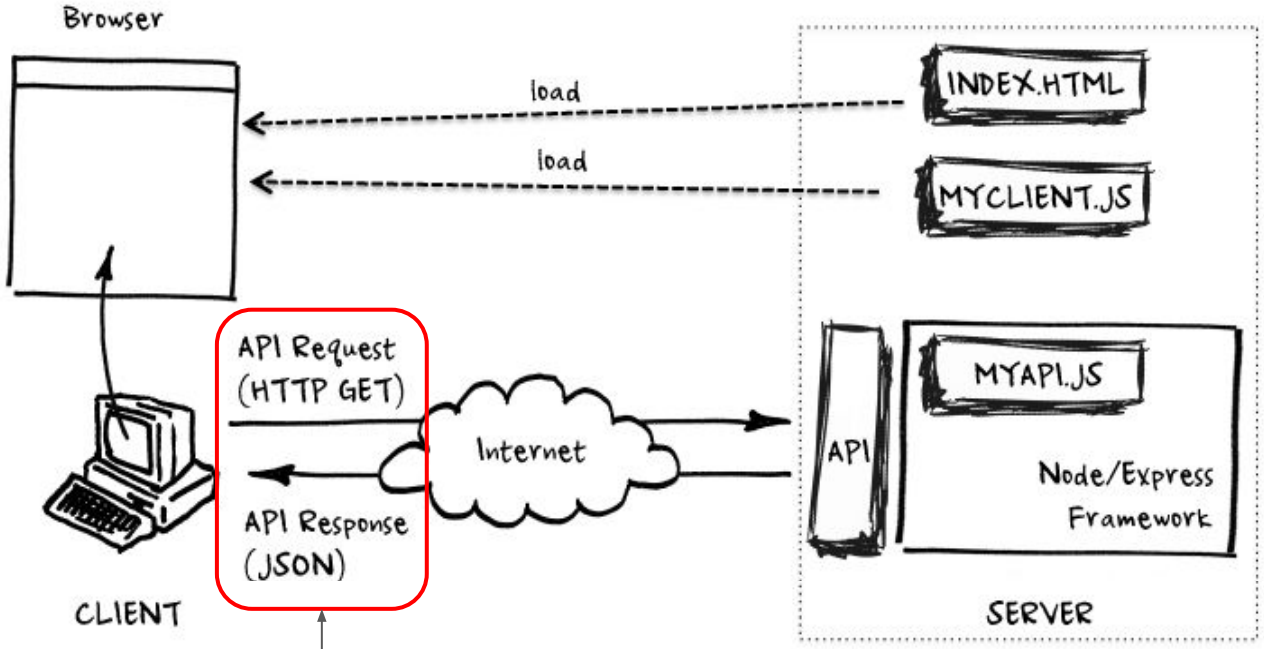
https://www.ibm.com/developerworks/community/blogs/dreamfactory

RESTful Services



https://www.ibm.com/developerworks/community/blogs/dreamfactory

REST가 사용하는 HTTP, JSON



** Serializer - Database에서 Query해 온 Data를 JSON으로 만들어주고 또한 반대로 만들어 줌

https://thecustomizewindows.com/2014/08/restful-api-for-wordpress-to-enable-public-http-get-request-for-posts/

