

# Django Framework

장고걸스 튜토리얼 참조하여 웹 어플리케이션 제작

양경석(gaeng4@gmail.com)

# 1. Introduction

Wednesday, August 10, 2016 9:59 AM

\*\*장고걸스 (<https://djangogirlsseoul.gitbooks.io>) 참조

장고(쟁고: Django)?

전 세계에서 가장 인기 있는 언어인 파이썬으로 작성된 웹 프레임워크로 다양하고 복잡한 기능을 지원

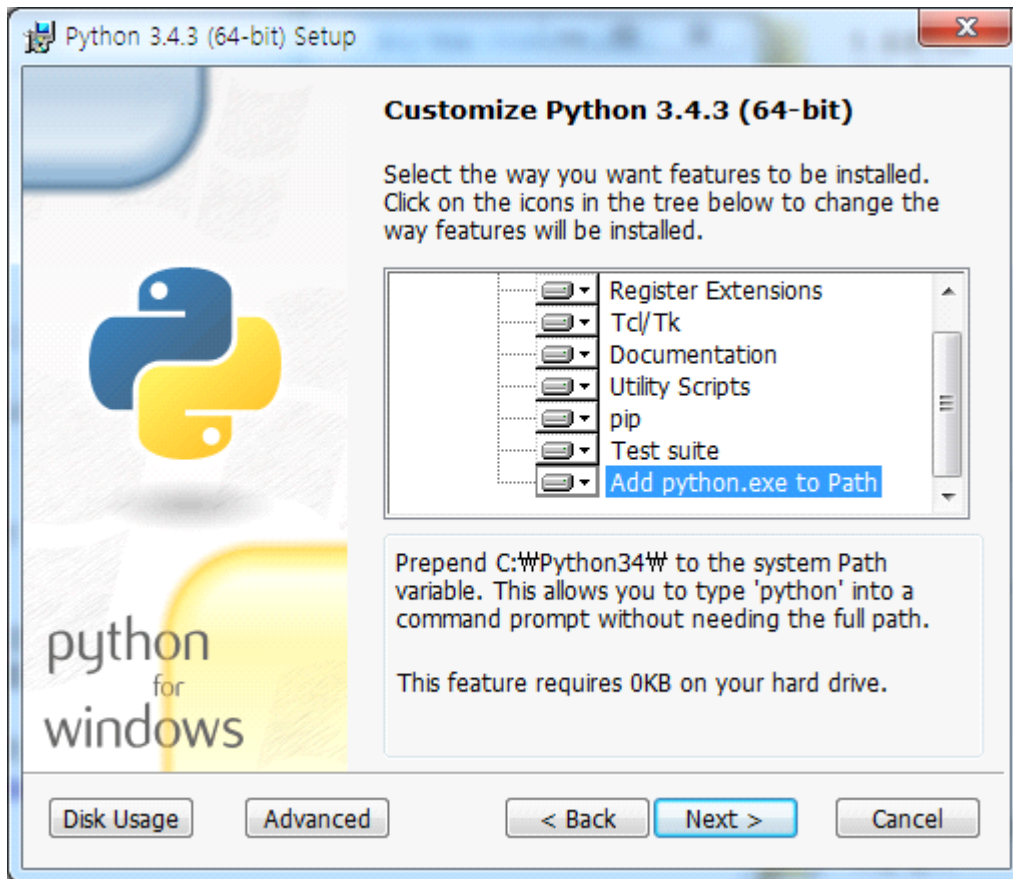
<설치하기> - 윈도우즈 기준으로 설치

파이썬, 장고, 코드 에디터를 설치

i) python 설치(python 3.4.3)

<https://www.python.org/downloads/> -> windows x86-64 msi installer 다운 후 실행

C:\Python34\ 에 설치



add python.exe to path 설정(Entire feature will be installed on local hard drive)

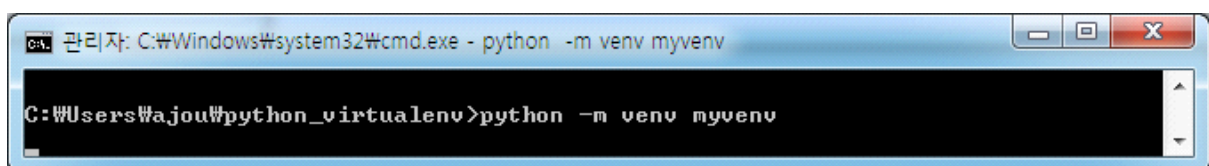
\*virtualenv(가상환경)설정하고 Django 설치

-개발 환경을 깔끔하게 관리하는 데 도움이되는 도구

-프로젝트 기초 전부를 Python/Django와 분리해줌(웹사이트가 변경되어도 개발중인 것에 영향을 미치지 않음)

virtualenv 생성할 폴더 생성(c:\users\ajou\python\_virtualenv)

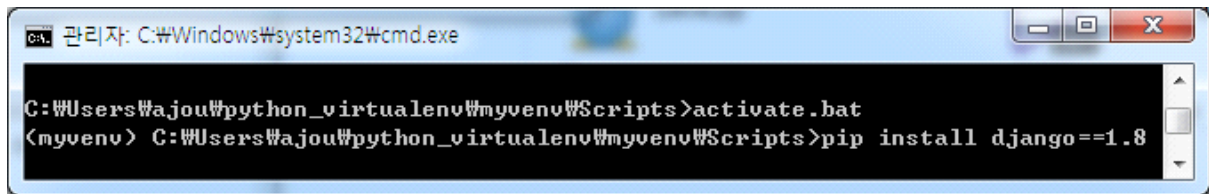
python -m venv 설치할 가상환경의 임의의 이름(자주 입력해야 하니 짧은게 좋음)



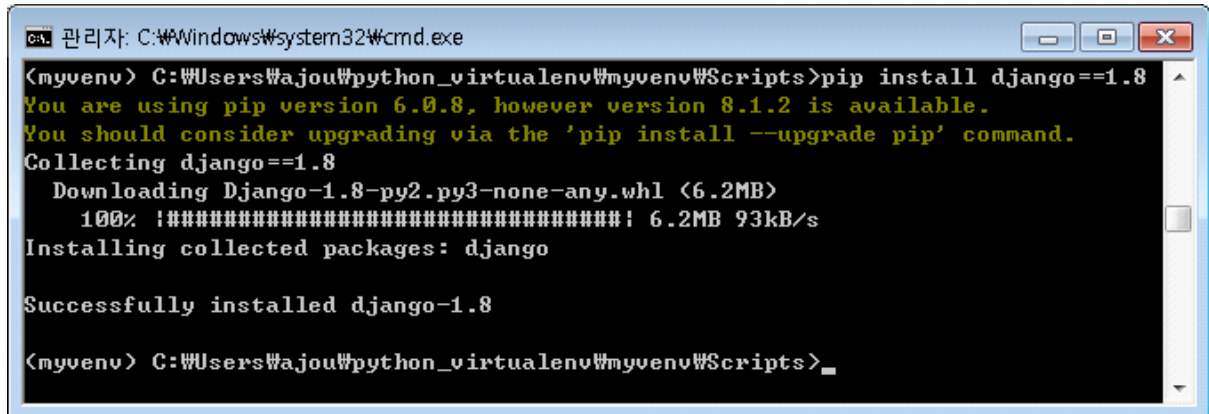
가상환경 실행

myvenv\scripts\activate.bat

pip install 실행하여 django 설치



```
관리자: C:\Windows\system32\cmd.exe
C:\Users\Wajou\python_virtualenv\myenv\Scripts>activate.bat
(myenv) C:\Users\Wajou\python_virtualenv\myenv\Scripts>pip install django==1.8
```



```
관리자: C:\Windows\system32\cmd.exe
(myenv) C:\Users\Wajou\python_virtualenv\myenv\Scripts>pip install django==1.8
You are using pip version 6.0.8, however version 8.1.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Collecting django==1.8
  Downloading Django-1.8-py2.py3-none-any.whl (6.2MB)
    100% |#####| 6.2MB 93kB/s
Installing collected packages: django
Successfully installed django-1.8
(myenv) C:\Users\Wajou\python_virtualenv\myenv\Scripts>_
```

설치 완료

<텍스트 에디터 설치>

Gedit

모든 운영체제에서 사용 가능. 무료 오픈소스

<https://wiki.gnome.org/Apps/Gedit#Download>

Sublime text2

가장 널리 알려진 프로그램. 무료

모든 운영체제에서 사용 가능

<http://www.sublimetext.com/2>

Atom

GitHub에서 만든 에디터. 윈도우/맥/리눅스에서 사용 가능

무료로 제공되는 오픈소스

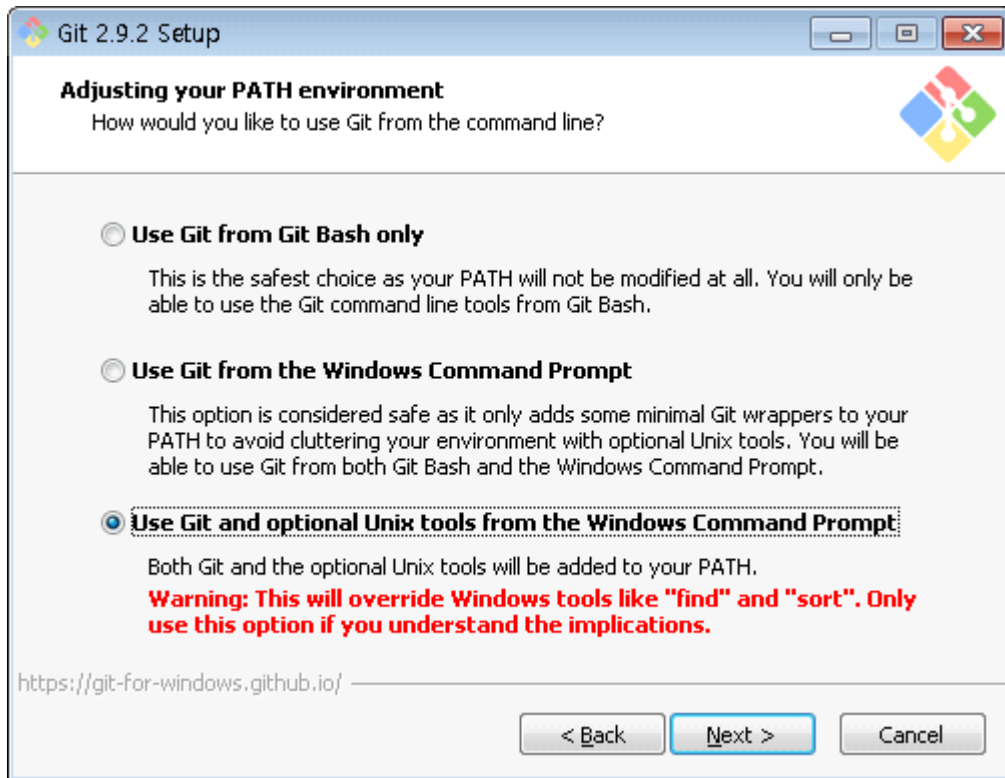
<https://atom.io/>

PyCharm 대부분의 파이썬 프로그래머가 사용하는 IDE(Integrated Development Environments)

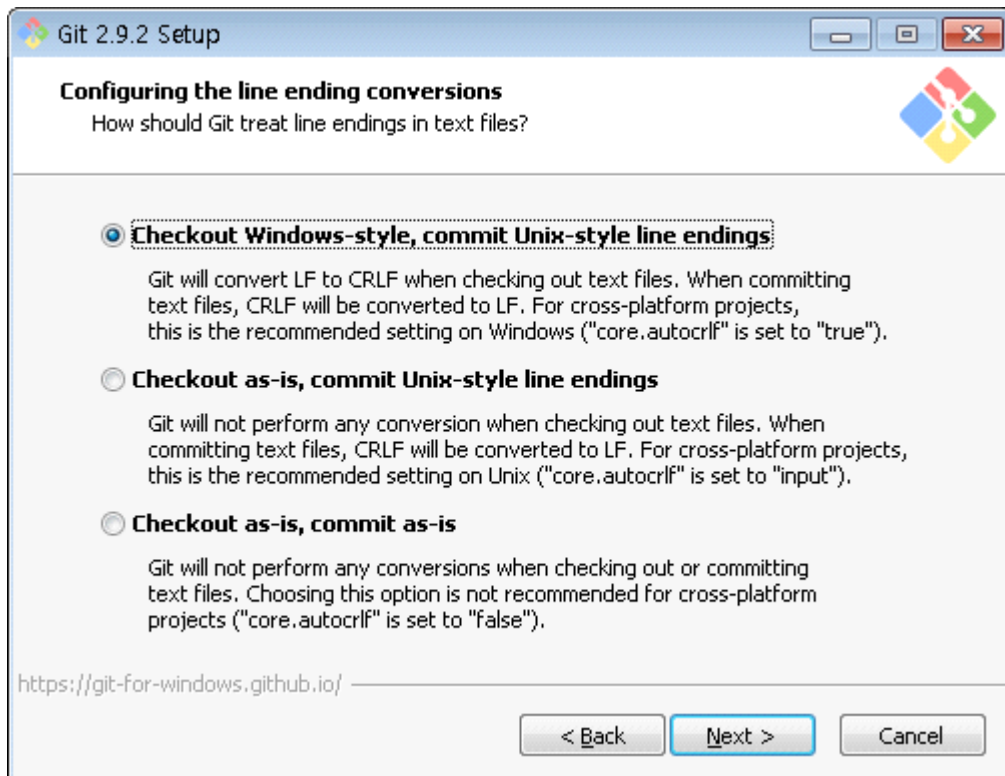
<https://www.jetbrains.com/pycharm/>

<git 설치>

<https://git-scm.com/download/win>



세번째 옵션으로 선택



첫번째 옵션으로 선택

github.com

<https://www.pythonanywhere.com>

(beginner account 가입: 아이디 akagaeng)

akagaeng.pythonanywhere.com

<command line>

python 명령어 command line에서 입력 가능

why command line?

easy(maybe) and powerful

command line = {prompt, console, terminal, shell, bash} 등 여러가지 이름으로 불림

but 본질은 OS, application에 명령을 내리기 위한 text interface

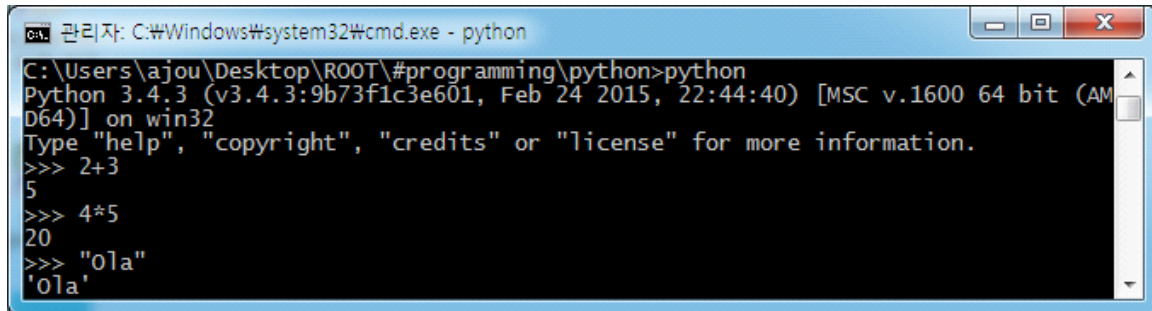
computer's Layers

user // applications // OS // hardware

## 2. 간단한 python 문법

2016년 8월 11일 목요일    오후 12:12

python 실행



```
C:\Users\ajou\Desktop\ROOT\#programming\python>python
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> 4*5
20
>>> "Ola"
'Ola'
>>> 'Ola'
'Ola'
```

\*파이썬 종료하기

-exit() 입력

-CTRL+Z (윈도우즈)

-CTRL+D (맥/리눅스)

\*사칙연산

그냥 콘솔에 입력 후 엔터

\* 문자열

큰따옴표나 작은 따옴표 안에 감싸서 넣음

'나 '를 입력하기 위해서는 \를 앞에 붙이거나 따옴표로 감쌘

-관련 함수

입력	출력	기능	속성
len("Ola")	3	글자수 반환	함수 (문자/숫자 등 모든 객체에 사용 가능)
"ola".upper()	'OLA'	대문자로 변환	메서드(문자열에만 쓸 수 있는 함수)

str() 문자열로 변환

int() 정수로 변환

텍스트->숫자 변환은 불가

print(변수 / 정수) 출력

\*변수

name = "Ola"

\*\*리스트

서로 다른 객체들을 일렬로 나열한 것

```
>>> []
```

```
[]
```

```
>>> lottery = [6,1,2,3,4,5]
```

```
>>> lottery
```

```
[6, 1, 2, 3, 4, 5]
```

```
>>> len(lottery)
```

```
6
```

```
>>> lottery.sort()
```

```
[1, 2, 3, 4, 5, 6] //이 때 lottery에 정렬된 상태가 저장됨
>>>lottery.reverse()
[6, 5, 4, 3, 2, 1] //이 때 lottery에 정렬된 상태가 저장됨
>>>lottery.append(200) //리스트의 마지막에 추가됨
>>>lottery
[6, 5, 4, 3, 2, 1, 200]
```

\*인덱스 파이썬도 0부터 인덱싱

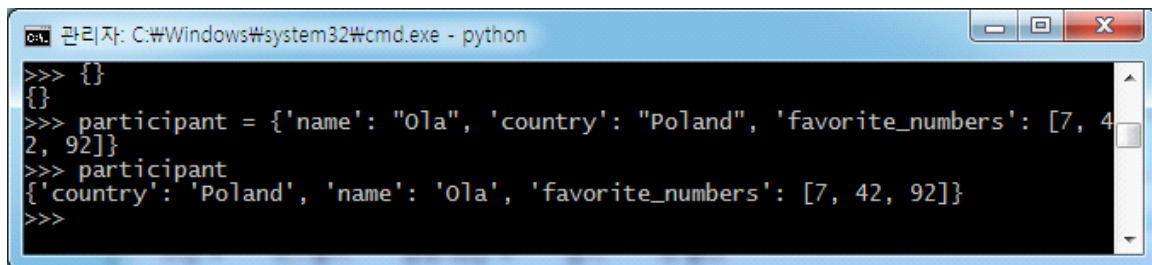
```
>>>print(lottery[0])
6
>>>lottery[0]
6
```

\*\*딕셔너리(Dictionary)

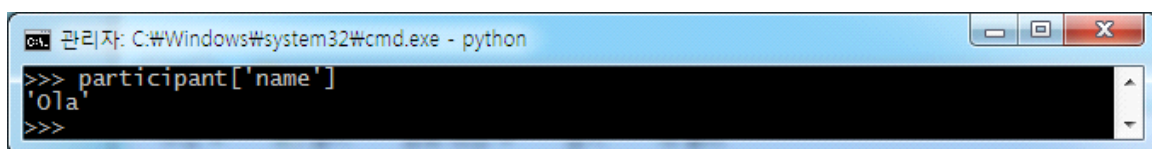
리스트와 유사하나, 인덱스가 아닌 키(key)로 값을 찾음

```
>>>{}
{}

```



```
C:\> 관리자: C:\Windows\system32\cmd.exe - python
>>> {}
{}
>>> participant = {'name': "Ola", 'country': "Poland", 'favorite_numbers': [7, 42, 92]}
>>> participant
{'country': 'Poland', 'name': 'Ola', 'favorite_numbers': [7, 42, 92]}
>>>
```



```
C:\> 관리자: C:\Windows\system32\cmd.exe - python
>>> participant['name']
'Ola'
>>>
```

리스트에서 lottery[0] 처럼 인덱스(0)로 값을 찾았듯이

딕셔너리에서는 participant['name']처럼 키('name')로 찾음

(만약 participant{3:7} 과 같이 정의했다면?

```
>>>participant[3]
7
```

또한, 키로 값을 찾기 때문에 처음 정의할때와 출력할 때가 순서가 달라질 수 있음

(name, country, favorite\_numbers 순으로 정의하였으나 country, name, favorite\_numbers 순으로 출력됨)

리스트	아이템 정렬이 필요할 때 사용
딕셔너리	키(key)와 값(value)이 서로 연관되어 있을 때 사용

```
>>>len(participant)
4
키-값 쌍의 개수를 반환
```

딕셔너리 삭제

```
>>>del participant('name')
키를 삭제하면 키&값 삭제됨
```



## 비교

참이면 True, 거짓이면 False (TRUE x true x 대소문자 확실히 구분할 것!)

`a > b`, `a == b`, `a != b`, `a >= b`, `a and b`, `a or b`

## 코드 저장하기

>>>에서 실행시키는 것은 "인터프리터"에서 실행. 저장이 되지 않음

PyCharm에서 파이썬 파일 생성 후 파일 저장

\*student인 경우 학생용 license 무료로 받을 수 있음

## 문법

### if문

있는 것	:
없는 것	{, ;

```
name = 'Ola'
if name == 'Ola':
    print('Hey Ola!')
elif name == 'Sonja': //else if가 아니라 elif인 것에 유의!
    print('Hey Sonja!')
else:
    print('Hey anonymous!')
```

### 함수 만들기

위에서부터 읽어오므로 함수를 위에다 정의해야 함

void형과 같은 함수

```
def hi():
    name = "yang"
    print("my name is " + name)

hi()
```

입력 값이 있는 함수

```
def hi(name):
    print("my name is " + name)

hi("gaeng")
```

반환은 return xx처럼 하면 됨(반환이 없을 수도 있음)

### 반복문

for 변수 in 범위/리스트

```
for i in range(1,6):  
    print(i)
```

1  
2  
3  
4  
5

```
girls = ['Rachel', 'Monica', 'Phoebe', 'Ola', 'You']  
  
for name in girls:  
    print(name)
```

Rachel  
Monica  
Phoebe  
Ola  
You

### 3. Django?

2016년 8월 11일 목요일    오후 4:59

Django(쟁고/장고)?

Python으로 만들어진 무료 오픈소스 웹 어플리케이션 프레임워크(Web Application Framework)

쉽고 빠르게 웹사이트를 개발할 수 있도록 도와주는 구성요소로 이뤄진 프레임워크

반복적으로 필요한 구성요소(ex: 회원가입, 로그인 ...)를 바로 사용할 수 있도록 갖추어놓은 것  
따라서 매번 새로 만들어야 할 필요가 사라지고, 웹사이트 개발시 뒤따르는 간접비용 부담 줄여줌

가상환경 활성화

```
C:\Users\ajou\python_virtualenv>myvenv\scripts\activate.bat
```

(myvenv) django-admin startproject mysite .

// 마지막에 점(.) 빼먹지 말 것! mysite 뒤에 한칸 띄어야 함!

디렉터리 구조

~python\_virtualenv

-manage.py

-myenv

-mysite

-settings.py

-urls.py

-wsgi.py

\_\_init\_\_.py

manage.py : 사이트 관리를 도와주는 역할. 다른 설치 없이 웹서버를 실행할 수 있음

settings.py : 웹사이트 설정이 있는 파일

urls.py : urlresolver가 사용하는 패턴 목록을 포함

settings.py 수정

```
TIME_ZONE = 'Asia/Seoul'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.8/howto/static-files/

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

TIME\_ZONE 수정

정적 경로 추가

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

## 데이터베이스 설정

sqlite3를 사용하도록 설정(default)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

\*콘솔창에서 다음과 같은 코드 입력

(myvenv) ~python\_virtualenv> python manage.py migrate

다음과 같이 실행됨

```
(myvenv) C:\Users\ajou\python_virtualenv>python manage.py migrate
Operations to perform:
  Synchronize unmigrated apps: messages, staticfiles
  Apply all migrations: admin, contenttypes, auth, sessions
Synchronizing apps without migrations:
  Creating tables...
  Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... 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 sessions.0001_initial... OK
(myvenv) C:\Users\ajou\python_virtualenv>
```

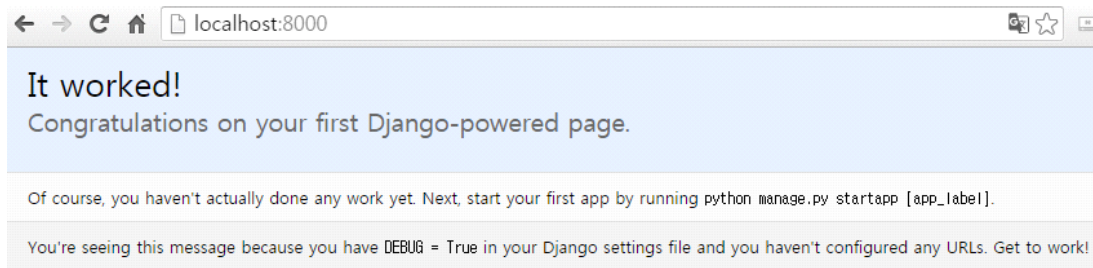
\*웹서버 실행

(myvenv) ~python\_virtualenv> python manage.py runserver

```
(myvenv) C:\Users\ajou\python_virtualenv>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
August 11, 2016 - 18:09:30
Django version 1.8, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

브라우저에서 접속한 화면



접속하면 다음과 같이 화면에 표시됨

```
[11/Aug/2016 18:10:02] "GET / HTTP/1.1" 200 1767  
[11/Aug/2016 18:10:02] "GET /favicon.ico HTTP/1.1" 404 1936
```

## 4. Django Model

2016년 8월 16일 화요일    오후 2:21

블로그 내 모든 포스트를 저장하는 부분 제작

객체지향 프로그래밍(Object Oriented Programming)

프로그램이 어떻게 작동해야 하는지 하나하나 지시하지 않고, 모델을 만들어 그 모델이 어떤 역할을 가지고 어떻게 행동해야 하는지 정의하여 서로 알아서 상호작용할 수 있도록 만드는 것

객체(Object) ?