REST가 사용하는 HTTP, JSON

HTTP 프로토콜

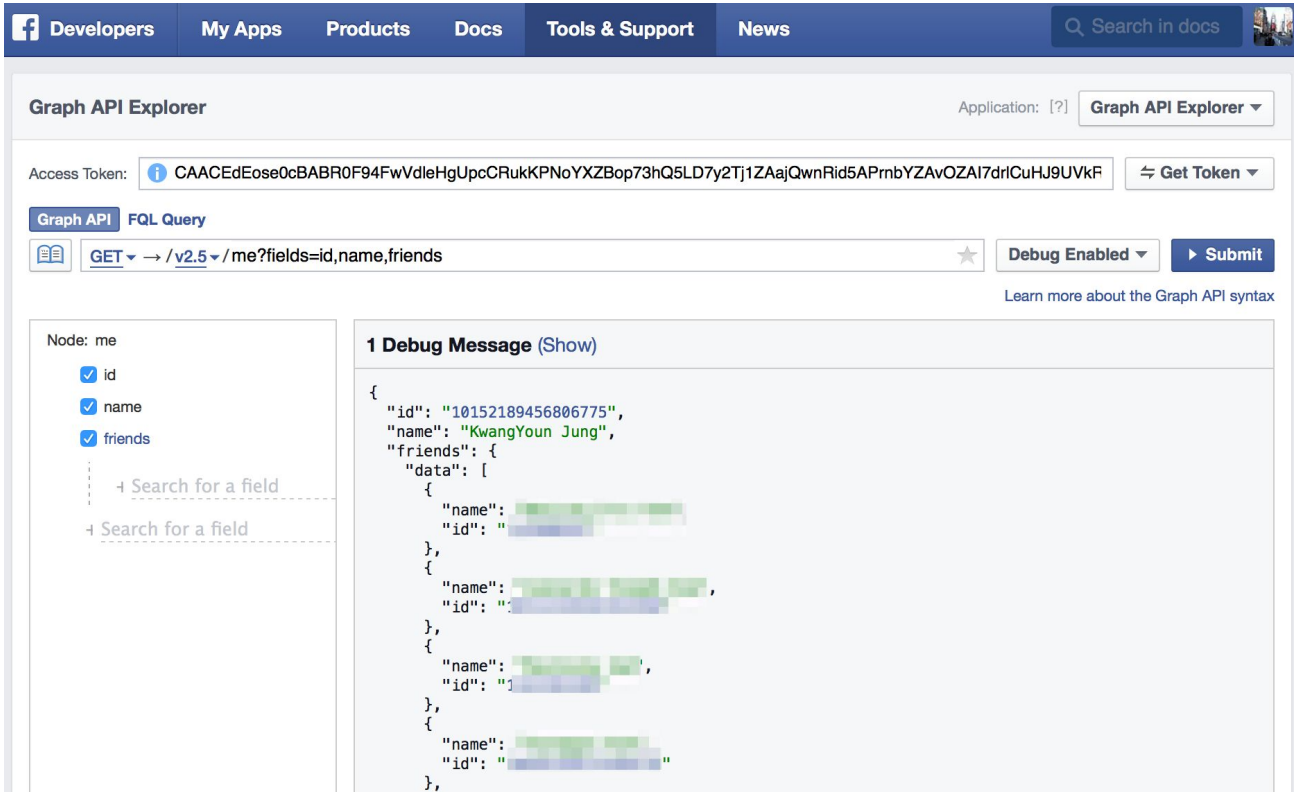
Create	HTTP PUT
Read	HTTP GET
Update	HTTP POST
Delete	HTTP DELETE

JSON(JavaScript Object Notation)

Javascript의 객체 표기법을 그대로 사용	
{ .. }	- 중괄호 안의 key와 value 쌍으로 이루어진 값 - 중첩 허용(객체 안의 객체 가능)