속성과 행동(행위)를 모아놓은 것

속성=> 객체 속성(properties)

행위=> 메서드(method)

ex) 자동차 클래스(소유주, 속도, 속도up(), 속도 down(), ...) : 개념

클래스의 instance = : 실제(객체와 동일한 말로 써도 무방. 클래스가 실체화된 것)

자동차 객체 1(소유주: aaa, 속도: 100km/h, 속도up(), 속도 down(), ... ) : 실체

자동차 객체 2(소유주: bbb, 속도: 50km/h, 속도up(), 속도 down(), ... ) : 실체

ex)

객체: 블로그 글

Post(게시글)

[property] title(제목)

[property] text(내용)

[property] author(글쓴이)

[property] created\_date(작성일)

[property] published\_date(게시일)

[method] publish(출판)

장고 모델

장고 안의 모델은 객체의 특별한 종류

이 모델 저장시 그 내용이 DB에 저장됨

어플리케이션 제작하기

```
C:\Users\ajou\python_virtualenv>myvenv\Scripts\activate.bat
(myvenv) C:\Users\ajou\python_virtualenv>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 76A0-8C0C

C:\Users\ajou\python_virtualenv 디렉터리

2016-08-16 오후 02:48 <DIR> .
2016-08-16 오후 02:48 <DIR> ..
2016-08-16 오후 02:48 <DIR> blog
2016-08-11 오후 06:06 36,864 db.sqlite3
2016-08-11 오후 06:01 249 manage.py
2016-08-11 오후 06:06 <DIR> mysite
2016-08-10 오후 06:25 <DIR> myvenv
2개 파일 37,113 바이트
5개 디렉터리 577,789,071,360 바이트 남음

(myvenv) C:\Users\ajou\python_virtualenv>python manage.py startapp blog
```

blog라는 디렉터리 생성됨

내부 구조는 다음과 같음

```
(myvenv) C:\Users\ajou\python_virtualenv>cd blog
(myvenv) C:\Users\ajou\python_virtualenv\blog>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 76A0-8C0C

C:\Users\ajou\python_virtualenv\blog 디렉터리

2016-08-17 오전 10:14 <DIR> .
2016-08-17 오전 10:14 <DIR> ..
2016-08-16 오후 02:48 63 admin.py
2016-08-16 오후 02:48 <DIR> migrations
2016-08-16 오후 02:48 57 models.py
2016-08-16 오후 02:48 60 tests.py
2016-08-16 오후 02:48 63 views.py
2016-08-16 오후 02:48 0 __init__.py
2016-08-17 오전 10:14 <DIR> __pycache__
5개 파일 243 바이트
4개 디렉터리 577,789,063,168 바이트 남음
```

setting 변경

python\_virtualenv/mysite/settings.py 편집

INSTALLED\_APPS에 'blog' 추가

```
# Application definition

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
)
```

python\_virtualenv/blog/models.py 편집(다음과 같이 작성)

```
from django.db import models //다른 파일에 있는 것을 추가하라는 의미. copy&paste해야 하는 작업을 from~으로 불러옴
from django.utils import timezone

class Post(models.Model): //모델(객체) 정의. class: 객체 정의시 사용.
    //Post: 객체명(첫글자 무조건 대문자로).
    //models: 장고 모델임을 의미함.
    //이 코드 때문에 장고는 Post가 DB에 저장되어야 한다고 알게 됨

    author = models.ForeignKey('auth.User') //ForeignKey: 다른 모델에 대한 링크(외래키)
    title = models.CharField(max_length=200) //CharField: 글자수 제한 있는 텍스트
    text = models.TextField() //TextField: 글자 수 제한 없는 텍스트
    created_date = models.DateTimeField( //날짜와 시간
        default=timezone.now)
    published_date = models.DateTimeField( //날짜와 시간
        blank=True, null=True)

    def publish(self): //def: 함수/메서드 정의, publish: 메서드명
        self.published_date = timezone.now()
        self.save()

    def __str__(self):
        return self.title //return: 메서드가 반환하는 것
```

c.f. \_\_ : dunder ( double underscore의 줄임말)

데이터베이스에 모델을 위한 테이블 작성

DB에 'blog'라는 모델 추가

마이그레이션 파일 준비

`python manage.py makemigrations blog`

```
(myvenv) C:\Users\ajou\python_virtualenv>python manage.py makemigrations blog
Migrations for 'blog':
  0001_initial.py:
    - Create model Post
```

글 모델을 DB에 저장

`python manage.py migrate blog`

```
(myvenv) C:\Users\ajou\python_virtualenv>python manage.py migrate blog
Operations to perform:
  Apply all migrations: blog
Running migrations:
  Rendering model states... DONE
  Applying blog.0001_initial... OK
```

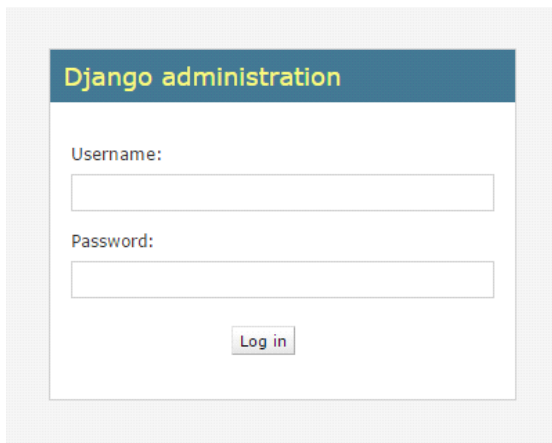
장고 관리자

/blog/admin.py 수정

```
from django.contrib import admin //앞에서 정의한 Post 모델 import
from .models import Post

admin.site.register(Post)
```

<http://localhost:8000/admin/> 입력



The image shows the Django administration login page. It has a blue header with the text "Django administration". Below the header, there are two input fields: "Username:" and "Password:". At the bottom, there is a "Log in" button.

위와 같은 페이지 볼 수 있음

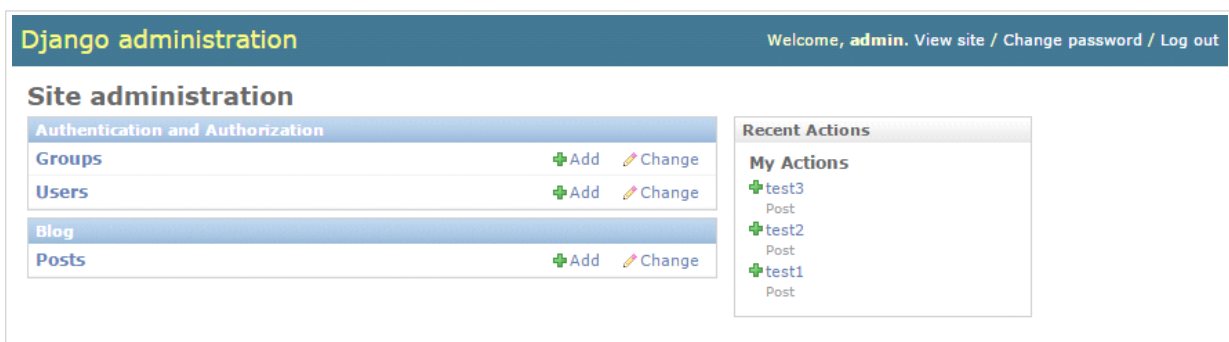
로그인을 위하여 슈퍼 사용자 생성

(myvenv) C:\Users\Wajou\python\_virtualenv> python manage.py createsuperuser

```
(myvenv) C:\Users\ajou\python_virtualenv>python manage.py createsuperuser
Username (leave blank to use 'ajou'): admin
Email address: akagaeng@gmail.com
Password:
Password (again):
Superuser created successfully.
```

장고 관리자 페이지에서 생성한 superuser로 로그인

그룹 관리, user 관리, post 관리 가능



The image shows the Django administration dashboard. The header is blue with the text "Django administration" and "Welcome, admin. View site / Change password / Log out". Below the header, there is a "Site administration" section. This section contains two main areas: "Authentication and Authorization" and "Blog". Under "Authentication and Authorization", there are links for "Groups" and "Users", each with "Add" and "Change" buttons. Under "Blog", there is a link for "Posts" with "Add" and "Change" buttons. To the right of the "Site administration" section, there is a "Recent Actions" section. This section contains a "My Actions" list with three entries: "test3", "test2", and "test1", each with a "Post" label.

시험 삼아 test1, test2, test3이라는 post작성한 화면



## 5. 배포(Deployment)!

2016년 8월 18일 목요일    오후 4:03

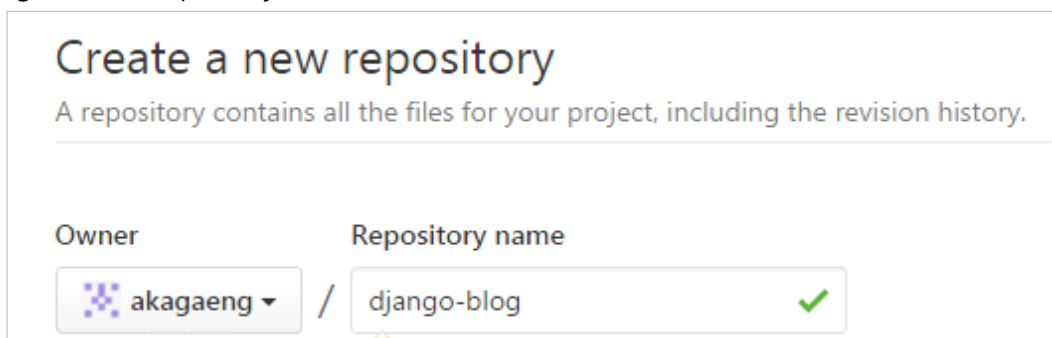
### 일단 배포(deploy)!

Local에서 작업 중인 내용을 호스팅 서비스에 업로드

GitHub(코드호스팅), PythonAnywhere(서버 호스팅)을 이용  
git-scm.com에서 설치

```
$ git init //최초 local git 저장소 생성시에만 작성
$git config --global user.name "akagaeng"
$git config --global user.email akagaeng@gmail.com //user등록
```

github에서 repository 만들기



clone with HTTPS로 주소 복사

콘솔창에서

```
git remote add origin 복사한 주소
git push -u origin master
```

username, password누르고 나면 push됨

디렉터리 중 변경점 추적하지 않을 폴더/파일은 .gitignore에 저장

.gitignore

```
*.pyc
__pycache__
myenv
db.sqlite3
.DS_Store
```

```
$ git status //변경된 것 있는지 확인
$ git add --all . //로컬 저장소에 파일 추가
$ git commit -m "My Django blog, 1st commit!" //로컬 저장소에 등록(커밋)
```

로컬 저장소와 원격 저장소 연결

```
$git remote add <저장소 별명> <복사한 주소>
```

```
(myenv) C:\Users\ajou\python_virtualenv>git remote add origin https://github.co
```

로컬 작업 내역을 원격 저장소에 등록(push)

`$git push <저장소 별명> <로컬 브랜치 이름> //` 원격 저장소에 등록(push)

\*처음에는 다음과 같이 실행

`$git push -u origin master //`이 때, `-u`는 `--set-upstream`으로써 remote source와 branch name입력을 생략할 수 있도록 함

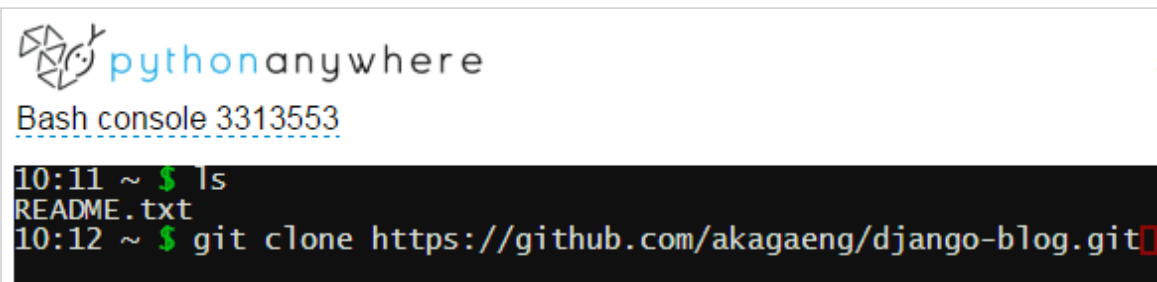
따라서 다음부터는 `$git push` 만 입력해도됨

## GitHub에서 PythonAnywhere로 코드 가져오기

<https://www.pythonanywhere.com> 접속

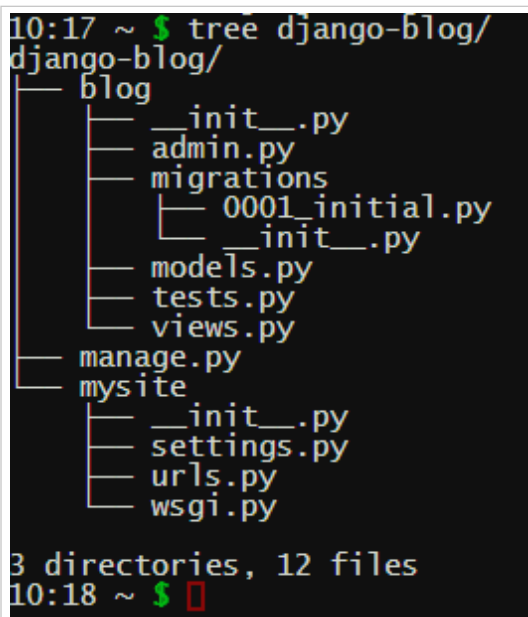
bash console 선택

github에 있는 코드를 clone해옴



```
pythonanywhere
Bash console 3313553
10:11 ~ $ ls
README.txt
10:12 ~ $ git clone https://github.com/akagaeng/django-blog.git
```

클론후 디렉터리 구조(tree 명령어로 확인)



```
10:17 ~ $ tree django-blog/
django-blog/
├── blog
│   ├── __init__.py
│   ├── admin.py
│   ├── migrations
│   │   └── 0001_initial.py
│   ├── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

3 directories, 12 files
10:18 ~ $
```

## PythonAnywhere에서 가상환경(virtualenv) 생성하기

```
$ cd django-blog
$ virtualenv --python=python3.4 myvenv //가상환경 설치
Running virtualenv with interpreter /usr/bin/python3.4
[...]
Installing setuptools, pip...done.
```

```
$ source myenv/bin/activate //가상환경 활성화
```

```
(myenv) $ pip install django whitenoise //whitenoise 설치
```

```
Collecting django
```

```
[...]
```

```
Successfully installed django-1.8.2 whitenoise-2.0
```

whitenoise?

정적파일(HTML, CSS) 제공을 위해 필요(서버에서는 PC와 환경이 다르기 때문)

collectstatic

장고가 서버에 있는 모든 정적 파일들을 모으도록 지시

```
(myenv) 10:40 ~/django-blog (master)$ python manage.py collectstatic
You have requested to collect static files at the destination
location as specified in your settings:

    /home/akagaeng/django-blog/static

This will overwrite existing files!
Are you sure you want to do this?
Type 'yes' to continue, or 'no' to cancel: yes
```

## PythonAnywhere에서 데이터베이스 생성하기

서버 데이터베이스 초기화

```
(myenv) $ python manage.py migrate
```

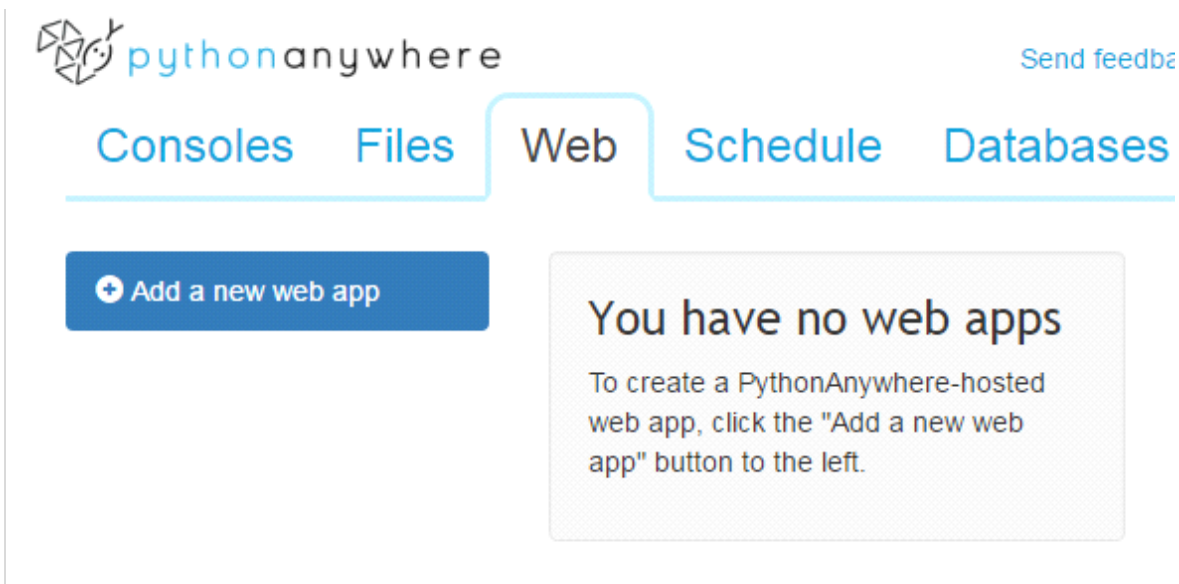
```
Operations to perform:
```

```
[...]
```

```
Applying sessions.0001_initial... OK
```

```
(myenv) $ python manage.py createsuperuser
```

## web app으로 블로그 배포하기



web-add a new web app-manual configuration-Python 3.4

가상환경 경로를 다음과 같이 설정해줌

**Virtualenv:**

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

</home/akagaeng/django-blog/myvenv>

[Start a console in this virtualenv](#)

wsgi 파일 설정

WSGI: 장고는 "WSGI 프로토콜"을 사용하여 작동함. 이 프로토콜은 파이썬을 이용한 웹사이트를 서비스하기 위한 표준으로 PythonAnywhere에서도 지원

WSGI configuration file: [/var/www/akagaeng\\_pythonanywhere\\_com\\_wsgi.py](/var/www/akagaeng_pythonanywhere_com_wsgi.py)

파일 편집



/ > var > www > akagaeng\_pythonanywhere\_com\_wsgi.py

```
1 import os
2 import sys
3
4 path = '/home/akagaeng/django-blog'
5
6 if path not in sys.path:
7     sys.path.append(path)
8
9 os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'
10
11 from django.core.wsgi import get_wsgi_application
12 from whitenoise.django import DjangoWhiteNoise
13 application = DjangoWhiteNoise(get_wsgi_application())
```

```
import os
import sys

path =
'/home/akagaeng/django
-blog'

if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SET
TINGS_MODULE'] =
'mysite.settings'

from django.core.wsgi
import
get_wsgi_application
from whitenoise.django
import
DjangoWhiteNoise
application =
DjangoWhiteNoise(get_w
sgi_application())
```

저장 후 reload한 후,  
akagaeng.pythonanywhere.com 접속하면  
로컬과 같은 화면 나옴.  
~/admin접속하면 관리자모드 진입 가능!

## 6. 페이지 제작

2016년 8월 18일 목요일    오후 9:41

장고 URL

장고는 URLconf(URL configuration) 사용: 장고에서 URL과 일치하는 뷰를 찾기 위한 패턴들의 집합

어떻게 작동되나?

mysite/urls.py

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    # Examples: //주석
    # url(r'^$', 'mysite.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),
    url(r'^admin/', include(admin.site.urls)), //관리자 URL, admin/로 시작하는 모든 url을 view와
    대조해서 찾아냄 => 이를 위해 정규표현식(Regex: Regular Expressions) 사용
]
```

URL 패턴을 만드는 규칙

^ : 문자열이 시작할 때  
\$ : 문자열이 끝날 때  
\d : 숫자  
+ : 바로 앞에 나오는 항목이 계속 나올 때  
( ) : 패턴의 부분을 저장할 때

ex) akagaeng.com/post/12345 를 예로 들면, 12345가 글번호라고 하자

뷰마다 모든 글번호를 매기는 것은 어렵다. 따라서 정규표현식으로 url패턴을 만들어 숫자값과 매칭되게 할 수 있다.

^post/(Wd+)/\$

^post: url이 post/로 시작함

(Wd+): 숫자가 있음

/\$: 숫자 뒤에 / 문자가 있고 이걸로 url이 끝남

mysite/urls.py 수정

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'', include('blog.urls')),
]
```

url(r'', include('blog.urls')), 에서

r: 정규표현식 작성시 붙여주는 문자

include('blog.urls'): <http://localhost:8000/> 으로 들어오는 모든 접속 요청을 blog.urls로 전송해 주  
가명령을 찾음

### blog/urls.py

blog/urls.py 파일 작성

```
from django.conf.urls import url
from . import views
```

장고의 메소드와 blog 애플리케이션에서 사용할 모든 view를 불러옴

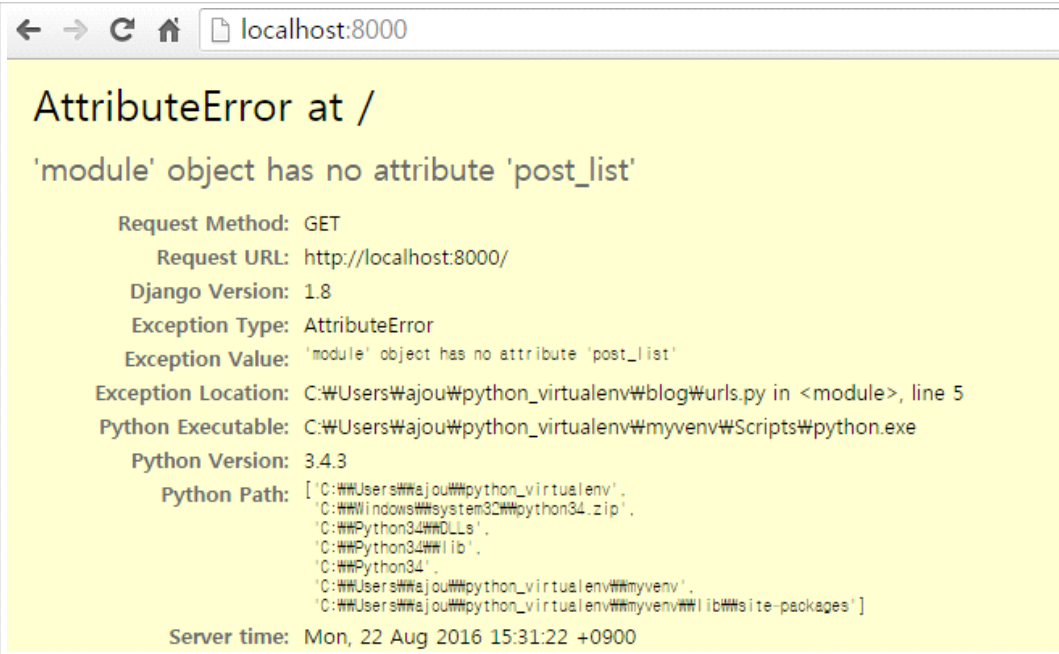
URL 패턴 추가

```
urlpatterns = [
    url(r'^$', views.post_list, name='post_list'),
]
```

post\_list라는 view가 ^\$(문자열이 아무것도 없는 경우) URL에 할당됨

이 패턴은 <http://localhost:8000/~> 에서 ~ 가 없는 경우 장고에게 views.post\_list를 보여주라고 명령함  
name='post\_list'는 URL의 이름.

<http://localhost:8000> 접속시 화면



아직 post\_list view가 없어서 오류남. => view 작성 필요

## Django view 만들기

view?