예	{
	“id”: “123456789”,
	”name”: “KwangYoun Jung”
	}

REST APIs 예제 - Facebook Social Graph API

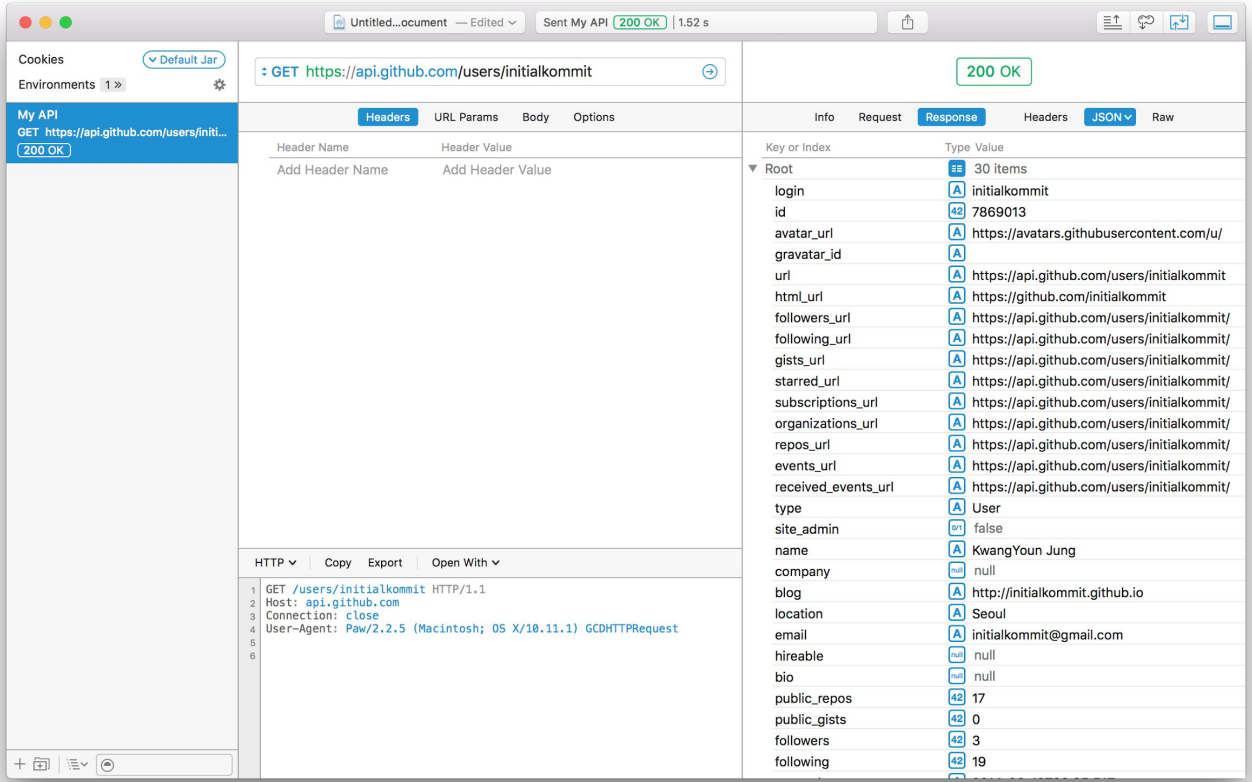


https://developers.facebook.com/tools/explorer?method=GET&path=me%3Ffields%3Did%2Cname%2Cfriends&version=v2.5

REST 개발 도구

- Django
  - Packages
    - [Django REST Framework](http://www.django-rest-framework.org/)
    - [Django Tastypie](http://tastypieapi.org/)
- 데이터 전송 테스트 도구
  - curl
    - Command line tool
  - Postman
    - An extension of Chrome browser
  - Paw for mac only

REST 개발 도구 - PAW

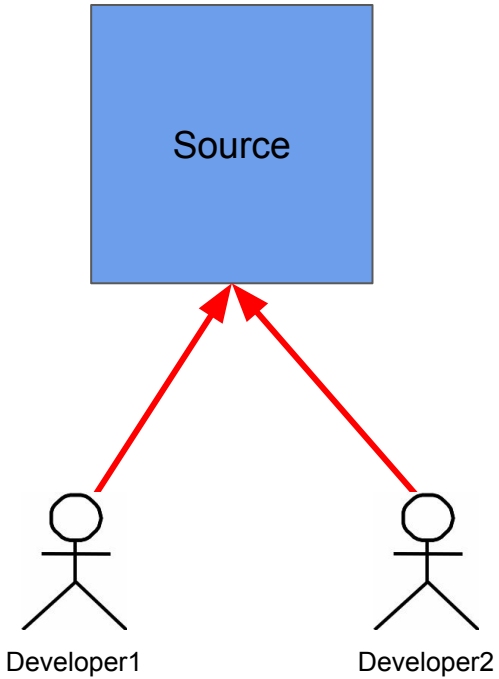


8. 산 넘어 산

SCM & Deployment

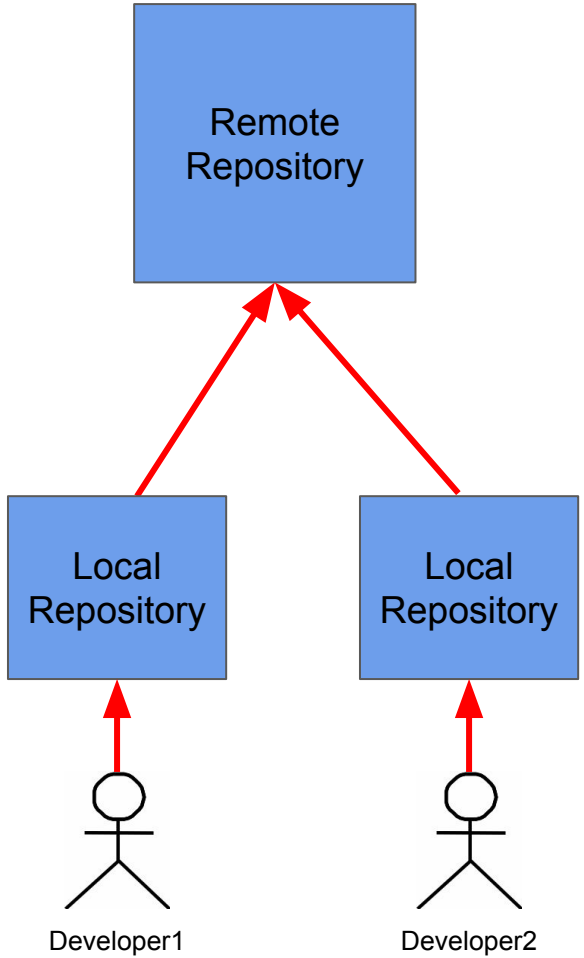
SCM

- Source Control Management
- 소스이력관리
- **중앙집중식**
  - Subversion(SVN)
    - TortoiseSVN
    - etc
- **분산처리식**
  - GIT
    - github.com
    - bitbucket.org
    - gitlab.com
    - etc
  - Mercurial
    - bitbucket.org