애플리케이션의 로직(logic)을 넣는 곳

파이썬 메서드

모델에서 필요한 정보를 받아와서 템플릿에 전달하는 역할

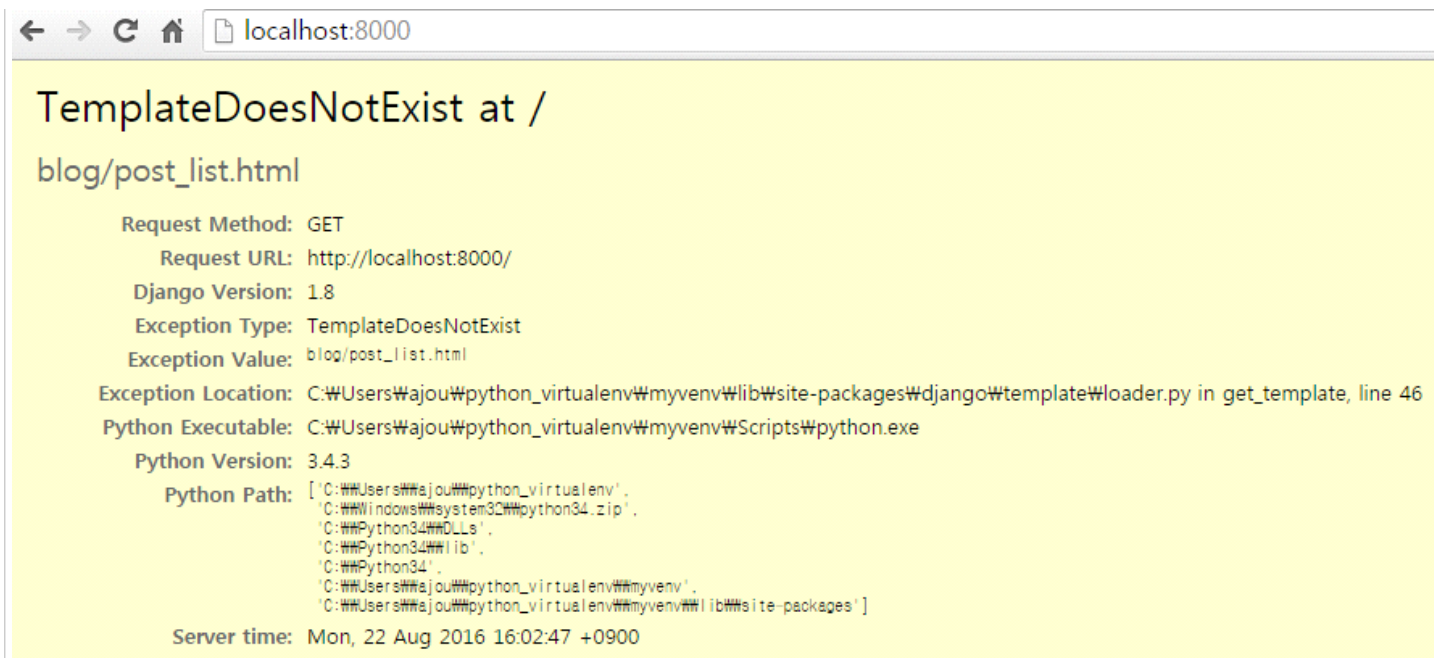
blogs/views.py 파일 수정

```
from django.shortcuts import render
# Create your views here.

def post_list(request):
    return render(request, 'blog/post_list.html', {})
```

post\_list 메서드 생성

요청(request)을 넘겨받아 render 메서드 호출. render 메서드는 request를 받아 'blog/post\_list.html' 템플릿을 호출하고 받은 내용을 return하며 이를 브라우저에 보여줌



방금전과 다른 에러 페이지 출력(템플릿이 존재하지 않는다) => 템플릿 작성 필요!

## 템플릿 작성

템플릿?

서로 다른 정보를 일정한 형태로 표시하기 위해 재사용 가능한 파일(형식/레이아웃)

장고는 템플릿 양식으로 HTML 사용

HTML?

HyperText: 하이퍼링크가 포함된 텍스트

Markup: 브라우저가 문서를 해석하도록 표시(mark)해둠(by tags)

Language

템플릿 저장 장소

blog/templates/blog에 저장

blog/templates/blog/post\_list.html 생성

이후 서버 restart

\$ python manage.py runserver

에러메시지 더이상 나오지 않음

post\_list.html 수정

```
<html>
  <head>
    <title>akagaeng's django page!</title>
  </head>
  <body>
    <div>
      <h1><a href="">akagaeng Django Blog</a></h1>
    </div>
    <div>
      <p>published: 22.08.2016, 16:43</p>
      <h2><a href="">My first post</a></h2>
      <p>contents1 of 1st post</p>
    </div>
    <div>
      <p>published: 14.06.2014, 12:14</p>
      <h2><a href="">My second post</a></h2>
      <p>contents1 of 2nd post</p>
    </div>
  </body>
</html>
```



```
</div>
</body>
```

```
</html>
```



배포하기!

Local

```
$git status //수정된 사항 확인(빨간색 표시)
$git add -A . //현재 디렉터리에서 수정된 사항 모두 포함
$git status //업로드할 파일 확인(녹색표시)
$git commit -m "HTML template added!"
$git push //github에 push(업로드)
```

pythonanywhere

github에서 pull(다운로드) 해오기

```
(myvenv) 07:53 ~/django-blog (master)$ git pull
Username for 'https://github.com': akagaeng
Password for 'https://akagaeng@github.com':
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0
Unpacking objects: 100% (10/10), done.
From https://github.com/akagaeng/django-blog
 a2b6597..df53ae7 master -> origin/master
Updating a2b6597..df53ae7
Fast-forward
 blog/templates/blog/post_list.html | 21 ++++++
 blog/urls.py                       | 7 +++++
 blog/views.py                      | 2 ++
 mysite/urls.py                     | 2 +-
 4 files changed, 31 insertions(+), 1 deletion(-)
 create mode 100644 blog/templates/blog/post_list.html
 create mode 100644 blog/urls.py
(myvenv) 07:54 ~/django-blog (master)$
```

정적 파일 수집

```
(myvenv) 07:53 ~/django-blog (master)$ ls
blog db.sqlite3 manage.py mysite myvenv static
(myvenv) 07:53 ~/django-blog (master)$ python manage.py collectstatic
```

<http://akagaeng.pythonanywhere.com/> 에서도 로컬과 같은 내용 확인 가능

## 7. 장고 ORM과 쿼리셋

2016년 8월 18일 목요일    오후 9:41

장고를 DB에 연결하여 데이터를 저장

쿼리셋(Query Sets)?

전달받은 모델의 객체 목록

DB로부터 데이터를 읽고, 필터링하거나 정렬할 수 있음

장고 셸

```
(myvenv) C:\Users\ajou\python_virtualenv>python manage.py shell
```

```
(InteractiveConsole)
>>>
```

장고 인터랙티브 콘솔

```
>>> from blog.models import Post
```

```
>>> Post.objects.all() //입력했던 모든 글 출력
```

```
>>> from blog.models import Post
>>> Post.objects.all()
[<Post: test1>, <Post: test2>, <Post: test3>]
>>>
```

게시된 글 목록 보임(/admin에서 작성한 글)

-> 파이썬으로 포스팅하기

객체 생성

DB에 새 글 객체 저장

```
>>> Post.objects.create(author=me, title='Sample title', text='Test')
```

이 때, name 'me'를 정의해야 함(정의되어있지 않으면 오류남-뒤에서 정의)

User model 불러옴

```
>>> from django.contrib.auth.models import User
```

DB에서 user가 하는 일?

```
>>> User.objects.all() //User(객체) 전체 보기
```

```
>>> User.objects.all()
[<User: admin>]
```

super user로 등록된 user밖에 없음

슈퍼유저로 등록한 사용자의 인스턴스(instance)를 가져옴

```
>>> me = User.objects.get(username='admin')
```

다시 DB에 새 글 객체 저장

```
>>> Post.objects.create(author=me, title='Sample title', text='Test')
```

```
>>> Post.objects.create(author=me, title='aaa', text='aaa text') //시험삼아 하나 더 추가
```

DB에 저장된 새 글 객체 보기

```
>>> Post.objects.all()
```

```
[<Post: test1>, <Post: test2>, <Post: test3>, <Post: Sample title>]
>>> Post.objects.create(author=me, title='Sample title', text='Test')
```

새로 추가한 제목의 글이 추가 되어 있음을 확인(확인용으로 글 몇 개 더 추가해두기)

## 필터링하기

## 쿼리셋의 중요한 기능

```
>>> Post.objects.filter(author=me) //저자가 me인 사람만 필터링
```

```
>>> Post.objects.filter(title__contains='title') //title에 title이라는 글자가 들어간것만 필터링
```

이 때, title\_\_contains에서처럼 title과 contains 사이에 underscore가 2개 있다. 장고 ORM은 필드 이름(title)과 필터(contains)를 \_\_를 이용하여 구분한다.

```
>>> from django.utils import timezone
```

```
>>> Post.objects.filter(published_date__lte=timezone.now())
```

결과: []

파이썬 콘솔에서 추가한 게시물 확인하기

1) 게시하려는 게시물의 인스턴스 얻기

```
>>> post = Post.objects.get(title="Sample title")
```

2) publish 메서드를 사용하여 글 게시

```
>>> post.publish()
```

3) 다시 게시된 글 목록 가져옴

```
>>> Post.objects.filter(published_date__lte=timezone.now())
```

결과

```
>>> post = Post.objects.get(title="aaa")
>>> post.publish()
>>> Post.objects.filter(published_date__lte=timezone.now())
[<Post: aaa>]
```

## 정렬하기

생성된날짜 순으로 올림차순

```
>>> Post.objects.order_by('created_date')
```

생성된날짜 순으로 내림차순

```
>>> Post.objects.order_by('-created_date') //-만 앞에 붙여주면 됨!
```

## 쿼리셋 연결하기

쿼리셋들을 연결(chaining)

```
>>> Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date')
```

게시날짜로 필터링 + 정렬 한번에 할 수 있어서 복잡한 쿼리도 작성 쉽게 할 수 있음

셸 종료

```
>>> exit()
```

## 템플릿 동적 데이터

블로그 글은 각각 다른 장소에 나누어져 있음

Post 모델: post\_list 파일

post\_list 모델: views.py 파일 ...

html 템플릿에서 글 목록을 보여주기 위해서는?

=> DB에 저장되어 있는 모델(콘텐츠)을 가져와 템플릿에 넣어 보여주는 것이 필요

뷰(view)?

모델과 템플릿을 연결하는 역할

post\_list를 뷰에서 보여주고, 이를 템플릿에 전달하기 위해서는 모델을 가져와야 함

일반적으로 뷰가 템플릿에서 모델을 선택하도록 만들어야 함

blog/views.py

```
from django.shortcuts import render
from .models import Post //새로 추가, '.'은 현재 디렉터리를 의미함
```

```
//동일한 디렉터리 내의 파일 불러올 때에는 models.py에서 .py 생략 가능
def post_list(request):
    return render(request, 'blog/post_list.html', {})
```

Post 모델 불러오기

Post 모델에서 블로그 글 가져오기 위해서는 쿼리셋(QuerySet) 필요

쿼리셋(QuerySet)[Django ORM 참조]

publish\_date 기준으로 정렬하여 블로그 글 목록 보기

shell(>>>)에서 작성했던 내용을 blog/biews.py에 post\_list() 메서드로 작성

blog/biews.py

```
from django.shortcuts import render
from django.utils import timezone
from .models import Post
def post_list(request):
    posts = Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date')
    return render(request, 'blog/post_list.html', {'posts': posts})
```

posts 변수(쿼리셋의 이름) 생성

render함수의 매개변수

request: 사용자의 요청

'blog/post\_list.html': 템플릿

{}: 템플릿 사용을 위한 매개변수 추가( {'posts': posts} 라고 작성)

이제 쿼리셋 설정은 끝

템플릿 파일로 가서 쿼리셋을 보이도록 만들기

## 8. 장고 템플릿(Template)

2016년 8월 23일 화요일 오전 11:43

### 템플릿 태그(template tags)

데이터를 보여주기 위해 장고가 제공하는 기능

원래 브라우저는 파이썬 코드를 이해할 수 없으며, HTML에 파이썬 코드를 직접 넣을 수 없음

또한 HTML은 정적, 파이썬은 동적

장고 템플릿 태그는 파이썬을 HTML로 바꾸어주어 동적인 웹사이트 제작이 가능하도록 해줌

### Post 목록 템플릿 보여주기

장고 템플릿 안에 있는 값 출력하기(변수 이름 안에 중괄호 두 개 안에 넣어서 표시)

```
{{ posts }}
```

### blog/templates/blog/post\_list.html 템플릿 수정

post_list.html 태그	출력
<code>{{ posts }}</code>	<Post: Test title1>, <Post: Test title2>, <Post: Test title3>
<code>{% for post in posts %}</code> <code>    {{ post }}</code> <code>{% endfor %}</code>	Test title1 Test title2 Test title3

<p>Print posts by for statement</p>

    {% for post in posts %}

        <div>

            <p>Published: {{ post.published\_date }}</p>

            <h1><a href="">{{ post.title }}</a></h1>

            <p>{{ post.text|linebreaks }}</p>

        </div>

    {% endfor %}

//post.published\_date: 작성 날짜

//post.title: 제목

//post.text: 내용

//post.text | linebreaks: 파이프 문자 사용하여 텍스트 행이 바뀌면 문단으로 변환해줌

# Test title1

Test Text1

Published: Aug. 23, 2016, 3:51 p.m.