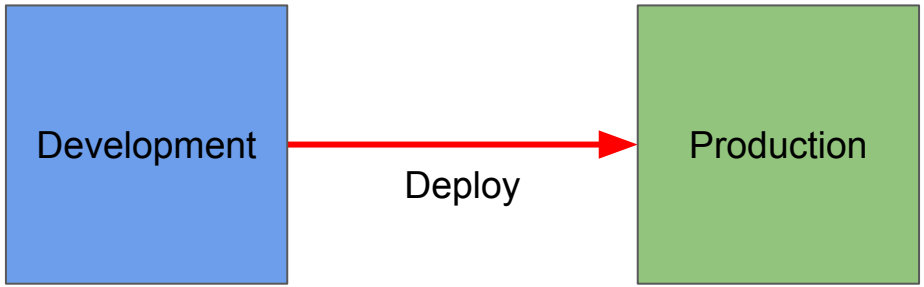
SCM

- Source Control Management
- 소스이력관리
- **중앙집중식**
  - SVN
    - TortoiseSVN
    - etc
- **분산처리식**
  - GIT
    - github.com
    - bitbucket.org
    - gitlab.com
    - etc



Deployment

- 서비스를 만들었다고 끝이 아니다.
- 이것을 실제 서비스로 작동하기 위해 소스를 웹서버에 올려야한다.
- 그것을 Deployment (배포) 라고 한다.

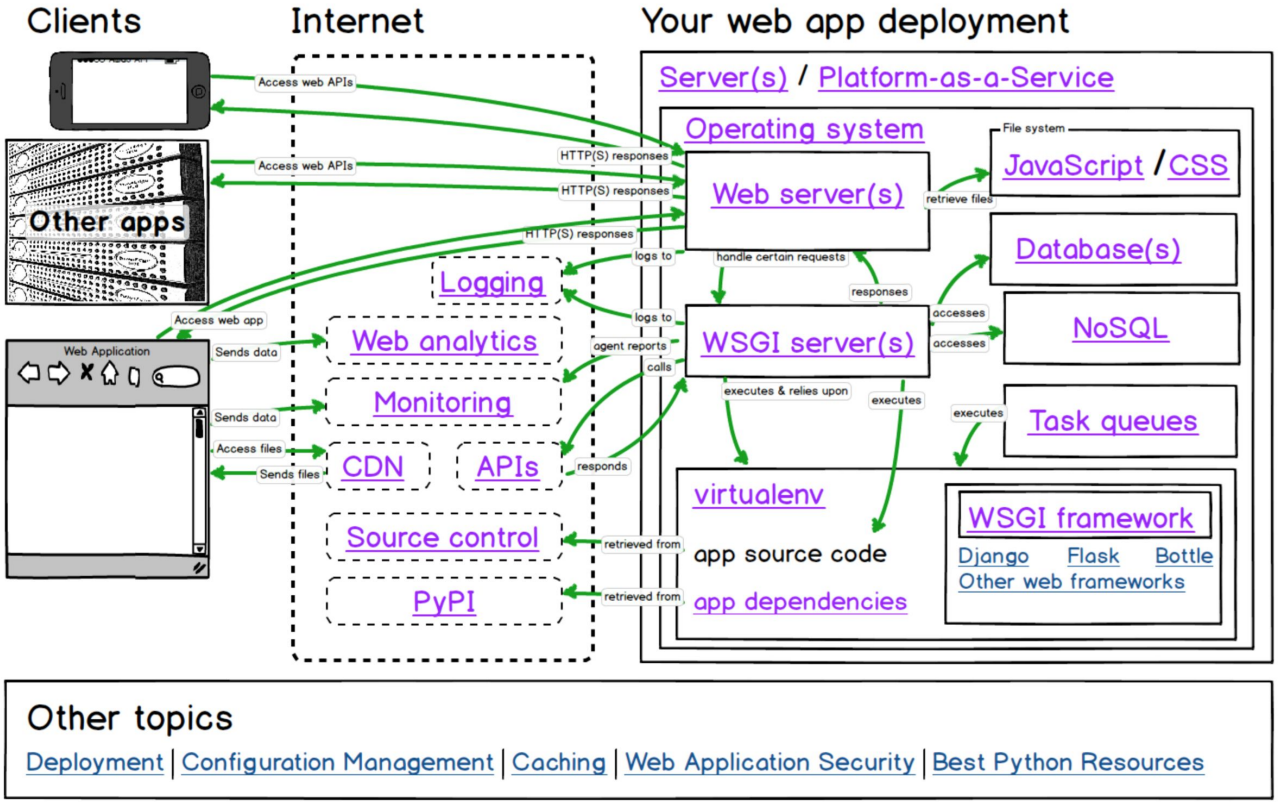


- 해외
  - AWS(JPN)
  - Digitalocean(SGP)
  - vultr(JPN)
  - Heroku(US)
  - pythonanywhere
- 우리나라
  - 가비아 gCloud
  - 다날 클라우드
  - 등

development, staging, production 환경을 최대한 비슷하게 유지

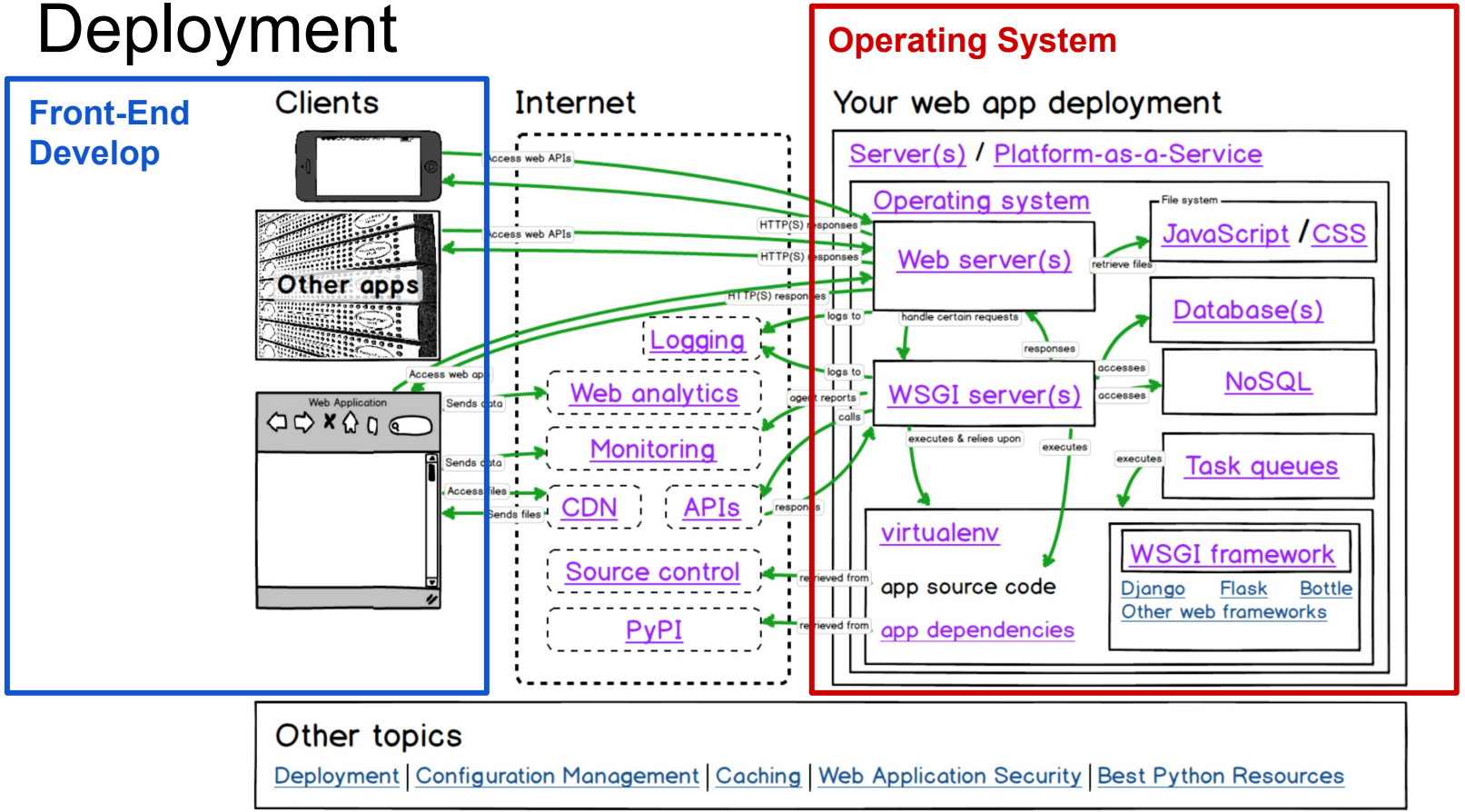
- http://12factor.net/ -

Deployment



http://www.fullstackpython.com/deployment.html

Deployment



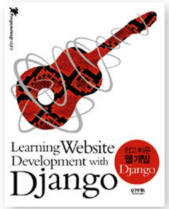
http://www.fullstackpython.com/deployment.html

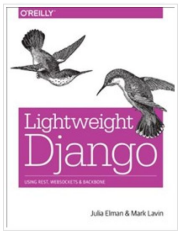
9. 로드맵

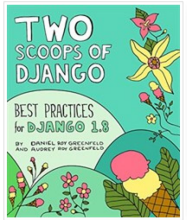
학습 로드맵 - 고지를 향하여

Roadmap

Basic	Intermediate	Advanced







[공식 홈페이지 튜토리얼]
(https://docs.djangoproject.com/en/1.8/intro/)

[차경묵님 입문자용 강좌]
(http://blog.hannal.com/category/start-with-django-lectures/)