# Test title2

Test Text2

Published: Aug. 23, 2016, 3:54 p.m.

좌측과 같이 태그 작성시 우측과 같이 출력됨

이까지 하고 다시 배포

Local

```
$ git status
```

```
$ git add -A
```

```
$ git status
$ git commit -m "Applied template with for statement!"
$ git push
```

pythonanywhere.com myvenv console

```
(myvenv) ~/django-blog (master)$ git pull
```

화면 리프레시 하여 배포된 것 확인

<http://akagaeng.pythonanywhere.com/admin> 에서 블로그 글 작성하면 글 보임(published date 꼭 추가해야함)

## CSS(Cascading Style Sheet)

HTML과 같은 Markup language로 쓰여진 웹사이트를 나타낼 때 꾸미는 용도로 사용하는 언어

부트스트랩(bootstrap)

웹사이트 개발시 사용하는 가장 유명한 HTML, CSS 프레임워크. 간단하게 꾸밀 수 있음

부트스트랩 설치

blog/templates/blog/post\_list.html의 <head> 안에 아래 링크 넣기

```
<head>

<link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-
theme.min.css">

</head>
```

이것만으로도 깔끔하게 변함

정적 파일

동적 파일이 아닌 것

CSS, 이미지 등

콘텐츠 요청이 없으므로 모든 유저들이 동일한 내용을 볼 수 있음

위치

python\_virtualenv/blog/static 에 디렉터리 만들기(장고가 알아서 인식)

python\_virtualenv/blog/static/css 에 css파일(blog.css) 넣기

blog.css 작성

```
h1 a { //CSS selector: <h1><a ...> ... </a></h1> 과 같은 코드를 선택
    color: #FAAADD; //다음과 같은 색으로 변경하도록 지시
}
```

태그 전체로 지정도 가능하고 class, id로 구분하여 적용하는 것도 가능.

(<http://www.w3schools.com/css/>) 참조

CSS를 HTML에 적용하기

blog/templates/blog/post\_list.html

```
{% load staticfiles %}
<html>
  <head>
    <title>title~</title>
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">
    <link rel="stylesheet" href="{% static 'css/blog.css' %}">
  </head>
```

제목 폰트 바꾸기

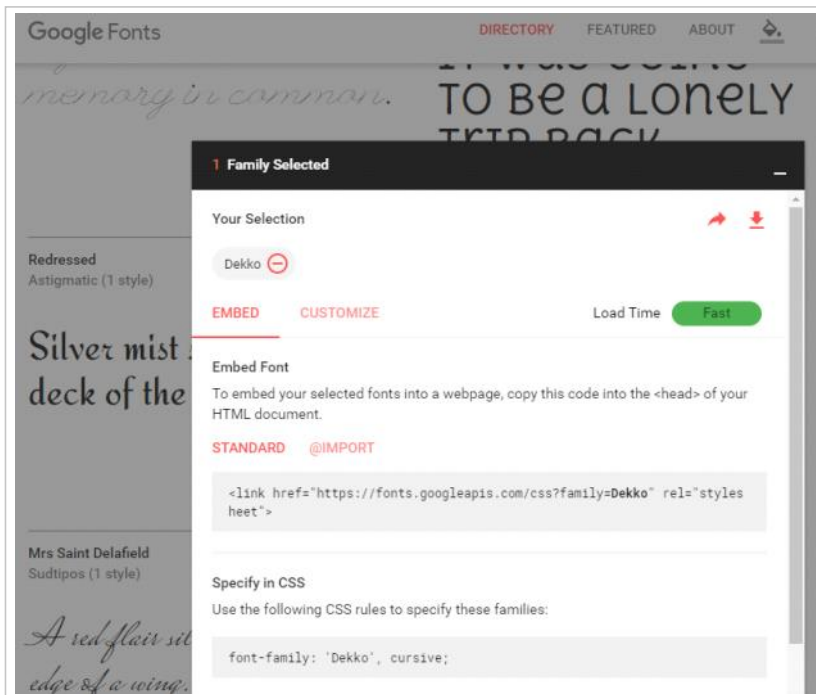
html파일과 css파일 모두 수정해야 함

구글 폰트( <https://fonts.google.com/> )참조

마음에 드는 폰트 선택 후

위쪽에 있는 태그는 html의 <head> 안에 넣고,

아래 쪽에 있는 태그는 css 파일의 셀렉터 안에 배치



blog/templates/blog/post\_list.html

```
<link href="https://fonts.googleapis.com/css?family=Dekko" rel="stylesheet">
```

blog/static/css/blog.css

```
font-family: 'Dekko', cursive;
```

기타 html 및 css코드 작성

## 템플릿 확장(template extension)

템플릿 상속(template inheritance)라고도 불림

웹사이트 내의 다른 페이지에서 HTML의 일부를 동일하게 재사용할 수 있도록 하는 것

각 페이지마다 변경이 필요한 부분만 코드를 작성하게 하는 기능

하나만 수정하면 다른 곳에서도 수정이 됨

### 1. 기본 템플릿(base template) 생성

blog/templates/blog/base.html 생성

(base 템플릿을 어디다 만드느냐는 개발자의 몫-> 설정은 settings.py 내의 TEMPLATES에서)

기존 blog/templates/blog/post\_list.html에 있는 내용을 그대로 base.html에 복사

(<http://pythonstudy.xyz/python/article/312-Django-%ED%85%9C%ED%94%8C%EB%A6%BF-%ED%99%95%EC%9E%A5> 참조)

각 페이지마다 변경/삽입을 통해 달리 보이게 할 부분을 {% block KEYWORD %} ~ {% endblock %} 로 감싼다  
KEYWORD 부분은 용도에 맞게 naming하면 됨

base로 extend하기 위하여 다음과 같은 코드를 post\_list.html의 맨 윗줄에 삽입한다.

```
{% extends 'blog/base.html' %}
```

이는 현재 문서를 blog/base.html까지 확장(extend)한다는 의미로써, base.html은 골격문서가 된다.

또한 base.html이 현재 문서의 "부모" 템플릿이 되며, 현재의 문서는 "자식" 템플릿이 된다.

**django documentation에서는 블록은 많이 설정할수록 좋다고 하여 불복 사용을 권장하고 있다.**

([https://django-document-korean.readthedocs.io/en/old\\_master/topics/templates.html#id9](https://django-document-korean.readthedocs.io/en/old_master/topics/templates.html#id9))

base.html 중 body의 일부 코드

```
{% block page-header %} //block의 이름을 page-header이라고 naming
<h1><a href="">page subject here</a></h1>
<h3>page subsubject here</h3>
<h5><a href="">page description here</a></h5>
{% endblock %}
```

블록 내에 있는 내용은 default로써, 만약 상속받을 페이지에서 page-header block이 설정되지 않았다면 base에 기재된 내용대로 출력된다.

다만, page-header block을 설정하여 그 안에 코드를 삽입하였다면, 삽입한 내용으로 변경되어 출력된다.

<제대로 extend 적용되지 않은 예>

post\_list.html

```
{% extends 'blog/base.html' %}
...
{% block page-header123 %} //block의 이름을 base.html과 달리 설정(설정이 없는 경우와 동일)
<h1><a href="">page subject here</a></h1>
<h3>page subsubject here</h3>
<h5><a href="">page description here</a></h5>
```



```
{% endblock%}
```



base에 입력된 block 이름과 다르거나, block 자체가 설정되어 있지 않은 경우 base.html에 있는 대로 출력된다.

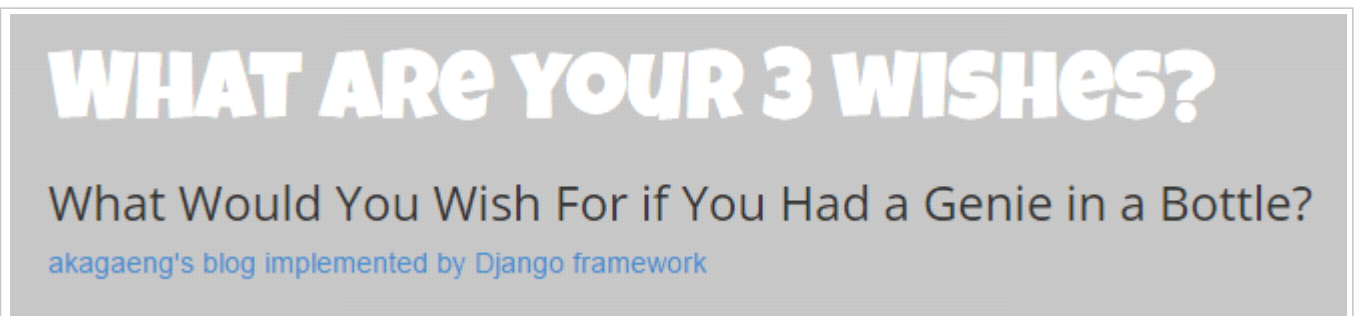
<제대로 extend 적용된 예>

post\_list.html

```
{% extends 'blog/base.html' %}

...

{% block page-header %} //block의 이름을 base.html와 동일하게 설정
    <h1><a href="">page subject here</a></h1>
    <h3>page subsubject here</h3>
    <h5><a href="">page description here</a></h5>
{% endblock%} //가독성을 위해 {% endblock page-header %}와 같이 작성해도 된다.
```



base.html에 적혀있는 내용과 달리 post\_list.html에 작성된 코드대로 변경되어 출력된다.

([https://django-document-korean.readthedocs.io/en/old\\_master/topics/templates.html#id9](https://django-document-korean.readthedocs.io/en/old_master/topics/templates.html#id9) 참조)

## 9. 프로그램 애플리케이션 확장

2016년 8월 24일 수요일    오후 2:02

블로그 게시글이 각 페이지마다 보이도록 만들기  
(Post 모델을 만들었으므로 models.py에 추가할 내용은 없음)

Post에 템플릿 링크 제작

blog/templates/blog/post\_list.html

```
{% for post in posts %}
    <div>
        <div class = "date">
            <p>Published: {{ post.published_date }}</p>
        </div>
        <h5><a href="to be modified">{{ post.title }}</a></h5>
        <p>{{ post.text|linebreaks }}</p>

    </div>
{% endfor %}
```

<h5><a href="to be modified">{{ post.title }}</a></h5>

to be modified 부분을 다음과 같이 수정

```
<h1><a href="{% url 'post_detail' pk=post.pk %}">{{ post.title }}</a></h1>
```

다음과 같은 오류메시지 출력됨

### NoReverseMatch at /

Reverse for 'post\_detail' with arguments '()' and keyword arguments '{'pk': 10}' not found. 0 pattern(s) tried: []

```
Request Method: GET
Request URL: http://localhost:8000/
Django Version: 1.8
Exception Type: NoReverseMatch
Exception Value: Reverse for 'post_detail' with arguments '()' and keyword arguments '{'pk': 10}' not found. 0 pattern(s) tried: []
Exception Location: C:\Users\#ajou\python_virtualenv\myvenv\lib\site-packages\django\core\urlresolvers.py in
_reverse_with_prefix, line 496
Python Executable: C:\Users\#ajou\python_virtualenv\myvenv\Scripts\python.exe
Python Version: 3.4.3
Python Path: ['C:\Users\#ajou\python_virtualenv',
'C:\Windows\system32\python34.zip',
'C:\Python34\DLLs',
'C:\Python34\lib',
'C:\Python34',
'C:\Users\#ajou\python_virtualenv\myvenv',
'C:\Users\#ajou\python_virtualenv\myvenv\lib\site-packages']
Server time: Thu, 25 Aug 2016 12:01:49 +0900
```

### Error during template rendering

오류 해결

```
{% url 'post_detail' pk=post.pk %}
```

{% %}: 장고 템플릿 태그

post\_detail: view 이름(이에 따른 뷰 경로는 blog.views.post\_detail이 됨)

## Post detail URL만들기

urls.py

첫 게시물의 post\_detail 페이지 url이 <http://localhost:8000/post/1> 이 되도록 만들어보자

blog/urls.py 수정

정규식 작성하여 urlpatterns에 추가

blog/urls.py

```
from django.conf.urls import include, url
from . import views

urlpatterns = [
    url(r'^$', views.post_list, name='post_list'),
    url(r'^post/(?P<pk>[0-9]+)/$', views.post_detail, name='post_detail'),
]
```

/\$', views.post\_detail, name='post\_detail')

^post/: post/로 시작

(?P<pk>[0-9]+):

pk 변수에 모든 값(숫자값)을 넣어 뷰로 전송한다.