[Try Django 1.8 Tutorial - 총 42강]
(https://www.youtube.com/watch?v=KsLHt3D_jsE)

[차경묵님 중급자용 강좌]
(http://blog.hannal.com/category/start-with-django-webframework/)

[TaskBuster Django Tutorial]
(http://www.marinamele.com/taskbuster-django-tutorial)

[Tango with Django]
(http://www.tangowithdjango.com)

[Django Book]
(http://www.djangobook.com)

기타

[Real Python]
(http://www.realpython.com)

[Full Stack Python]
(http://www.fullstackpython.com)

[Django Girls]
(http://www.djangogirls.org/seoul)

[Django Study]
(https://github.com/initialkommit/lightweight-django)

오픈소스에 참여하기

- [Django](https://github.com/django/django/): Django Web Framework
- [Wagtail](https://github.com/torchbox/wagtail/): Django CMS
- [Cactus](https://github.com/koenbok/Cactus/): Static site generator
- [Spirit](https://github.com/nitely/Spirit/): Forum

10. 옆 길

Django CMS - Wagtail 소개

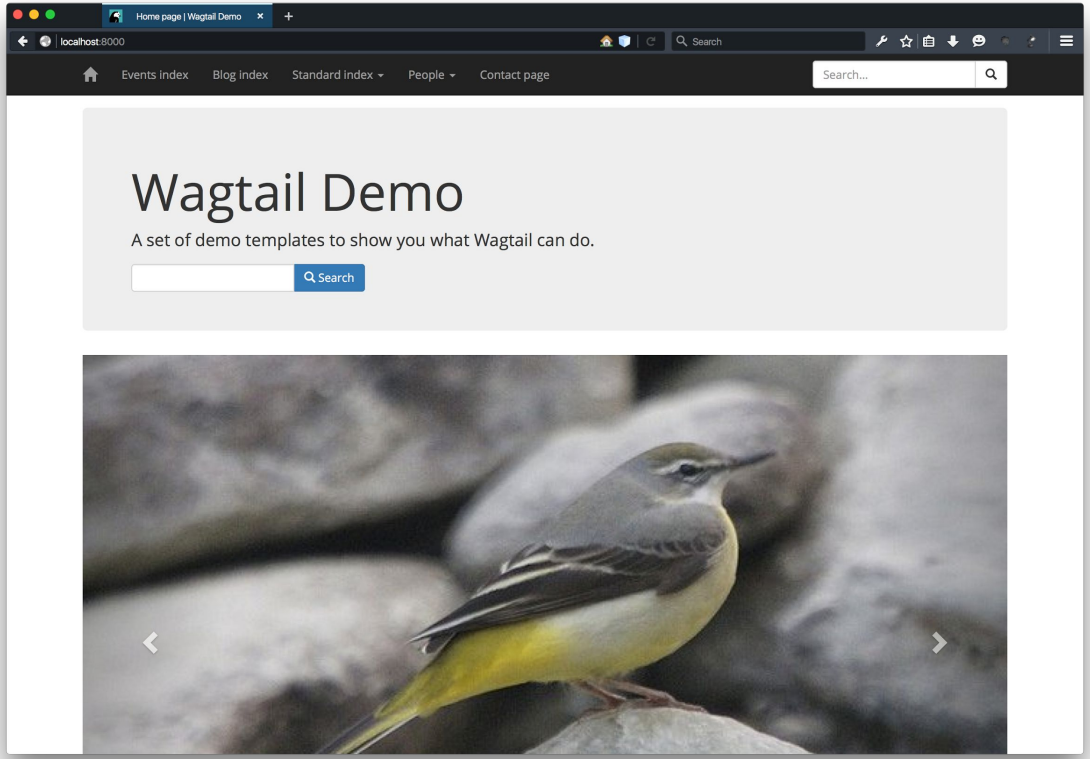
djangopackages.com

PACKAGE	DJANGO CMS	WAGTAIL CMS	MEZZANINE	FEINCMS	DJANGO-FIBER	ELLA
Description	The easy-to-use and developer-friendly CMS	A Django content management system focused on flexibility and user experience	CMS framework for Django	A Django-based CMS with a focus on extensibility and concise code	Django Fiber - a simple, user-friendly CMS for all your Django projects	Ella is a CMS based on Python web framework Django with a main focus on high-traffic news websites and Internet ...
Category	Framework	Framework	Framework	Framework	App	Framework
# Using This	135△	29△	75△	48△	19△	7△
Python 3?	✔	✔	✔	✔	✖	✖
Development Status	Production/Stable	Production/Stable	Production/Stable	Production/Stable	Production/Stable	Beta
Last updated	Nov. 14, 2015, 2:33 p.m.	Nov. 12, 2015, 11:55 a.m.	Oct. 29, 2015, 11:41 p.m.	Sept. 8, 2015, 9:31 a.m.	Oct. 27, 2015, 5:02 a.m.	April 1, 2014, 8:45 p.m.
Version	3.1.3	1.2rc1	4.0.1	2.0a11	1.3.1	3.0.13
Repo	Github	Github	Github	Github	Github	Github
Commits						
Stars	3743	3063	2452	629	518	259
Repo Forks	1451	489	962	205	86	63