(pk: primary key의 약자로서 pk라고 변수명을 주로 쓰나 변경해도 무방)

단, 0~9 사이 숫자만 올 수 있으며,

뒤에 +가 있어서 1자리 이상의 숫자가 올 수 있음.

/\$: 마지막에 /로 끝남. 따라서 이 뒤에는 더 이상 문자가 올 수 없음

만약 <http://localhost:8000/post/5> 라고 입력한다면, 장고는 post\_detail 뷰를 찾아 매개변수 pk가 5인 값을 찾아 뷰로 전달한다.

blog/urls.py를 수정하였으나, 다음과 같이 새로운 오류가 나옴(뷰가 없으므로!)

```
AttributeError at /
'module' object has no attribute 'post_detail'
```

뷰 추가

views.py에 다음과 같은 코드를 추가

```
from django.shortcuts import render, get_object_or_404

def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)
    return render(request, 'blog/post_detail.html', {'post': post})
```

Post\_detail 템플릿 만들기

blog/templates/blog/post\_detail.html 생성

```
{% extends 'blog/base.html' %}

{% block content %} //base.html에 블록 추가 지정 필요함
<div class="post">
    {% if post.published_date %} //published_date가 있으면 출력하고 아니면 출력x
```

```
<div class="date">
    {{ post.published_date }}
</div>
{% endif %}
<h1>{{ post.title }}</h1>
<p>{{ post.text|linebreaks }}</p>
</div>
{% endblock content%}
```

이제 제목 클릭시 상세한 내용보기 페이지로 이동하게 됨

다시 배포할 타이밍!

로컬

```
$ git status
$ git add -A .
$ git status
$ git commit -m "Added view and template for detailed blog post as well as CSS
for the site."
$ git push
```

pythonanywhere.com

```
$ cd my-first-blog
$ source myenv/bin/activate
(myenv)$ git pull
[...]
(myenv)$ python manage.py collectstatic
[...]
```

## 10. Django form

2016년 8월 25일 목요일    오후 2:59

폼을 만들어 Post 모델에 적용

blogs/forms.py 생성

```
from django import forms //forms model import
from .models import Post //Post model import
class PostForm(forms.ModelForm): //우리가 만들 form의 이름(PostForm)
    //장고에게 forms.ModelForm이라는 것을 알려줌
class Meta: //이 폼을 만들기 위해서 어떤 model이 쓰여야 하는지 장고에게 알려줌
    model = Post //model이 Post임을 알려줌
    fields = ('title', 'author', 'text',) //필드에 title, author, text 넣어줌
```

base.html

page-header div class에 링크 추가

```
<a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon glyphicon-plus"></span></a>
```

blog/urls.py

아래 구문 추가

```
urlpatterns=[
...
url(r'^post/new/$', views.post_new, name='post_new'),
]
```

blog/views.py

아래와 같이 코드 추가

```
from .forms import PostForm

def post_new(request):
    form = PostForm()
    return render(request, 'blog/post_edit.html', {'form': form})
```

blog/templates/blog/post\_edit.html

```
{% extends 'blog/base.html' %}
{% block content %}
    <h1>New post</h1>
    <form method="POST" class="post-form">{% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="save btn btn-default">Save</button>
    </form>
{% endblock content %}
```

new post 폼 생성 완료

<http://localhost:8000/post/new/>

# New post

**Title:**

**Author:**

**Text:**

현재는 폼 작성 후 글이 전송(get/post)되지 않음

## Form submit(get/post 방식)

blog/view.py 중 post\_new 메서드

```
def post_new(request): //request: 우리가 입력한 데이터. request.POST가 가지고 있음
    //이 때 POST는 post나 Post와 달리 submit 방식 중 POST방식을 의미함
    form = PostForm()
    return render(request, 'blog/post_edit.html', {'form': form})
```

두가지 상황에서의 뷰 처리

- i) 새 글 작성시(처음 페이지에 접속시): 폼이 비어있으므로 폼에 입력된 페이지를 view로 가져올 필요 없음
- ii) 글 수정시: 폼에 입력된 페이지를 view로 가져 와야 함

blog/view.py 중 수정된 post\_new 메서드

```
def post_new(request):
    if request.method == "POST":
        form = PostForm(request.POST)
        if form.is_valid(): //폼에 값이 있고 그 값들이 올바른 경우
            post = form.save(commit=False) //폼에 저장된 내용을 post 에 저장
            //commit=False는 넘겨진 데이터를 바로 Post 모델에 저장하지 말라는 뜻.
            //이후에 author를 추가할 것이므로
            post.author = request.user //저자 정보 저장
            post.published_date = timezone.now() //작성시간 저장
            post.save() //위의 모든 내용들을 Post모델에 저장
            return redirect('post_detail', pk=post.pk) # 원래는 'blog.views.post_detail'이라 되어있으나 오류 발견!
        else:
            form = PostForm()
            return render(request, 'blog/post_edit.html', {'form': form})
```

이제 폼에 글을 입력하고 save 버튼을 누르면 post\_detail.html로 이동

<<이 때, 로그인을 하지 않고 글을 작성할 경우, 오류가 남>>

이에 대한 해결은 뒤에서 다시~

## 폼 수정

blog/templates/blog/post\_detail.html

```
...
<div class="date">
    {{ post.published_date }}
    <a class="btn btn-default" href="{% url 'post_edit' pk=post.pk %}"><span class="glyphicon glyphicon-pencil"></span></a>
</div>
...
```

blog/urls.py의 urlpatterns에 다음 내용을 추가

```
url(r'^post/(?P<pk>[0-9]+)/edit/$', views.post_edit, name='post_edit'),
```

views.py에 다음과 같은 메서드 추가

```
def post_edit(request, pk):
    post = get_object_or_404(Post, pk=pk) //수정하고자하는 Post 모델의 인스턴스로 가져옴
    if request.method == "POST":
        form = PostForm(request.POST, instance=post) //이전에 입력한 데이터 저장
        if form.is_valid():
            post = form.save(commit=False)
            post.author = request.user
            post.published_date = timezone.now()
            post.save()
            return redirect('post_detail', pk=post.pk) # 원래는 'blog.views.post_detail'이라 되어있으나 오류 발견!
        else:
            form = PostForm(instance=post)
    return render(request, 'blog/post_edit.html', {'form': form})
```

수정한 화면



배포

pythonanywhere.com에 배포하면 redirect시 redirect가 제대로 되지 않아

[https://akagaeng.pythonanywhere.com/post/3/edit/blog.views.post\\_detail](https://akagaeng.pythonanywhere.com/post/3/edit/blog.views.post_detail) 와 같이 이동한다.

```
return redirect('blog.views.post_detail', pk=post.pk)
```

해결( 스스로)

```
post_new, post_edit 메서드 내
return redirect('post_detail', pk=post.pk) # 원래는 'blog.views.post_detail'이라 되어있으나 오류 발견!
이를 통해 해결
```

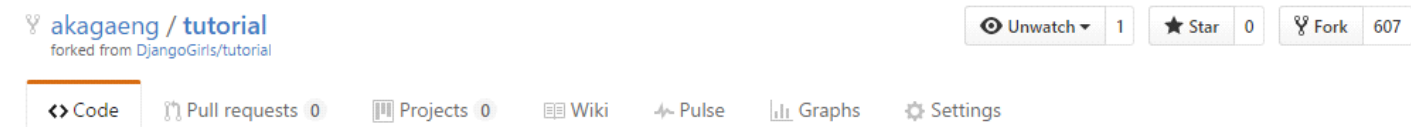
해결 내용 django girls github에 commit

1. django girls 프로젝트 fork



우측 Fork 버튼 누름

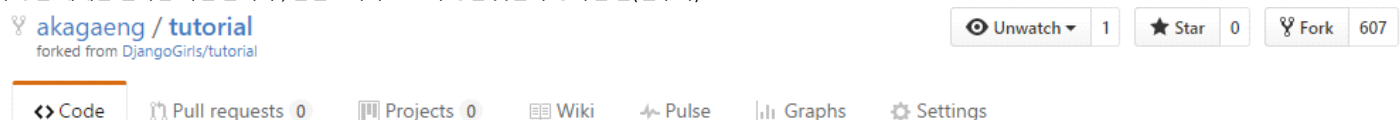
2. 코드 수정하여 브랜치 생성



This is a tutorial we are using for Django Girls workshops <http://tutorial.djangogirls.org/> — Edit

나의 저장소에 fork된 내용을 보고 수정

수정할 때에는 원하는 파일 클릭 후, 연필표시 누르고 수정할 곳을 수정하면 됨(맨우측)



Code Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master tutorial / ko / django\_forms / README.md

Find file Copy path

dennybiasioli translations: removed unused import of django.conf.urls.include

24a5804 on 5 Jul

4 contributors

Executable File 403 lines (275 sloc) 18.7 KB

Raw Blame History

3. 수정이 완료되면 Preview changes 확인하고, Commit changes 작성.

(이 경우에는 master에 직접 commit하는 옵션 말고, new branch를 만드는 것으로 선택할 것!



## Commit changes

Update

Add an optional extended description...

- ☒ Commit directly to the master branch.
- ☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

4. django girls tutorials에 pull request  
진행중..

Code Issues 46 Pull requests 18 Projects 0 Wiki Pulse Graphs

## redirect target fixes #848

Open akagaeng wants to merge 2 commits into DjangoGirls:master from akagaeng:akagaeng-patch-1

Conversation 2 Commits 2 Files changed 1

akagaeng commented on 26 Aug • edited

Modified to correctly run when deployed on pythonanywhere.com  
it redirected to wrong url: (yourID.pythonanywhere.com/post/new/blog.views.post\_detail).  
However, after this modification it redirects to correct url:  
(yourID.pythonanywhere.com/post/post\_no)

1

pythonanywhere.com에 배포시 제대로 redirect하지 못하는 문제 해결

cf53265

patjouk commented 5 hours ago Django Girls member

Hi! Thanks for this PR! Can you also fix instructions in the text? Thanks! 🍷

1

akagaeng commented 2 hours ago

Thank you for your advice, patjouk! Done it.

추가로 PR (Pull Request) 요청해서 추가 commit -> Merge됨.

2 commits (+3, -3), (+1, -1)

[https://github.com/akagaeng/DjangoGirls\\_tutorial/commit/cf5326560dba4b97329ac139d813e7a5984ab7c2](https://github.com/akagaeng/DjangoGirls_tutorial/commit/cf5326560dba4b97329ac139d813e7a5984ab7c2)

[https://github.com/akagaeng/DjangoGirls\\_tutorial/commit/57e26f27d944fb53c5c32585bfce0b9ddd932959](https://github.com/akagaeng/DjangoGirls_tutorial/commit/57e26f27d944fb53c5c32585bfce0b9ddd932959)

### pythonanywhere.com에 배포시 제대로 redirect하지 못하는 문제 해결

Browse files

원래 코드는 return redirect('blog.views.post\_detail', pk=post.pk)와 같이 되어있었는데,  
로컬에서는 제대로 동작하나 이를 pythonanywhere.com에 배포하면 새글작성/글수정후 제출시  
yourID.pythonanywhere.com/post/new/blog.views.post\_detail 와 같이 이동하며 redirect가 제대로 동작하지 않았습니다.