wagtail

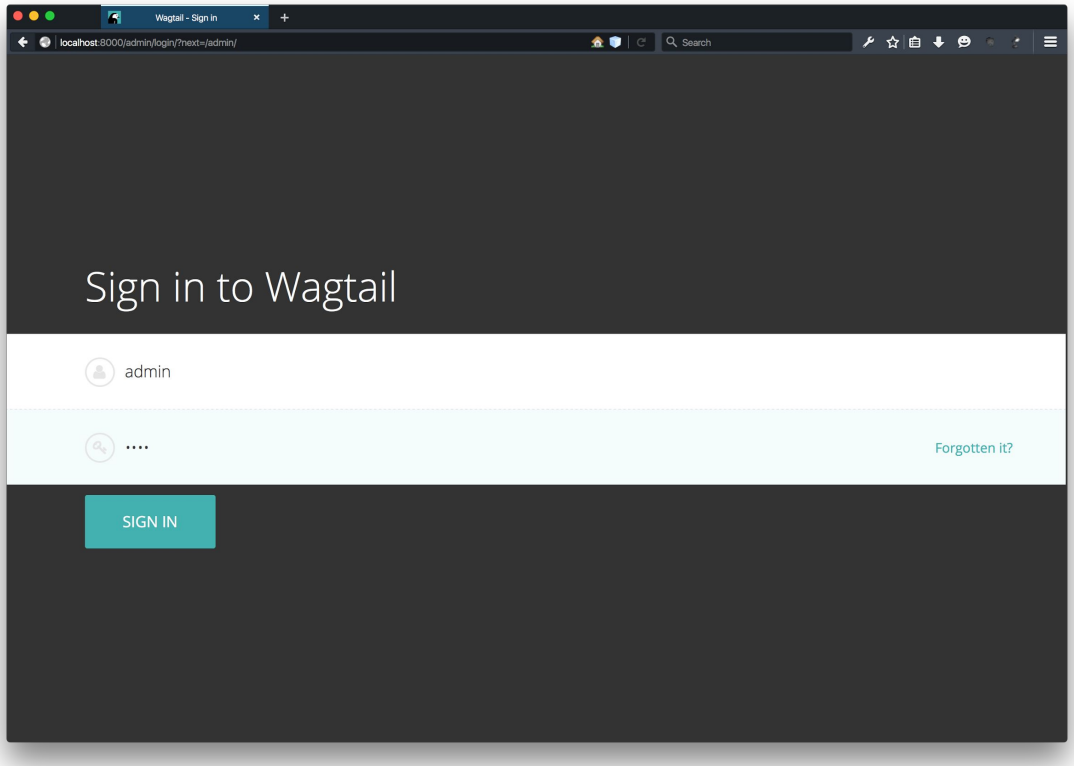
공식 홈페이지	wagtail.io
제작사 정보	torchbox.com
	UK digital agency
	Web Design, UX, Drupal, Wagtail, Django
용도	Django로 만들어진 CMS
오픈 소스	https://github.com/torchbox/wagtail
wagtail 뜻	할미새
version	1.2 (11/13/2015)
support	Python 3.5, Django 1.8.6, Django REST Framework Browsable API
Features	이미지 파일 관리, 첨부파일 관리, 깔끔한 인터페이스, 강력한 에디터 등