## pythonanywhere.com에 배포시 제대로 redirect하지 못하는 문제 해결

[Browse files](#)

원래 코드는 `return redirect('blog.views.post_detail', pk=post.pk)`와 같이 되어있었는데, 로컬에서는 제대로 동작하나 이를 pythonanywhere.com에 배포하면 새글작성/글수정후 제출시 yourID.pythonanywhere.com/post/new/blog.views.post\_detail 와 같이 이동하며 redirect가 제대로 동작하지 않았습니다. 이에 `return redirect('post_detail', pk=post.pk)`로 수정하여 `urls.py`의 `url(r'^post/(?P<pk>[0-9]+)/$', views.post_detail, name='post_detail')`에서 지정한 name인 'post\_detail'와 일치시킨 결과, pythonanywhere.com에 배포한 블로그 어플리케이션에서도 원하는 url로 이동하며 새글작성/글수정 기능이 제대로 수행됩니다.

akagaeng-patch-1 (#848)

akagaeng committed on GitHub on 26 Aug

1 parent 37d8a8a commit cf5326560dba4b97329ac139d813e7a5984ab7c2

Showing 1 changed file with 3 additions and 3 deletions.

Unified Split

6 ko/django\_forms/README.md


<> View

```
@@ -213,7 +213,7 @@ from django.shortcuts import redirect
213 213 취 코드를 여러분의 파일 맨 취에 추가하세요. 그리고 새로 작성한 글을 볼 수 있도록 'post_detail' 페이지로 가라고 수정합니다.
214 214
215 215 python
216 216 - return redirect('blog.views.post_detail', pk=post.pk)
216 216 + return redirect('post_detail', pk=post.pk)
217 217
218 218
219 219 'blog.views.post_detail'은 우리가 이동해야 할 뷰의 이름이므로 *post_detail 뷰* 는 'pk'변수가 필요한 거 기억하고 있겠죠? 'pk=post.pk'를 사
동해서 뷰에게 값을 넘겨줄 거예요. 여기서 'post'는 새로 생성한 블로그 글이예요.
@@ -229,7 +229,7 @@ def post_new(request):
229 229     post.author = request.user
230 230     post.published_date = timezone.now()
231 231     post.save()
232 232 - return redirect('blog.views.post_detail', pk=post.pk)
232 232 + return redirect('post_detail', pk=post.pk)
233 233
234 233 else:
234 234     form = PostForm()
235 235     return render(request, 'blog/post_edit.html', {'form': form})
@@ -308,7 +308,7 @@ def post_edit(request, pk):
308 308     post.author = request.user
309 309     post.published_date = timezone.now()
310 310     post.save()
311 311 - return redirect('blog.views.post_detail', pk=post.pk)
311 311 + return redirect('post_detail', pk=post.pk)
312 312
313 312 else:
313 313     form = PostForm(instance=post)
314 314     return render(request, 'blog/post_edit.html', {'form': form})
```

## fixed an instruction in the text

[Browse files](#)

modified from "blog.views.post\_detail" to "post\_detail" in the text as well

 akagaeng-patch-1 (#848)

 akagaeng committed on GitHub on 21 Oct

1 parent [cf53265](#)

commit [57e26f27d944fb53c5c32585bfce0b9ddd932959](#)

 Showing **1 changed file** with **1 addition** and **1 deletion**.

Unified Split

2  ko/django\_forms/README.md

  View

⌵	@@ -216,7 +216,7 @@ from django.shortcuts import redirect
216	216
217	217
218	218
219	-`blog.views.post_detail`은 우리가 이동해야 할 뷰의 이름이에요 <i>*post_detail</i> <del>##</del> 는 `pk`변수가 필요한 거 기억하고 있겠죠? `pk=post.pk`를 사용해서 뷰에게 값을 넘겨줄 거예요. 여기서 `post`는 새로 생성한 블로그 글이에요.
	+`post_detail`은 우리가 이동해야 할 뷰의 이름이에요 <i>*post_detail</i> <del>##</del> 는 `pk`변수가 필요한 거 기억하고 있겠죠? `pk=post.pk`를 사용해서 뷰에 게 값을 넘겨줄 거예요. 여기서 `post`는 새로 생성한 블로그 글이에요.
220	220
221	221
222	222
⌵	잘 했어요. 너무 설명이 길어졌네요. 이제 <i>*view*</i> 의 전체 코드를 확인할게요.

# 11. 기타 새로운 기능 추가

2016년 8월 25일 목요일    오후 9:50

편의기능

제목 누르면 root로 이동

base.html

```
<h1><a href="{%url 'post_list' %}">What are your 3 wishes?</a></h1>
```

post\_detail.html에서 기능 추가

글 삭제

목록으로 이동

아이콘: bootstrap 이용( <http://getbootstrap.com/components/> )

blog/templates/blog/post\_detail.html 에 list, delete 아이콘 배치

```
{% block content %}
<div class="post">
  {% if post.published_date %}
    <div class="date">
      {{ post.published_date }}
      <a class="btn btn-default" href="{% url 'post_list' %}"><span class="glyphicon glyphicon-list"></span></a>
      <a class="btn btn-default" href="{% url 'post_edit' pk=post.pk %}"><span class="glyphicon glyphicon-pencil"></span></a>
      <a class="btn btn-default" href="{% url 'post_remove' pk=post.pk %}"><span class="glyphicon glyphicon-remove"></span></a>
    </div>
  {% endif %}
</div>
```

blog/urls.py

```
url(r'^post/(?P<pk>\d+)/remove/$', views.post_remove, name='post_remove'),
```

blog/views.py

```
def post_remove(request, pk):
    post = get_object_or_404(Post, pk=pk)
    post.delete()
    return redirect('blog.views.post_list')
```

post\_edit.html에서 기능 추가

```
<div class = bottom_btn>
  <a class="btn btn-default" href="{% url 'post_list' %}"><span class="glyphicon glyphicon-list"></span></a>
  <button type="submit" class="btn btn-default"><span class="glyphicon glyphicon-ok"></span></button>
  <a class="btn btn-default" href="{% url 'post_list' %}"><span class="glyphicon glyphicon-remove"></span></a>
</div>
```

## 보안 기능 추가

권한 있는 자만 글 작성/수정/삭제 가능하도록 기능 추가

blog/views.py

```
from django.contrib.auth.decorators import login_required
```

각 메소드 앞에 @login\_required라고 기재

```
@login_required
def post_new(request):
    [...]

@login_required
def post_edit(request):
    [...]
```

```
@login_required
def post_remove(request):
    [...]
```

글 작성/수정/삭제가 가능하다면 localhost:8000/admin/logout 하여 로그아웃 후 다시 시도해본다.

이 경우 다음과 같은 페이지가 나옴

## Page not found (404)

**Request Method:** GET  
**Request URL:** http://localhost:8000/accounts/login/?next=/post/new/

Using the URLconf defined in mysite.urls, Django tried these URL patterns, in this order:

1. ^\$ [name='post\_list']
2. ^post/(?P<pk>[0-9]+)/\$ [name='post\_detail']
3. ^post/new/\$ [name='post\_new']
4. ^post/(?P<pk>[0-9]+)/edit/\$ [name='post\_edit']
5. ^post/(?P<pk>[0-9]+)/remove/\$ [name='post\_remove']
6. ^admin/

The current URL, accounts/login/, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change standard 404 page.

decorator는 login page로 redirect해주는데, 현재 login page가 없기 때문에 page not found(404) error가 발생한다.

## 로그인 페이지 작성

mysite/urls.py

파란색 칠한 부분 추가

```
from django.conf.urls import include, url
import django.contrib.auth.views

from django.contrib import admin
admin.autodiscover()

urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^accounts/login/$', django.contrib.auth.views.login, name='login'),
    url(r'', include('blog.urls')),
]
```

로그인페이지 작성

blog/templates/registration/login.html

```
{% extends "blog/base.html" %}
{% block content %}
{% if form.errors %}
<p>Your username and password didn't match. Please try again.</p>
{% endif %}
<form method="post" action="{% url 'login' %}">
{% csrf_token %}
<table>
<tr>
<td>{{ form.username.label_tag }}</td>
<td>{{ form.username }}</td>
</tr>
<tr>
```

```

        <td>{{ form.password.label_tag }}</td>
        <td>{{ form.password }}</td>
    </tr>
</table>

<input type="submit" value="login" />
<input type="hidden" name="next" value="{{ next }}" />
</form>
{% endblock %}

```

글 작성/수정/삭제 요청시에 다음과 같은 페이지 나옴

mysite/settings.py 에 아래 내용 추가.

```
LOGIN_REDIRECT_URL = '/'
```

로그인하면 최상위 레벨 index로 이동

## 레이아웃 개선

로그인한 경우 표시하기

blog/templates/blog/base.html

```

<body>
    <div class="page-header">
        {% if user.is_authenticated %} //로그인된 경우 글쓰기 버튼 보이기
        <a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon glyphicon-plus"></span></a>
        {% else %} //로그인되지 않은 경우 자물쇠 버튼 보이기
        <a href="{% url 'login' %}" class="top-menu"><span class="glyphicon glyphicon-lock"></span></a>
        {% endif %}
        <h1><a href="{% url 'blog.views.post_list' %}">Django Girls</a></h1>
    </div>
    <div class="content">
        <div class="row">
            <div class="col-md-8">
                {% block content %}
                {% endblock %}
            </div>
        </div>
    </div>
</body>

```

blog/templates/blog/post\_detail.html

```

{{ post.published_date }}
<a class="btn btn-default" href="{% url 'post_list' %}"><span class="glyphicon glyphicon-list"></span></a> //리스트는 항상 보여줌
{% if user.is_authenticated %} //로그인된 경우에만 수정/삭제버튼 보임
    <a class="btn btn-default" href="{% url 'post_edit' pk=post.pk %}"><span class="glyphicon glyphicon-pencil"></span></a>
    <a class="btn btn-default" href="{% url 'post_remove' pk=post.pk %}"><span class="glyphicon glyphicon-remove"></span></a>
{% else %}
{% endif %}

```

blog/templates/blog/post\_edit.html

```

<a class="btn btn-default" href="{% url 'post_list' %}"><span class="glyphicon glyphicon-list"></span></a> //리스트는 항상 보임
{% if user.is_authenticated %} //로그인된 경우에만 수정/삭제버튼 보임 (실제로는 로그인되지 않은 경우는 post_edit.html에 접근도 불가함)
    <button type="submit" class="btn btn-default"><span class="glyphicon glyphicon-ok"></span></button>
    <a class="btn btn-default" href="{% url 'post_list' %}"><span class="glyphicon glyphicon-remove"></span></a>
{% else %}

```

```
{% endif %}
```

## 로그인시 hello message 출력 및 로그아웃 만들기

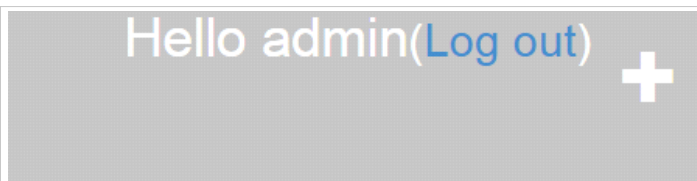
blog/templates/blog/base.html

```
<div class="page-header">
  {% if user.is_authenticated %}
  <a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon glyphicon-plus"></span></a>
  <p class="top-menu">Hello {{ user.username }}<small><a href="{% url 'logout' %}">Log out</a></small></p>
  {% else %}
  <a href="{% url 'login' %}" class="top-menu"><span class="glyphicon glyphicon-lock"></span></a>
  {% endif %}
```

mysite/urls.py에 django.contrib.auth.views.logout view 추가해야 함

```
from django.conf.urls import include, url, patterns
from django.contrib.auth import views
from django.contrib import admin
admin.autodiscover()
urlpatterns = patterns('',
    url(r'^admin/', include(admin.site.urls)),
    url(r'^accounts/login/$', django.contrib.auth.views.login, name='login'),
    url(r'^accounts/logout/$', django.contrib.auth.views.logout, name='logout', kwargs={'next_page': '/'}),
    url(r'', include('blog.urls')),
)
```

로그인시 다음과 같이 페이지에 출력됨



일단 여기까지만 구현하고 배포!

## 12. 나중에 추가할 사항

2016년 8월 31일 수요일    오후 5:25

일단은 기본적으로 학습을 위한 내용만을 위주로 실습하였음  
향후 실제 필요한 기능 구현을 위하여

1. 저자만 게시글 수정/삭제만 가능하도록
2. admin은 모든 글에 대한 수정/삭제 가능하도록
3. 댓글 기능 추가
4. 회원가입기능
5. ...

등 기능 추가 구현할 예정

+ Heroku로 배포하기!도 해볼 예정

<http://heroku.com>