wagtail demo



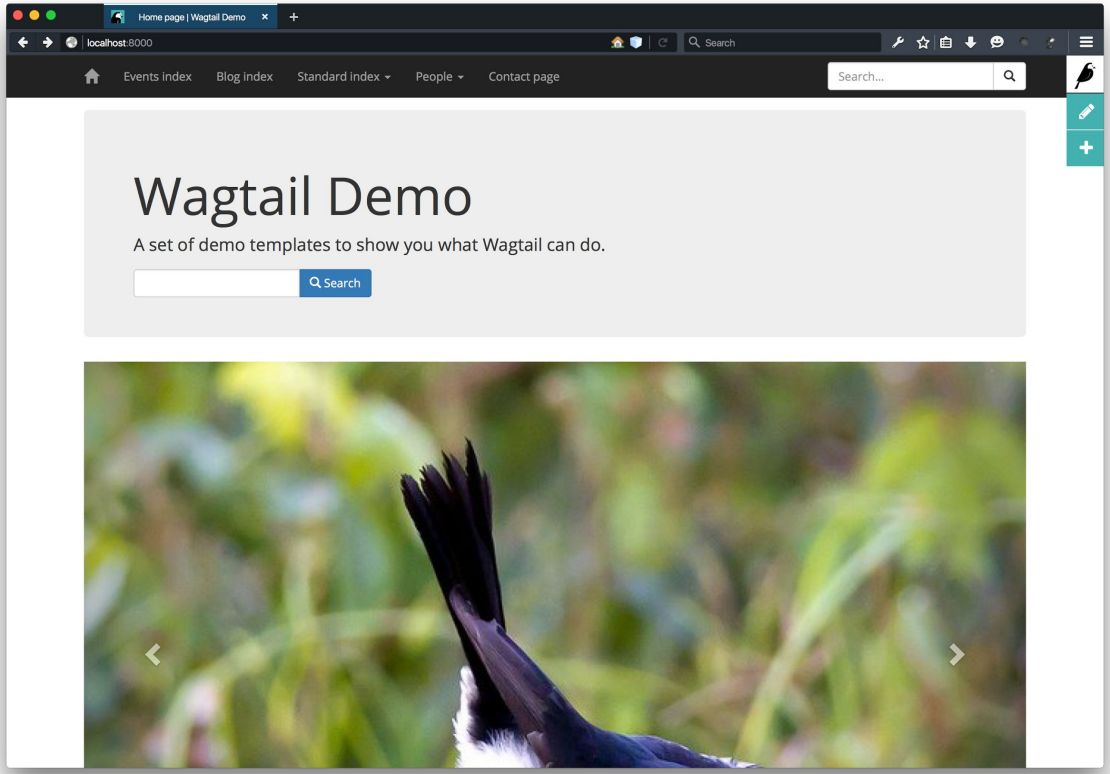
https://github.com/torchbox/wagtaildemo

wagtail demo



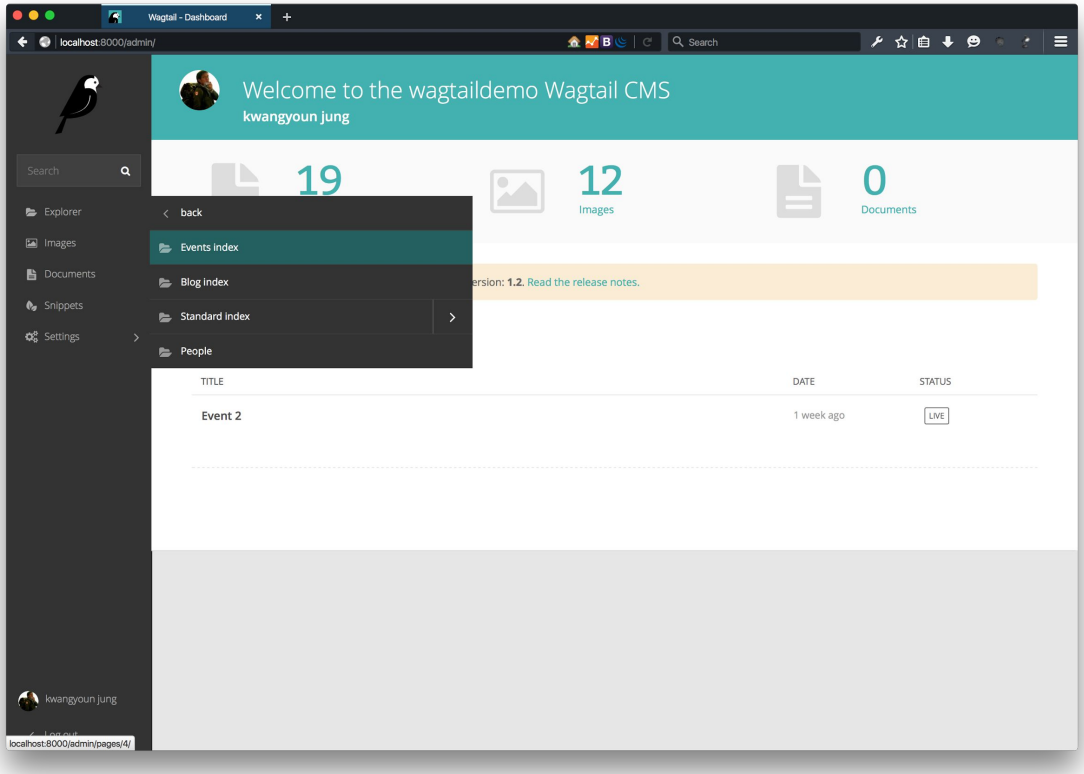
https://github.com/torchbox/wagtaildemo

wagtail demo



https://github.com/torchbox/wagtaildemo

wagtail demo



https://github.com/torchbox/wagtaildemo

wagtail demo 시연

https://github.com/torchbox/wagtaildemo

Feel free to contact me

if you have any question.

정광윤/Python Developer
initialkommit@gmail.com